**R O B O T R O N**

**Mikrorechnerbausatz Z 1 0 1 3**

## Bedienungsanleitung

**-Die Bedienungsanleitung und das Handbuch fuer den Mikrorechnerbausatz Z 1013 beziehen sich im wesentlichen auf die Variante Z 1013.01. Fuer die Variante Z 1013.12 ergeben sich auf Grund des begrenzten Arbeitsspeicher von 1 KByte entsprechende Einschraenkungen.**

**-Bei Loetarbeiten, ausser Anschluss der Tastatur entsprechend Montageanleitung, sowie bei Schaeden durch falsche Beschaltung erlischt der Garantieanspruch.**

**-Bei Postversand ist der Z 1013 transportsicher zu verpacken.**

# 1. Bedienungsanleitung

## 1.1. Vorstellung des MRB Z 1013

Zur Grundausbaustufe des Mikrorechnerbausatzes gehoeren eine bestueckte Leiterplatte im Format 215 x 230mm, eine Folienflachtastatur mit den Abmessungen 80 x 160 mm sowie einiges Zubehoer, wie ein Stueck Bandkabel und ein Paar Flachansteckhuelsen.

Die Leiterplatte enthaelt einen kompletten funktionstuechtigen Mikrorechner auf der Basis des Mikroprozessors U 880 mit allen Steuerungen fuer das Betreiben von Baugruppen und Geraeten, die fuer die Arbeit mit der Grundausbaustufe notwendig sind.

Als erstes benoetigt man ein Eingabegeraet, um dem Rechner etwas mitteilen zu koennen. Dazu dient die Tastatur in Verbindung mit der Ein-/Ausgabesteuerung. Damit auch der Rechner dem Bediener etwas mitteilen kann, braucht man ein Datenanzeigegeraet. In unserem Fall ermoeglicht eine Bildschirmsteuerung den Anschluss eines handelsueblichen Fernsehgeraetes. Sollen nun die dem Rechner mitgeteilten Daten bzw. die von ihm errechneten Daten beim Abschalten der Stromversorgung erhalten bleiben, muessen sie auf ein Magnetband gerettet werden. Deshalb enthaelt die Grundausbaustufe auch eine entsprechende Steuerung und Anschlussmoeglichkeit fuer Magnetbandgeraete.

Die ungefaehre Lage dieser Baugruppen auf der Leiterplatte sowie die Lage der Anschlusspunkte fuer externe Geraete koennen der Abbildung 1.1 entnommen werden. Diese Darstellung ist nicht ganz exakt. Sie ist eigentlich als Groborientierung gedacht. Die genaue Zuordnung der Schaltkreise zu den einzelnen Funktionsgruppen ist fuer das Betreiben der Grundausbaustufe ja auch nicht unbedingt noetig. Diese koennen Sie aus dem Belegungsplan und den Stromlaufplaenen, die Sie im Anhang des Handbuches finden, entnehmen. Was Sie fuer die Inbetriebnahme tun muessen und fuer die Arbeit mit der Grundausbaustufe wissen sollten, erfahren Sie in den naechsten Abschnitten. Es werden dann bereits Begriffe benutzt werden, die Kenntnisse auf dem Gebiet der Mikrorechentechnik voraussetzen. Lassen Sie sich dadurch nicht entmutigen. Fuehren Sie trotzdem die angewiesenen Arbeiten aus und lernen Sie mit dem Rechner umgehen.

## 1.2. Inbetriebnahme des MRB Z 1013

### 1.2.1. Anschliessen der Stromversorgung

Die Inbetriebnahme des Geraetes erfordert als erstes die Realisierung der Stromversorgung. Um Ihnen das zu erleichtern, befindet sich auf der Leiterplatte der Grundausbaustufe der wesentliche Teil der Stromversorgungsschaltung: die Gleichrichtungs-, Glaettungs- und Regelschaltung fuer alle benoetigten Spannungen. Dies sind die Betriebsspannungen +5 V (5P), +12 V (12P) und -5 V (5N).

Sie muessen dieser Schaltung noch eine Wechselspannung zufuehren. Diese Spannung muss im Bereich von 11 V bis 12 V liegen. Die Leistungsaufnahme betraegt bis zu 20 W. Diese Wechselspannung kann einem Schutztransformator nach TGL 200-1766 (Schutzkleinspannung) entnommen werden. Ausreichend dafuer sind handelsuebliche Transformatoren.
Bei einer Eigenanfertigung hat unbedingt die Abnahme durch einen Fachmann zu erfolgen. Es ist beim Einsatz auf folgendes zu achten:

Zum sicheren Schutz vor zu hohen Beruehrungsspannungen ist der Transformator mit einem Gehaeuse zu verkleiden, dessen Ausfuehrung einer zulaessigen Schutzmassnahme nach TGL 200-0602 BL. 3 entspricht.

Weiterhin muss der Transformator primaerseitig mit einem Funkentstoerkondensator nach TGL 11840 (250 V, 100 nF und 2 x 2500 pF ) abgeblockt werden, damit eine hochfrequente Abstrahlung von Stoerspannungen ueber das 220 V - Netz unterbunden wird.

Die Verbindung des Transformators mit der Leiterplatte geschieht mit Hilfe der beigelegten Plastaderleitung und der Flachansteckhuelsen, die an die Leitung angeloetet werden.

Achtung: Die Huelsen mit Isolierschlauch oder Isolierband so isolieren, dass sie bei gegenseitiger Beruehrung oder beim zufaelligen Aufliegen auf der Leiterplatte keinen Kontakt geben.

Vor den Anlegen der Stromversorgung muessen Sie unbedingt noch folgenden Hinweis beachten: Legen Sie die Leiterplatte auf eine nichtleitende Unterlage oder schrauben Sie diese mit Hilfe der an Rand der Platte befindlichen Bohrungen und mit Abstandshuelsen auf eine Grundplatte. Ansonsten koennen Kurzschluesse auftreten, die zur Zerstoerung des Rechners fuehren wuerden.

Haben Sie alle Hinweise beachtet, stecken Sie die Huelsen an die Flachansteoker X3 (s. Abb. 1.1) an. Ihr Rechner arbeitet bereits.

## 1.2.2. Anschluss eines Fernsehgeraetes

Jetzt wollen wir ein Datensichtgeraet anschliessen. Dazu benoetigen Sie ein Fernsehgeraet beliebigen Typs und ein handelsuebliches Koaxialkabel mit Koaxialsteckern an beiden Enden. Dieses Kabel stecken Sie in die Antennenbuchse fuer den VHF-Bereich Ihres Fernsehgeraetes. Auf der Leiterplatte des/ vor Ihnen liegenden Z 1013 befindet sich an der Abschirmung des HF-Modulators (s. Abb. 1.1) eine Koaxialbuchse. An dieser Stelle wird die von der Bildschirmsteuerung erzeugte Bildinformation in Form eines normgerechten Fernsehsignals zur Verfuegung gestellt. Hier stecken Sie das andere Ende des Verbindungskabels an.

An dieser Stelle ist ein Hinweis notwendig, dem Sie unbedingt Folge zu leisten haben. Wenn Sie den MHB Z 1013 mit ihrem Fernseher verbinden, darf dies nur zur persoenlichen Nutzung und nur mit dem dafuer vorgesehenen Verbindungskabel geschehen. Anderes missbraeuchliches Betreiben wird entsprechend Paragraph 63 des Gesetzes ueber das Post- und Fernmeldewesen geahndet.

Nach Herstellen der Verbindung muessen Sie noch den Kanal 3 einstellen, bis ein scharfes, stehendes Bild entsteht. Jetzt sehen Sie ein quadratisches Bild. Wenn Sie sich das etwas genauer ansehen, werden Sie erkennen, dass sich das Bild aus einzelnen Zeichen zusammensetzt, Sie werden weiter feststellen, dass diese Zeichen in 32 Zeilen angeordnet sind, wobei in einer Zeile wiederum 32 Zeichen getrennt werden koennen.

Aus den Vergleich einzelner Zeichen ist zu ersehen, dass ein Zeichen nie aus mehr als 8 x 8 Bildpunkten besteht. Welche Zeichen das im einzelnen sind, die mit der Grundausbaustufe auf dem Bildschirm abgebildet werden koennen, ist der Anlage 7 des Handbuches zu entnehmen. Einen grossen Teil koennen Sie auch unmittelbar mit der Tastatur zur Darstellung bringen. Lesen Sie dazu weiter.

## 1.2.3. Grundzustand des MRB Z 1013

**Betaetigen Sie jetzt die RESET-Taste auf der Leiterplatte. Es wird dann der Bildschirm geloescht und am oberen Bildrand erscheint die Ausschrift "robotron Z 1013/2.02" und in der naechsten Zeile ein Doppelkreuz als Zeichen einer ordnungsgemaessen Funktion, sowie nach einer Luecke ein volles Kaestchen.**

**Anzeige des Grundzustandes:**

```
robotron Z 1013/2.02
# _
```

**Die Ausschrift in der ersten Zeile zeigt immer den Grundzustand des Rechners an. Das Doppelkreuz wird als Quittungs- oder Promptsymbol bezeichnet und bedeutet, dass der Rechner jetzt auf eine Eingabe von der Tastatur wartet. Das nachfolgende Zeichen, die Luecke, nennt man Leerzeichen oder "Space".**

**Das volle Zeichen *(hier '_')* wird hier als Kursor genutzt. Der Kursor zeigt immer die Position auf dem Bildschirm an, wo das naechste einzugebende Zeichen dargestellt wird.**

**Das Gesagte wird leichter verständlich, wenn wir nun die Taststatur anschliessen und damit umgehen lernen.**

## 1.2.4. Anschluss der Tastatur

### 1.2.4.1. Montageanleitung

**Vor Ausfuehrung der Montage beachten Sie bitte unbedingt die Hinweise in 1.4., um die Reparaturfaehigkeit des Z 1013 zu erhalten! Fuer den Anschluss der Tastatur entnehmen Sie aus der Verpakkung die Folienflachtastatur und das Stueck Bandkabel. Dann vereinzeln Sie die Adern der Bandleitung in einer Laenge von ca. 3 cm auf beiden Seiten, entfernen dann jeweils ca. 5 mm die Isolierung und verzinnen die Enden. An einer Seite sind die verzinnten Draehte dann auf 1 bis 2 mm zu kuerzen und unter Verwendung von Loetzinn mit Kolophonium entsprechend der Abb. 1.2 auf der Rueckseite der Tastatur (Draehte nicht in die Bohrungen in Tastaturplatine stecken!) anzuloeten. An der anderen Seite des Bandkabels kuerzen Sie die Draehte auf 2 bis 3 mm, beten diese auf der Unterseite der Leiterplatte an den vorgeschriebenen Loetaugen (nicht am Pruefkamm!) an. Dazu muss sich der Z 1013 im stromlosen Zustand befinden.**

**Achtung! Nur Loetkolben mit max. 30 Watt Heizleistung bei max. 3 s Loetdauer verwenden.**

**Es empfiehlt sich, an den Loetstellen fuer eine Zugentlastung zu sorgen, um Leitungsbruch zu vermeiden. Sollten Sie Ihren MRB Z 1013 auf einer Grundplatte aufgeschraubt haben, Ist es ratsam, die Tastatur ebenfalls darauf zu befestigen, so dass haeufiges Bewegen des Kabels vermieden wird. Ist diese Arbeit beendet, lesen Sie bitte weiter.**

## 1.2.4.2. Benutzung der Z 1013 Tastatur

Schauen Sie sich jetzt einmal die Tastatur etwas genauer an. Sie sehen dann, dass die oberen drei Tastenreihen alle eine mehrfache Beschriftung tragen und die unteren nur eine einfache (s. Abb. 1.3).

Durch die Organisation der Tastatur in vier Zeilen und acht Spalten koennten theoretisch 32 verschiedene Tasten rea1isiert werden. Fuer eine alphanumerische Tastatur ist das aber zu wenig. Aus diesem Grund wurden einige Tasten mit einer Umschaltfunktion belegt (Shift: S1, S2, S3 und S4), damit sind die anderen Tastem meherfach nutzbar. Diese Mehrfachbelegung ist auf dem jeweilgen Tastenfeld angegeben.

Eine Besonderheit der Folientastatur ist die kaum wahrnehmbare Ausloesung des gewuenschten Zeichens. Deshalb muss man schon sehr genau die Reaktion den Mikrorechners verfolgen, um die erfolgreiche Betaaetigung der Tasten eindeutig zu registrieren. Wenn man sich an die Verwendung der Folienfalchtastatur gewoehnt hat, tritt dieser Nachteil kaum noch in Erscheinung

Befindet sich der Rechner im Grundzustand, koennen Sie jetzt den Umgang mit der Tastatur ueben:

1. Betaetigen Sie der Reihe nach oben links beginnend alle Tasten. Was beobachten Sie? Zunaechst erscheinen auf dem Bildschirm 24 Zeichen.

   ```
   #  @ABCDEFGHIJKLMOPQRSTUVW
   ```

   Die Zeichen werden immer an der Stelle abgebildet, wo vorher der Kursor stand. Beim Druecken von S1 bis S4 passiert nichts auf dem Bildschirm, aber bei  <--  bewegt sich der Kursor eine Stelle nach links. Nach Druecken von ' ' wird an diese Stelle ein Leerzeichen geschrieben und nach  --> wandert der Kursor eine Position nach rechts. Das Betaetigen der Taste  Ent  (Enter) bewirkt die Abbildung eines Fragezeichens (?), da die Zeichenkette in den internen Code umgewandelt und fuer den Rechner ohne Sinn ist; sowie eines Doppelkreuzes und Kursor, als Aufforderung einer erneuten Eingabe.

2. Betaetigen Sie jetzt die Taste  S1  und gleichzeitig die Tasten der oberen drei Reihen in der gleichen Reihenfolge wie oben und anschliessend  Ent  . Es erscheint das Bild

   ```
   # XYZ[/]^_0123456789:;<=>?
  ? # _
   ```

3. Druecken Sie jetzt  S3  und die Tasten wie oben. Das Fernsehbild sieht nun folgendermassen aus

   ```
   # 'abcdefghijklmnopqrstuvw
  ? # _
   ```

4. Es werden  S2  und wieder die Tasten wie oben betaetigt.

   ```
   # xyz{|}~_!"#$%&'()*+,-./
   ```

5. S4 hat unter den Shift-Tasten wieder eine besondere Bedeutung:

   S4 T Bildschirm geloescht. Kursor oben links auf dem Bildschirm

   S4 U wirkt wie Enter. Es erscheint ? # _

   S4 P Kursor bewegt sich nach links.

   S4 Q Kursor bewegt sich nach rechts.

   S4 G Die Belegung der Tastatur wird geaendert. Wieder holt man jetzt die Uebungen 1 bis 4, werden nicht mehr die alphanumerischen Zeichen, sondern Grafikzeichen abgebildet.

   S4 A Damit wird wieder in den Alpha-Modus umgeschaltet, d. h. die Bedeutung der Tasten ist wieder die urspruengliche.

## 1.2.5. Anschluss eines Magnetbandgeraetes

Mit dem Magnetbandgeraet koennen Sie Informationen (z. B. Programme), die Sie in den Rechner eingeben, speichern und wieder einlesen.
Sie koennen die auf der Kassette aufgezeichneten Programme aufbewahren und spaeter, wenn Sie diese Programme wieder verwenden moechten, von der Kassette in den Speicher des MRB Z 1013 laden.
Als Magnetbandgeraet koennen Sie sowohl Kassettenmagnetbandgeraete als auch Spulentonbandgeraete verwenden. Voraussetzung ist

- das Vorhandensein einer kombinierten Aufnahme-/Wiedergabebuchse mit einer Kontaktbelegung nach TGL 28200/05:
  Kontakt 1 = Eingang U = 60 bis 100 mV (vom MRB Z 1013)
  Kontakt 3 = Ausgang U >= 120 mV
- die Faehigkeit, hohe Frequenzen (f >=8 kHz nach TGL 27616/2) einwandfrei wiederzugeben,
- die einwandfreie Funktionsfaehigkeit des von Ihnen eingesetzten Geraetes, d. h. keine schwankende Wiedergabe der hoeheren Frequenzen.

Zu empfehlen sind die Kassettenrekorder GERACORD, ANETT, KR 650/660 u. ae., sowie alle Spulentonbandgeraete fuer Mono. Sollten Sie ein Stereogeraet verwenden, nutzen Sie nur eine Spur fuer die Aufnahme. Nicht einsetzen koennen Sie den Rekorder SKR 900.

Einige technische Besonderheiten sollte Ihr MBG noch besitzen:

- Aussteuerautomatik bzw. Handaussteuerung mit Aussteuerungsanzeige, um optimale und konstante Aufzeichnungspegel zu ermoeglichen. Geraete mit Handaussteuerung haben noch den Vorteil, dass Sie die Aussteuerung fuer eine sichere Aufzeichnung durch Probieren ermitteln koennen. So koennen Sie Magnetbandgeraete, die bei automatischer Aussteuerung nicht funktionieren, durch Uebersteuerung verwendungsfaehig machen.

- Bandlaengenzaehlwerk, damit Sie die Bandstelle mit Ihrem gewuenschten Programm schneller finden. Ist dies nicht vorhanden, helfen Sie sich durch Aufsprechen eines Programmamens, den Sie dann durch Abhoeren wiederfinden koennen.

Wie das MBG zur Informationsspeicherung genutzt wird, koennen Sie am Beispiel des Abschnittes 1.3 ueben. Aber anschliessen

wollen wir es jetzt schon. Dazu wird das MBG ueber ein handelsuebliches Diodenkabel (Achtung! kein Ueberspielkabel) mit der Buchse X5 (s. Abb. 1.1) verbunden. Anschliessend legen Sie noch eine Kassette ein bzw. legen ein Band auf. Nun ist Ihr Heimrechenzentrum fertig. In den naechsten Abschnitten soll gezeigt werden, wie Sie damit umgehen muessen.

## 1.3. Monitorkommandos des Z 1013

### 1.3.1. Allgemeine Form

Nach erfolgreicher Inbetriebnahme des Mikrorechners, ein ordnungsgemaesser Anschluss der Geraete vorausgesetzt, wird, durch das Aufforderungszeichen "#" signalisiert, eine Bedienereingabe erwartet. Dieses Zeichen gibt an, dass sich der Mikrorechner mit der Progammabarbeitung im sogenannten Betriebsprogramm (Monitor) befindet. In diesem Monitor sind alle Befehlsfolgen enthalten, die unbedingt benoetigt werden, um mit dem Mikrorechner arbeiten zu koennen. Diese Monitorleistungen sind mit Kommandos abrufbar.

Im folgenden soll, von der allgemeinen Form der Kommandos ausgehend, der Monitor vorgestellt werden.

Die allgemeine Form der Kommandos lautet:

```
XY aaaa bbbb cccc (ENTER)
```

Dabei bedeuten:

```
X    : ASCII-Zeichen (ASCII-Zeichen: Zeichen, die im Alpha-
       Modus von der Tastatur geliefert werden; s. 1.2.4.2)
Y    : Leerzeichen
aaaa bbbb cccc: eventuell vorhandene Parameter
ENTER:   Abschluss mit der Enter-Taste
```

Die Parameterangaben sind vom jeweiligen Kommando abhaengig. Es sind maximal drei vierstellige Hexadezinalzahlen (siehe dazu Abschn.1.3.2.) moeglich. Die Eingabe fuehrender Nullen ist nicht erforderlich. Fehlerhafte Werte der Kommandozeile koennen korrigiert werden, dazu ist mit den beiden Kursortasten "Kursor links '<-'" und "Kursor rechts '->'" auf der Tastatur der Kursor auf das fehlerhafte Zeichen zu positionieren und die Korrektur auszufuehren.

Vor Betaetigung der Enter-Taste ist der Kursor wieder hinter das letzt gueltige Zeichen zu positionieren. Sofern die Kommandozeile richtig eingegeben wurde, wird das Kommando ausgefuehrt. Bei fehlerhaften Kommandozeilen wird ein Fragezeichen und anschliessend wieder ein Aufforderungszeichen ausgegeben. Falls die Parameterangaben eines Kommandos denen des vorherigen entsprechen, kann eine Neueingabe entfallen und mit dem Zeichen ":" auf diese Parameter verwiesen werden.

## 1.3.2. Kommandos

In diesen Abschnitt sollen die Kommandos geordnet in alphabetischer Reihenfolge genannt werden und ihre Wirkung beschrieben werden. Lesen Sie zunaechst diesen Abschnitt, auch wenn nicht alles klar wird. Im nachfolgenden Abschnitt sollen diese geuebt werden.

Monitorkommandos:

- A (Alphaumschaltung)
  Schaltet die Tastatur wieder in den Grundzustand, sofern sie vorher mit dem Kommando "H" umgeschaltet war.

- B hadr (Breakpoint-Haltepunkt)
  Es wird eine Haltepunktadresse eingegeben. Diese Adresse muss im RAM-Bereich liegen und auf das erste Byte eines Befehles zeigen. Zur Kontrolle wird der eingetragene Haltepunkt BP.:...., die dort befindlichen Befehlsbytes BS.:.... sowie alle Registerinhalte angezeigt. Ein zu testendes Programm haelt beim Erreichen dieser Adresse an und gibt eine Reihe von Informationen aus. Das sind wieder die Haltepunktadresse sowie die ab dieser Adresse stehenden Befehlsbyte und alle Registerinhalte. Danach werden Monitorkommandos erwartet. Voraussetzung ist, dass die Haltepunktadresse auf das erste Byte eines Befehls zeigt.

- C adr1 adr2 anz (Compare)
  Dieses Kommando wird genutzt, um zwei Speicherbereiche miteinander zu vergleichen. Sind die Speicherbereiche gleich, meldet sich wieder der Monitor. Bei Ungleichheit erfolgt eine Fehlerausschrift in der Form: aaaa xx bbbb yy, wobei aaaa und bbbb Adressen und xx und yy deren Byte inhalte darstellen, zwischen denen die Ungleichheit besteht. Mit Betaetigen der Entertaste wird der Vergleich fortgesetzt, eine andere Taste bricht den Vergleich ab.

- D aadr eadr (Display Memory)
  Mit diesem Kommando koennen beliebige Speicherbereiche zwischen einer Anfangs- und einer Endadresse angezeigt werden. Die Anzeige des Bereiches zwischen FFF8 und FFFF ist mit dem D-Kommando nicht moeglich, dafuer muss das M-Kommando verwendet werden. Die Anzeige erfolgt zeilenweise in hexadezimaler Form. Zuerst wird die Adresse des jeweiligen Bereiches ausgegeben, danach folgen acht Byte des Speicherinhaltes, gefolgt von einer dreistelligen Pruefsumme. Es wird immer eine Zeile vollständig ausgegeben, auch wenn die Endadresse eine andere Anzahl von Bytes verlangt.

- E sadr (Execute)
  Es wird ein Maschinenprogramm ab der eingegebenen Startadresse unter Beachtung einer eventuell eingegebenen Haltepunktadresse gestartet. Zu Beginn werden alle Register der CPU mit definierten Inhalten aus dem Registerrette-Bereich geladen. Mit Erreichen eines Haltepunktes werden die CPU-Register im Registerrette-Bereich gespeichert und in den Monitor verzweigt. Eine Programmfortsetzung des zu testenden Programmes kann auf mehreren Wegen erfolgen:

* Festlegen eines neuen Haltepunktes mit dem B-Kommando und Fortsetzung mit dem G-Kommando (siehe dort)
* Schrittweise Abarbeitung mit dem N-Kommando (s. d.)
* Fortsetzung mit dem G-Kommando ohne Neufestlegung eines Haltepunktes

- F aadr anz aa bb cc .. (Find)
  Ab der angegebenen Adresse soll eine bestimmte Anzahl aufeinanderfolgender Bytes im Speicher gesucht werden. Werden diese Bytes gefunden, erfolgt ein Uebergang zum M-Kommando, die Bytes koennen gelesen und/oder veraendert werden. Wird die Bytefolge nicht gefunden, erfolgt die Aussohrift "NOT FOUND" auf dem Bildschirm.

- G (Go)
  Fortsetzung eines Programmes ab der Haltepunktadresse. Zuvor werden die geretteten CPU-Register wieder geladen. Das G-Kommando kann auch nach dem Schrittbetrieb gegeben werden. Wurde zuvor mit dem B-Kommando ein neuer Haltepunkt eingegeben, laeuft das zu testende Programm bis zu dieser neuen Haltepunktadresse.

- H (Hexadezimalumschaltung)
  Schaltet in der Tastaturkodetabelle die Zahlen 0 bis 9 sowie die entsprechenden Sonderzeichen in die Shiftebene 0, d. h. anstelle der Zeichen "H" bis "Q". Dadurch sind hexadezimale Eingaben ohne Benutzung der Shift-Taste moeglich.

- I (Initialisierung)
  Es erfolgt ein Loeschen des Registerrette-Bereiches, so dass nach Programmstart mit dem E-Kommando die CPU-Register mit definierten Anfangswerten geladen (geloescht) werden. Der weitere Ablauf ist wie nach Betaetigen der Reset-Taste, es wird der Grundzustand des Mikrorechners hergestellt.

- J sadr (Jump)
  Es wird ein Programm ab der Startadresse aktiviert, eine eventuell eingegebene Haltepunktadresse wird nicht beachtet, die Inhalte der CPU-Register sind undefiniert.

- K aadr eadr bb (Kill)
  Damit ist es moeglich, einen angegebenen Speicherbereich zu loeschen oder mit dem Byte bb zu fuellen. Wird das Kommando ohne Parameter verwendet, wird der gesamte adressierbare Speicher geloescht. Weiterarbeit ist dann nur nach Betaetigen der Resettaste moeglich.

- L aadr eadr (Load from Cassette)
  Ein mit dem S-Kommando ausgegebener Speicherbereich kann mit diesem Kommando wieder geladen werden. Dabei werden die ankommenden Byte ab der Anfangsadresse bis zur Endadresse im Speicher plaziert. Diese Adressen muessen nicht mit denen des S-Kommandos identisch sein, wichtig ist nur die Uebereinstimmung der Byteanzahl. Waehrend des Lesens wird mittels der aufgezeichneten Pruefsumme die Richtigkeit der ankommenden Daten kontrolliert. Stimmen errechnete und vom Band gelesene Pruefsumme nicht ueberein,

wird eine Fehlermeldung ausgegeben: CS<aerr. Der fehlerhafte Bereich unterhalb der Adresse aerr muss dann manuell kontrolliert werden. Moeglicherweise ist auch nur die Pruefsumme falsch gelesen worden. Reicht die Anzahl der eingelesenen Bytes nicht aus, den Speicher bis zur Endadresse zu fuellen, bleibt das Programm in der Eingabe haengen, der Monitor kann nur wieder mit der Reset-Taste erreicht werden.

- M aadr (Modify)
  Es ist moeglich, mit diesem Kommando einen Speicherbereich ab der angegebenen Anfangsadresse byteweise anzuzeigen und gegebenenfalls zu veraendern. Es erfolgt die Ausgabe der aktuellen Adresse und des Inhaltes des zugehoerigen Bytes. Anschliessend wird mit dem Zeichen "#" zur Eingabe aufgefordert. Soll der alte Inhalt beibehalten werden, ist nur die Enter-Taste zu betaetigen, ansonsten wird vorher eine hexadezimale Zahl eingegeben. Es koennen auch mehrere Byteinhalte, durch Leerzeichen voneinander getrennt, eingegeben werden.

  Nach Betaetigung der Enter-Taste wird die aktuelle Adresse erhoeht und auf der naechsten Zeile fortgesetzt. Wird versucht, einen nicht vorhandenen Speicherbereich oder einen ROM zu beschreiben, erfolgt eine Fehleraussohrift: ER aerr bb, wobei aerr die Adresse und bb den fehlerhaften Inhalt darstellen. Anschliessend wird eine erneute Eingabe erwartet. Diese Fehlerausschrift wird vor allem dann auftreten, wenn versucht wird, nicht vorhandene Speicher oder Festwertspeicher zu beschreiben. Mit Eingabe des Zeichens "R" kann die aktuelle Adresse bei Bedarf zurueckgestellt werden.

  Die Komandoausfuehrung wird beendet durch Eingabe eines Semikolon ";". Die aktuelle Adresse wird als Endadresse uebernommen. Mit dem Kommando 'D :' kann der aktualisierte Speicherbereich nochmals auf dem Bildschirm angezeigt werden.

- N (Next)
  Dieses Kommando veranlasst die Ausfuehrung genau eines Befehls des zu testenden Programmes (Schrittbetrieb). Das N-Kommando kann nur angewandt werden, wenn zuvor ein Haltepunkt gesetzt und das zu testende Programm mit dem E-Kommando gestartet wurde. Nach der Ausfuehrung des Befehls werden alle Registerinhalte gerettet. Angezeigt werden der Befehlszaehler, die abzuarbeitenden Befehlsbyte sowie alle Registerinhalte. Waehrend des Schrittbetriebes duerfen in dem zu testenden Programm keine der nachfolgenden Befehle auftreten:

    'IM0', 'IM1' - Veraenderung im Interruptmodus
    'LD I, A'    - Veraenderung des Interruptvektors in der CPU
    'DI'         - Verbieten Interrupt

- R rg/rg' (Register Display/Modify)
  Mit diesem Kommando ist es moeglich, Inhalte beliebiger Doppelregister der CPU einschliesslich des Austauschregistersatzes anzuzeigen und zu veraendern, Nach Eingabe der Registerbezeichnung (AB, DC, DE, HL, IX, IY, PC, SP,

AF', BC', DE', HL') wird der Inhalt des ausgewaehlten Doppelregisters ausgegeben und mit den Zeichen "#" die Eingabe des neuen Wertes erwartet. Wird an Stelle einer Registerbezeichnung ein Doppelpunkt ":" eingegeben, werden alle Registerinhalte angezeigt.

```
BP:XXXX BS:XXXXXX   S Z C X X X
SP:XXXX PC:XXXX IX:XXXX IY:XXXX
AF:XXXX BC:XXXX DE:XXXX HL:XXXX
AF:XXXX BC:XXXX DE:XXXX HL:XXXX'
```

Zu beachten ist, dass nur das S-, Z- und C-Flag einzeln angezeigt wird, die Belegung der anderen Flags ist dem AF-Register zu entnehmen.

- S aadr eadr (Save to Cassette)
  Der Speicherbereich von Adresse aadr bis zur Adresse eadr wird ueber das Magnetbandinterface auf Magnetband ausgegeben. Nach einem etwa 1,5 Sekunden langen Kennton werden die Daten in Bloecken zu 32 Byte mit einer anschliessenden Prüfsumme pro Block ausgegeben.

- T aadr zadr anz (Transfer)
  Es erfolgt ein Transport eines Speicherbereiches ab der Anfangsadresse auf eine Zieladresse mit der festgelegten Anzahl von Bytes. Dabei ist eine Ueberlappung der beiden Bereiche moeglich

- W aaaa eeee (Window)
  Dieses Kommando realisiert eine Fensterfunktion, innerhalb dessen die Rollfunktion des Bildschirms erhalten bleibt. Ausserhalb dieses Fensters wird die Bildschirmausgabe als Standbild realisiert. Der Anfang des Fensters wird mit dem Parameter aaaa, dessen Ende mit eeee festgelegt. Der kleinste realisierbare Fensterausschnitt besteht aus zwei Zeilen. Sollen nur die letzten beiden Zeilen rollen, sind als Parameter die Angaben aaaa=EFC0 und eeee=EFFF+1=F000 notwendig. Der volle Bereich wird durch die Parameter aaaa=EC00 und eeee=EFFF=F000 eingestellt. Der Bildschirm wird nicht geloescht, der Kursor wird an den Anfang des Fensters positioniert.

## 1.3.3. Verwendung der Monitorkommandos

Anhand eines Beispielprogrammes wollen wir jetzt alle Monitorkommandos trainieren. Das Beispiel wurde so gewaehlt, dass Sie das ordnungsgemaesse Arbeiten des Programmes auf dem Bildschirm verfolgen koennen. Mit Hilfe des Programmes werden die schwarzen Schachfiguren in der Bildschirmmitte abgebildet.

Programm:

```
Adresse  Maschinenkode Mnemonik    Kommentar

1000     06 06         LD B, 6     ; Zahl der Schachfiguren
1002     DD 21 08 EE   LD IX,EE08H ; Position der ersten Bi-
                                   ; gur auf Bildschirm (BS)
1006     11 1C 10      LD DE,101CH ; Adresse der Tabelle fuer
                                   ; Zeichencode der Schach-
                                   ; figuren
1009     1A        Ml: LD A,(DE)   ; oberen Teil der Schach-
100A     DD 77 00      LD (IX+0),A ; figur auf BS
100D     13            INC DE      ; naechsten Zeichencode
100E     1A            LD A, (DE)  ; unteren Teil der Schach-
100F     DD 77 20      LD (IX+20),A; figur auf BS
1012     13            INC DE      ; naechsten Zeicheneode
1013     DD 23         INC IX      ; Abstand zum naechsten
1015     DD 23         INC IX      ; Zeichen einstellen
1017     DD 23         INC IX      ;
1019     10 EE         DJNZ Ml     ; naechste Figur, bis alle
                                   ; 6 abgebildet
101B     FF            DB FFH      ; Ruecksprung in Monitor
101C     0E 11 0F 11               ; Tabelle fuer Zeichenkode
1020     10 11 12 13               ; der Schachfiguren
1024     14 16 15 16
```

Falls Sie nicht alles verstanden haben, machen Sie trotzdem weiter mit. Die Wirkungsweise der Kommandos wird auch so deutlich. Sie koennen dann beim Erlernen der Maschinensprache immer das Gelernte am Rechner ausprobieren.

Da Sie beim Eingeben des Pregrammes vorwiegend Hexadezimalzahlen (siehe Abschn.2.4.1. ) benutzen, schalten Sie als erstes die Tastatur durch Betaetigen der Tasten 'H' und 'ENT' um. Jetzt muessen Sie beim Eingeben eines alphanumerischen Zeichens, wie z. B. die Monitorkommandos, immer gleichzeitig 'S1' druecken.

Zunaechst geben Sie das Programm ab Adresse 1000H ein.

Eingabe:

'S1' und 'M'  ' '  '1'  '0'  '0'  '0'  'ENT'

BS:

```
# M 1000
1000 # _
```

Jetzt tippen Sie die Zahlen aus der Spalte Maschinenkode paarweise mit jeweils einem Leerzeichen ein und schliessen Sie diese mit ';' ab.

**Eingabe:**

```
'0' '6' 'ENT' '0' '6' ' ' 'D' 'D' ' ' '2' '1' 'ENT' ...
'ENT' '1' '5' ' ' '1' '6' ';' 'ENT'
```

**Sie bemerken, Sie koennen anstelle des Leerzeichens auch 'ENT' druecken, dann erfolgt die weitere Eingabe immer auf der naechsten Zeile.**

**Bildschirm:**

```
1000 xx # 06
1001 xx # 06 DD 21
     .
     .
     .
1026 xx # 15 16;
# _
```

**Fuehren Sie jetzt eine Kontrolle Ihrer Eingabe durch folgende Aktivitaet durch:**

**Eingabe:**

```
'D' ' ' '1' '0' '0' '0' ' ' '1' '0' '2' '0' 'ENT'
```

**BS:**

```
# D 1000 1020
1000 06 06 DD 21 08 EE 11 1C 22D
1008 10 1A DD 77 00 13 1A DD 288
1010 77 20 13 DD 23 DD 23 DD 387
1018 23 10 EE FF 0E 11 0F 11 25F
1020 10 11 12 13 14 16 15 16 09B
```

**Stimmt Ihr Ergebnis mit dem hier angegebenen ueberein?**

**Sie brauchen dafuer nur die letzte Spalte mit den dreistelligen Ziffern, der sogenannten Pruefsumme, vergleichen. Wenn Sie das Programm an einer anderen Stelle im Speicher haben wollen, koennen Sie es auch in einen anderen Bereich transportieren, z. B. auf die Adresse 2000. Das Programm umfasst 28H, d. h. dezimal 40, Speicherplaetze.**

**Eingabe:**

```
'S1' und 'T' ' ' '1' '0' '0' '0' ' ' '2' '0' '0' '0' ' '
'2' '8' 'ENT'
```

**BS:**

```
# T 1000 2000 28
# _
```

**Kontrollieren Sie die Ausfuehrung des Transports, indem Sie den Speicherbereich ab Adresse 2000H anzeigen.**

**Eingabe:**

**'D' ' ' '2' '0' '0' '0' ' ' '2' '0' '2' '0' 'ENT'**

**BS: wie oben, aber vorn die aktuellen Adressen**

**Sie koennen den Vergleich auch mit dem C-Kommando den Rechner selbst ausfuehren lassen.**

**Eingabe:**

**'C' ' ' '1' '0' '0' '0' ' ' '2' '0' '0' '0' ' ' '2' '8' 'ENT'**

**BS:**

```
# C 1000 2000 28
# _
```

**Die 40 Speicherplaetze ab Adresse 1OOOH und die ab Adresse 2000H stimmen ueberein, da sonst die Stelle, ab der Ungleichheit herrscht, angezeigt werden wuerde.**

**Wollen Sie jetzt einen bestimmten Befehl aendern, dann lassen Sie diesen vom Rechner suchen, und zwar mit dem F-Kommando. Aendern Sie Jetzt den Befehl LD DE, 101CH (Maschinenkode: 11 1C 10) in dem Bereich ab Adresse 2000H.**

**Eingabe:**

**'F' ' ' '2' '0' '0' '0' ' ' '3' ' ' '1' '1' ' ' '1' 'C' ' ' '1' '0' 'ENT' 'ENT'**

**BS:**

```
2006 11 #
2007 1C # _
```

**Der naechste Speicherplatz soll in 20 geaendert werden.**

**Eingabe:**

**'ENT' '2' '0' ';' 'ENT'**

**Nun haben Sie ab Adresse 1000H und ab Adresse 2000H jeweils ein lauffaehiges Programm. Um dessen Wirkung besser verfolgen zu koennen, loeschen wir den Bildschirm mit dem K-Kommando.**

**Eingabe:**

**'S1' und 'K' ' ' 'E' 'C' '0' '0' ' ' 'E' 'F' 'F' 'F' ' ' '2' '0' 'ENT'**

**Die Adressen EC00H und EFFFH schliessen den Bildschirmbereich ein, 20H ist der hexadezimale Code fuer das Leerzeichen. Sie koennen fuer den Leerzeichenkode auch den fuer das A, also 41H, eingeben. Sie fuellen also den Bereich mit Leerzeichen.**

**BS:**

**# _ (in der letzten Zeile)**

**Jetzt starten Sie das Programm ab Adresse 1000H.**

**Eingabe:**

**'S1' und 'J' ' ' '1' '0' '0' '0' 'ENT'**

**BS:**

**Schachfiguren in der Bildmitte**

**Wir wollen jetzt die Arbeitsweise des Programmes schrittweise verfolgen, was bei selbsterstellten, eventuell noch fehlerhaften Programmen vorteilhaft ist. Dazu stellen wir mit dem I-Kommando den Grundzustand ein.**

**Eingabe:**

**'H' 'ENT'**
**'S1' und 'I' 'ENT'**

**BS: Grundzustand**

**Wir schalten wieder in den Hexa-Modus um. Da bei schrittweiser Abarbeitung die Aenderung der Registerinhalte Aufschluss ueber die richtige Ausfuehrung der Befehle gibt, sind mit dem R-Kommande die Register darstellbar und aenderbar.**

**Eingabe:**

**'S1' und 'R' ' ' ':'**

**BS:**

```
BP:  XXXX BS:  XXXXXX    S  Z  C  0  0  0
SP:  0090 PC:  0000  IX:  0000  IY:  0000
AF:  0000 BC:  0000  DE:  0000  HL:  0000
AF:  0000 BC:  0000  DE:  0000  HL:  0000'
```

**Das Programm soll nach Abbildung, der ersten Schachfigur unterbrochen werden, also bei Adresse 1012H.**

**Eingabe:**

**'B' ' ' '1' '0' '1' '2' 'ENT'**

**Anzeige:**

**BP: 1012 BS: 13 DD 23**
**alle Register wie oben**

**Nun muessen Sie das Programm mit dem E-Kommando starten.**

**Eingabe:**

**'E' ' ' '1' '0' '0' '0' 'ENT'**

**BS: 1 Bauer in BS-Mitte im Zeichengewirr**

**Damit der Programmablauf weiter verfolgt werden kann, richten wir uns auf dem Bildschirm ein Anzeigefenster ein, in dem alle von Ihnen gemachten Eingaben angezeigt werden. Die Anzeige soll nur noch im unteren Viertel des Bildschirmes erfolgen. Das entspricht dem Speicherbereich EF00H bis EFFFH.**

**Eingabe:**

**'S1' und 'I' 'ENT'**
**'H' 'ENT'**
**'S1' und 'W' ' ' 'E' 'F' '0' '0' ' ' 'E' 'F' 'F' 'F' 'ENT'**

**BS:**

**# H**
**# W EF00 EFFF**

**# _ (am Beginn des unteren Viertels)**

**So, nun geben Sie wieder den Abbruchpunkt ein und starten wieder mit dem E-Kommando.**

**Eingabe:**

**'B' ' ' '1' '0' '1' '2' 'ENT' (-->Registeranzeige)**
**'E' ' ' '1' '0' '0' '0' 'ENT'**

**BS:**

**- Bauer in Bildmitte**
**- unteres Viertel: # E 1000**
**Registeranzeige**

**Jetzt koennen Sie das Programm Befehl fuer Befehl mit dem N-Kommando abarbeiten, indem Sie wiederholt 'S1' und 'N' sowie 'ENT' druecken.**

**Damit Sie nicht immer drei Tasten druecken muessen, schalten Sie die Tastatur wieder in den Alpha-Modus zurueck.**

**Eingabe:**

**'A' 'ENT'**

**BS:**

**# A**
**# _**

Fahren Sie nun mit 'N' und 'ENT' fort. Sie koennen in der Anzeige die Aenderung des Abbruchpunkte (BP) sowie die der im Programm verwendeten Register DE, IX und BC verfolgen.

In der Bildschirmmitte werden nach und nach die Schachfiguren aufgebaut. Ist das Programm vollstaendig durchlaufen, erscheint auf dem Bildschirm:

BS:

```
? # _
```

Sie haben nun ausfuehrlich mit den Monitorkommandos gearbeitet und wollen die Arbeit zunaechst beenden. Damit Ihr Programm nicht verloren geht, speichern Sie dieses mit Hilfe des S-Kommandos auf einem Magnetband:

Eingabe:

```
'H' 'ENT'
'S1' und 'S' ' ' '1' '0' '0' '0' ' ' '1' '0' '3' '0'
```

Aufnahmebereitschaft des Magnetbandgeraetes herstellen; Aufnahme starten

```
'ENT'
```

BS:

```
? # H
# S 1000 1030
```

Sie hoeren jetzt zunaechst einen laengeren Signalton und anschliessend ein knarrendes Geraeusch. Das Programm befindet sich auf Magnetband, wenn auf dem Bildschirm wieder das Quittungssymbol '#' erscheint. Sie koennen jetzt alles ausschalten.

Ueben Sie noch das Einlesen eines Programmes vom Magnetband. Schalten Sie Ihren MRB Z1013 noch einmal an und gehen Sie folgendes ein:

Eingabe:

```
'H' 'ENT'
'S1' und 'L' ' ' '1' '0' '0' '0' ' ' '1' '0' '3' '0'
```

Magnetband an den Programmanfang, entsprechend des von Ihnen notierten Bandzaehlerstandes, zurueckspulen; Wiedergabe starten; bei Ertoenen des Signaltones druecken von 'ENT'

BS:

```
# H
# L 1000 1030
```

Wird das Programm fehlerfrei gelesen, wird auf dem Bildschirm das Quittungesymbol '#' ausgegeben. Beim Auftreten von Fehlern beim Einlesen ermittelt der Rechner die fehlerhaften Pruefsummen und gibt die Adresse des letzten Speicherplatzes des fehlerhaften Blockes aus, z. B.

```
# L 1000 1030
CS<1020
```

Sie haben nun das Programm ohne Fehler eingelesen. Starten Sie es mit dem J-Kommando. Sie merken, das Programm befindet sich wieder im Speicher.

Sie haben also gelernt, den MRB Z1013 in Betrieb zu nehmen und ihn zu bedienen. Wie der Z1013 arbeitet, was er noch bietet und was man alles mit ihm machen kann, erfahren Sie in den nachfolgenden Teilen des Handbuches zum MRB Z1013.

## 1.4. Wichtige Hinweise zur Beibehaltung der Reparaturfähigkeit

Beim vorliegenden Geraet handelt es sich um eine komplexe mikroelektronische Baugruppe. Eine Pruefung und Reparatur ist nur computergestuetzt moeglich. Das erfordert bestimmte Eigenschaften der Leiterplatte:

Jegliche Aenderung von konstruktiven und elektrischen Werten (Aenderung der Leiterplattenkontur, Anbringen von anderen bzw. zusaetzlichen Steckverbindern, Austausch von Bauelementen, Nachruesten von Speicherschaltkreisen, Schaltungsaenderungen usw.) bringen den Z 1013 in einen nichtreparaturfaehigen Zustand.

Deshalb ist der Z 1013 im Reparaturfall im Originalzustand abzugeben. Eine nach der in der Bedienungsanleitung (s. 1.2.4.1.) angebrachte Tastatur ist dabei zulaessig. Wollen Sie auf einen steckbaren Anschluss Ihrer Tastatur nicht verzichten, ist Ihnen als einzige Ausnahme das Anloeten einer Buchsenleiste BuL 202-26 TGL 29331/04 gestattet.

Eingriffe (ausser den genannten Tastatur-Massnahmen) durch den Kunden fuehren selbstverstaendlich zum Erloeschen der Garantieansprueche. Zur Wiederherstellung der Reparaturfaehigkeit werden kleine Aenderungen zu Lasten und auf Risiko des Besitzers rueckgaengig gemacht. Laesst sich die Reparaturfaehigkeit nicht wieder herstellen, so kann keine Instandsetzung im VEB Robotron-Elektronik Riesa durchgefuehrt werden.

## 1.5. Technische Daten

| | |
|---|---|
| **Mikroprozessor:** | **U 880** |
| **Festwertspeicher:** | **2 KByte ROM Betriebssystem**<br>**2 KByte ROM Zeichengenerator (96 Ziffern, Buchstaben und Sonderzeichen, 146 Grafikzeichen)** |
| **Arbeitsspeicher:** | **16 KByte dyn. RAM (Z1013.01) bzw.**<br>**1 KByte stat. RAM (Z1013.12) Anwenderspeicher**<br>**1 KByte Bildwiederholspeicher** |
| **Speichererweiterung:** | **bis max, 64 KByte ueber Systembus moeglich** |
| **Ein- und Ausgabe:** | **8 Kanaele eines E/A-Tores des E/A-Bausteines U 855** |
| **Tastatur:** | **Folienflachtastatur mit 4x8 Tasten (Ziffern, Buchstaben, Sonderzeichen, Steuertasten)** |
| **Bildschirmsteuerung:** | **Anschluss eines Fernsehgeraetes ueber Antennenbuchse (Bereich I / Kanal 3); Nutzung des BAS-Signals moeglion; keine Farbe; Bildaufbau 32 Zeilen x 32 Zeichen** |
| **Magnetbandinterface:** | **Kassetten- oder Spulentonbandgeraet;**<br>**Uebertragung diphasenkodierter Signale;**<br>**Eingang U = 60 bis 100 mV**<br>**Ausgang U >= 120 mV;** |
| **Stromversorgung:** | **externe Zufuehrung von 12V/1A Wechselspannung; interne Erzeugung und Stabilisierung von +5V, -5V und +12V** |
| **Erweiterungsmoeglichkeiten:** | **ueber Systembus (K 1520-kompatibel) sowie PIO-Anwendertor** |
| **Programmierung:** | **U 880-Maschinencode** |
| **Abmessungen** | **215 x 230 mm** |

R O B O T R O N

Mikrorechnerbausatz Z 1 0 1 3

Handbuch Teil I

Bestandteile des Handbuches:
---

Handbuch Teil I
Handbuch Teil II
Anlagenteil

## 2. Grundbegriffe der Mikrorechentechnik

### 2.1. Hardware oder Software?

Dieses Kapitel ist vor allem fuer den Leser gedacht, der in der Mikrorechentechnik nicht bewandert ist. Es werden hier einige Grundbegiffe erlaeutert, die das Verstaendnis der nachfolgenden Kapitel erleichtern sollen.
Der erste Begiff, der zu klaeren waere, ist der des Mikrorechners. Ein Mikrorechner ist ein komplexes System verschiedener Funktionseinheiten auf der Basis mikroelektronischer Schaltkreise, die auf bestimmte Art miteinander in Verbindung treten und durch ihr Gesamtverhalten eine vorgegebene Aufgabe (Programm) loesen. Ein Programm stellt dabei eine Folge von Anweisungen (Befehlen) dar.
Gekennzeichnet wird ein Mikrorechner im wesentlichen durch seine Hard- und Software. Unter Hardware wird dabei sowohl die Gesamtheit der mechanischen und elektronischen Bauelemente, wie integrierte Schaltkreise, Transistoren, Widerstaende usw., als auch die Art und Weise der Verschaltung dieser Bauelemente verstanden.
Als Software eines Rechners werden seine Programme, z. B. Betriebsprogramm und BASIC-Interpreter, bezeichnet. Das Betriebsprogramm (oder auch Betriebssystem) enthaelt die Programme, die die Zusammenarbeit der einzelnen Systemkomponenten organisieren bzw. ueberhaupt ermeeglichen. Worin unterscheidet sich aber nun ein Mikrorechner von einer herkoemmlichen Schaltung?
Um eine bestimmte Steuerungsaufgabe loesen zu koennen oder immer wiederkehrende Berechnungen zu realisieren, muss nicht immer ein Mikroprozessor verwendet werden. Vielfach ist es einfacher, eine Schaltung mit einfachen Logikschaltkreisen aufzubauen. Eine solche Schaltung haette ueberdies den Vorteil, schneller als ein Mikroprozessor zu arbeiten. Aber bereits einfache Aenderungen der Aufgabenstellung wuerden einen neuen Schaltungsentwurf erfordern, der mit einem bestimmten Arbeitsaufwand realisiert werden musste. Komplexere Aufgabenstellungen liessen sich auf diese Art ueberhaupt nicht realisieren, da der Aufwand zu hoch werden koennte. Die Loesung einer Aufgabe mit Hilfe eines Mikrorechners ist weitaus einfacher.

Der Mikroprozessor ist in der Lage, alle Verknuepfungsmoeglichkeiten der Logikschaltkreise nachzubilden und damit jedes gewuenschte Verhalten zu realisieren. Die uebrigen Funktionseinheiten des Mikrorechnere enthalten dann in den Speichereinheiten den Loesungsablauf der Aufgabe in Form von Anweisungen fuer den Mikroprozessor, die Ausgangsdaten sowie konstante Werte. Ueber andere Funktionseinheiten werden Signale aufgenommen sowie Steuersignale wieder abgegeben. Die erreichbare Arbeitsgeschwindigkeit ist kleiner als bei reinen Logikschaltungen. Da aber nicht die maximal erreichbare Arbeitsgeschwindigkeit, sondern die fuer den jeweiligen Prozess oder die Steuerung benoetigte Geschwindigkeit entscheidend

ist, ist dieser Nachteil nur in wenigen Faellen von Bedeutung. Eine Aenderung der Aufgabenstellung fuehrt meist nur zu einer Aenderung der Anweisungen fuer den Mikroprozessor. Diese Aenderung ist schnell realisierbar. Mit dem Mikroprozessor lassen sich ohne technische Veraenderungen vielerlei Aufgabenstellungen besen, es ist meist nur erforderlich, andere Anweisungen zu erarbeiten.

## 2.2. Bestandteile eines Mikrorechners

### 2.2.1. Zentrale Verarbeitunseinheit

Die Zentrale Verarbeitungseinheit, im Englischen als "Central process unit" (CPU) bezeichnet, ist der wichtigste Bestandteil eines Mikrorechners. Eine solche CPU liesse sich aus diskreten Elementen, d. h. Transistoren, Widerstaenden und Kondensatoren aufbauen, wuerde aber einen sehr grossen Aufwand erfordern. Mit der Entwicklung der Mikroelektronik konnte diese Funktionseinheit in Form einer integrierten Schaltung als sogenannter Mikroprozessor bereitgestellt werden und damit zu einer wesentlichen Vereinfachung im Schaltungsentwurf beitragen. Am Beispiel des Mikroprozessors U880, der im MRB Z1013 Verwendung findet, sollen einige wichtige Bestandteile erlaeutert werden. Dazu gehoeren:

- CPU-Steuerung/Befehlsdekodierung
  Hier werden anhand eines vorgegebenen Befehls bestimmte Signale erzeugt. Bestimmte Zustaende, die von der CPU-Steuerung erkannt werden, sowie der zugefuehrte Takt erzeugen zeitlich festgelegte Signalfolgen, die sowohl den Ablauf innerhalb der CPU steuern, die aber auch als Steuersignale in allen angeschlossenen Funktionseinheiten ausgewertet werden koennen und die gesamten Ablaeufe eines Mikrorechners koordinieren (siehe Zeitdiagramme Anlage 10).
- Arithmetisch-logische Einheit (ALU)
  In der ALU koennen Daten entsprechend eines Befehle verknuepft werden. Zu diesen Operationen mit Daten gehoeren: Addition, Subtraktion, UND-Verknuepfung (Konjunktion), ODER-Verknuepfung (Disjunktion) sowie eine Reihe weiterer Operationen wie Verschiebungen und Bitmanipulationen. Eine Veraenderung der Daten ist nur in der ALU moeglich, erforderlichenfalls muessen diese erst in die ALU geholt und danach zurueckgetransportiert werden.
- Registersatz (Zwischenspeicher)
  In der CPU existieren Zwischenspeicher, die als Register bezeichnet werden. Hier koennen Zwischenergebnisse aufbewahrt und in der ALU miteinander verknuepft werden. Einige Register besitzen spezielle Bedeutung, wie z. B. der sogenannte Kellerzeiger (Stackpointer SP), Befehlszaehler (PC), Refreshregister und Interruptregister (s. auch Abschn. 4.).

Bestimmte Register sind doppelt vorhanden und koennen durch einen Befehl umgeschaltet werden. Ein Register wird benutzt, um den Zustand der CPU waehrend der Befehlsabarbeitung zu speichern. Es wird als Flag-Register bezeichnet (die Bezeichnung "Flag" sollte als Anzeiger verstanden werden). In einem Register wird der gelesene Befehl zwischengespeichert, bis die durch ihn veranlasste Operation beendet ist. Dieses Register heisst demzufolge Befehlsregister.

Die Arbeit der CPU wird durch eine Reihe von Systemsignalen gekennzeichnet, die als Anschluesse herausgefuehrt wurden und das Zusammenwirken mit den angeschlossenen Funktionseinheiten steuern.

## 2.2.2. Speicher

In der Mikrorechentechnik haben sich zur Speicherung von Informationen Halbleiterspeicher weitgehend durchgesetzt. Es sind integrierte Schaltungen in unterschiedlichen Gehaeusegroessen, je nach Kapazitaet des Speichers. Speicher werden zu verschiedenen Zwecken benoetigt, z. B. um der CPU die abzuarbeitenden Befehle zur Verfuegung zu stellen. Da die Register der CPU meist nicht ausreichen, alle Zwischenergebnisse aufzubewahren, muessen diese ebenfalls in den Speicher gebracht werden.

Als Modell eines Speichers mag ein langer Schrank mit vielen Faechern dienen. Diese Faecher sind einzeln numeriert. Diese Numerierung soll bei Null beginnen und lueckenlos bis zu einem Endwert erfolgen. Jedes Fach entspricht einem Speicherplatz und kann eine Information enthalten. Die maximale Anzahl der Faecher bestimmt die Kapazitaet dieses Speichers.

Die Zeit, die vom Anlegen einer Speicherplatzadresse bis zur Bereitstellung der gespeicherten Daten beneetigt wird, wird Zugriffszeit genannt.

Zwischen der Speicherung von Daten und Programmen bestehen einige Unterschiede. Programme werden meist in Speichern aufbewahrt, die auch nach Abschalten der Versorgungsspannung ihren Inhalt behalten. Allerdings koennen diese Speicher nur gelesen werden, zum Beschreiben dieser Speicher sind spezielle Einrichtungen notwendig.

### 2.2.2.1. Programmspeicher (Nur-Lese-Speicher)

In einem Programmspeicher sind die Anweisungen fuer einen Mikroprozessor enthalten. Diese Anweisungen gehen auch nach Ausschalten des Rechners nicht verloren, sie sind nicht fluechtig. Diese Speicher bezeichnet man als Nur-Lese-Speicher (Read only memory - ROM), die Informationen werden einmal eingegeben und stehen staendig zur Verfuegung.

Je nach Eingabe der Information unterscheidet man: ROM's, die bereits waehrend der Herstellung ihre Informationen erhalten und ROM's, die nachtraeglich elektrisch programmiert werden koennen (ein einmaliger Vorgang, da die Struktur des Speichers veraendert wird). Diese letztgenannten Speicher heissen PROM (Prograinmable ROM). Eine weitere Speicherart kann sowohl programmiert als auch wieder geleescht werden. Das Loeschen erfolgt mit ultraviolettem Licht (UV-Licht) und loescht immer den gesamten Speicher. Diese Speicherart nennt man EPROM (Erasable PROM). Das Einschreiben der Programme in den EPROM geschieht mit speziellen Funktionseinheiten, sogenannten EPROM-Programmiergeraeten.

In den EPROM wird die zu speichernde Information mittels einer Programmierspannung als Ladungsmenge eingegeben. Nach Erreichen einer vorgegebenen Ladung ist der Baustein programmiert. Die Bestrahlung mit UV-Licht hat zur Folge, dass die gespeicherte Ladungemenge wieder abgebaut wird. Nach dem Loeschen ist der EPROM wieder programmierbar.

### 2.2.2.2. Datenspeicher (Schreib-Lese-Speicher)

Zur Aufbewahrung von Zwischenergebnissen oder anderen veraenderbaren Informationen werden Schreib-Lese-Speicher verwendet. Da diese Speicher wahlweise gelesen oder beschrieben werden koennen, nennt man sie Speicher mit wahlfreiem Zugriff (Random access memory - RAM). Mit Abschalten der Stromversorgung verlieren RAM's ihren Inhalt, sie sind also nicht zur Aufbewahrung von Informationen verwendbar, die immer verfuegbar sein muessen. Es werden zwei grundsaetzliche Typen unterschieden: statische und dynamische RAM's.
In den statischen RAM's werden Transistorkombinationen zur Aufbewahrung der Informationen verwendet. Eine solche Transistorkombination kann zwei verschiedene Zustaende annehmen und behaelt eine somit eingetragene Information bis zum Abschalten oder Ueberschreiben mit einer neuen Information.
Dynamische RAM's speichern die, Information als Ladung eines kleinen Kondensators ab. Diese Ladung muss, auf Grund der Selbstentladung, periodisch erneuert werden, dieser Vorgang wird mit REFRESH (Auffrischen) bezeichnet. Das Auffrischen wird bereits erreicht, wenn der Speicher gelesen wird. Die CPU U880 unterstuetzt diesen Vorgang durch Aussenden einer REFRESH-Information, um die zeitlichen Bedingungen zum Auffrischen unter allen Umstaenden zu gewaehrleisten. Werden die Zellen der dynamischen RAM's nicht spaetestens nach 2 Millisekunden aufgefrischt, geht ihre gespeicherte Information verloren.
Trotz des nicht unerheblichen Mehraufwandes werden dynamische RAM's verwendet, da sie bei gleichen Abmessunngen der Bausteine eine groessere Speicherkapazitaet und kleinere Leistungsaufnahme gegenueber statischen RAM's aufweisen.

### 2.2.3. Ein/Ausgabe-Einheiten

Unter externen Geraeten sollen im folgenden alle Geraete verstanden werden, mit denen Informationen in den Mikrorechner eingegeben oder vom Mikrorechner ausgegeben werden. Damit ist es moeglich, sowohl Daten als auch Programme in den Mikrorechner zu bringen und die Ergebnisse fuer den Nutzer sichtbar zu machen. Solche Geraete sind Lochbandleser und -stanzer, Magnetbandtechnik, Tastaturen, Bildschirm usw.
Die Verbindung dieser Geraete mit dem Mikroprozessor erfolgt ueber sogenannte E/A-Funktionseinheit en, in denen spezielle integrierte Schaltungen enthalten sind. Diese Funktionseinheiten steuern selbstaendig die Arbeit der Geraete und treten mit der CPU nur zur Informationsuebermittlung in Kontakt. Damit wird die CPU entlastet und die Programmabarbeitung wesentlich effektiver.
Weiterhin koennen auch Funktionseinheiten angeschlossen werden, die beliebig zur Verfuegung gestellte Meldesignale aus zu ueberwachenden Prozessen aufnehmen und sie fuer den Mikroprozessor aufbereiten. Gleichermassen ist die Abgabe von Steuersignalen zur Beeinflussung bestimmter zu steuernder Prozesse moeglich.

Als integrierte Schaltkreise werden dazu im
MRB Z1013 parallele E/A-Schaltkreise (Parallel Input Output-
PIO) vom Typ U855 verwendet.

### 2.2.4. Verbindung der Funktionseinheiten

Der Mikroprozessor (CPU) sendet Signale ab und wertet bestimmte empfangene Signale aus. Diese Signale werden i. a. in allen angeschlossenen Funktionseinheiten benoetigt.
Die Leitungen zur Uebermittlung von Daten, Adressen und Systemsignalen, wie z. B. LESEN, SCHREIBEN, werden entsprechend ihrer Funktion zu Leitungsbuendeln zusammengefasst.
Da diese Leitungen die Daten und Informationen zwischen den einzelnen Funktionseinheiten transportieren, wurde der Begriff "Bus" fuer ein solches Leitungsbuendel gepraegt.
Demzufolge bezeichnet man die Datenleitungen als DATENBUS, die Adressleitungen werden als ADRESSBUS und die Systemleitungen als STEUERBUS bezeichnet. Gelegentlich steht der Begiff "SYSTEMBUS" auch fuer alle Leitungen innerhalb des Mikrorechnersystems.
Durch Verwendung eines einheitlichen Systembusses ist es moeglich, beliebige Funktionseinheiten einem bestehenden System hinzuzufuegen, d. h. das System staendig zu erweitern. Voraussetzung ist die Uebereinstimmung der elektrischen Anschluesse der jeweiligen Einheiten.

## 2.3. Programmabarbeitung

### 2.3.1. Ablauf in der CPU

Wie bereits gesagt, beneetigt die CPU zur Lossung ihrer Aufgaben Anweisungen, die den gesamten Ablauf des Mikrorechners steuern. Diese Anweisungen oder Befehle findet die CPU in den angescohlossenen Speichereinheiten in einer ganz bestimmten, fuer sie verstaendlichen Form, die als Maschinenkode bzw. MC bezeichnet wird. Alle Anweisungen an die CPU muessen also in Form dieses MC vorliegen bzw. sind in diese Form zu bringen. Im Anhang befindet sich eine Uebersicht, in der die Befehle des U880 sowie deren Darstellung im Maschinenkode enthalten sind.
Um eine bestimmte Anweisungsfolge abzuarbeiten, ist es notwendig, der CPU mitzuteilen, in welchem Speicherbereich diese Befehle zu finden sind. Die CPU liest in diesem Bereich den Speicher und versucht die gelesenen Informationen als Befehl auszufuehren. Dazu werden die gelesenen Informationen ins Befehlsregister transportiert und steuern von hier den Ablauf in der CPU. In Abhaengigkeit vom konkreten Befehl werden entweder zusaetzliche Informationen aus dem Speicher gelesen, werden Daten zum oder vom Speicher transportiert oder bestimmte logische Verknuepfungen in der ALU vorgenommen. Alle diese Aktivitaeten der CPU sind mit dem Aussenden bestimmter Steuersignale verbunden, die die jeweilige Art der Operation anzeigen. Einen Ueberblick ueber die Steuersignale bei einigen ausgewaehlten Operationen gibt Anlage 10. Zwischenzeitlich waehrend der Befehlsverarbeitung sendet die CPU mit Hilfe des REFRESH-Registers eine Information zum Auffrischen des Speicherinhaltes eventuell angeschlossener dynamischer Speicher aus. Waehrend des Refresh-Zyklus wird der Befehl ausgefuehrt.
War der eben abgearbeitete Befehl ein Verarbeitungs- oder Transportbefehl, dann wird die Verarbeitung mit dem im Speicher fol-

genden Befehl fortgesetzt. Einige Befehle veraendern aber diese Abarbeitungsreihenfolge, sie teilen der CPU mit, in welchem Speicherbereich der naechste Befehl zu finden ist.
Nach erfolgter Programmabarbeitung kann die CPU anhalten oder ein Steuerprogramm bearbeiten, mit dem z. B. die naechste Aufgabe ausgewaehlt werden kann.

### 2.3.2. Holen der Befehle

Fuer die Organisation der Befehlsabarbeitung besitzt die CPU ein besonderes Register, den Befehlszaehler (PC). Der Befehlszaehler besitzt 16 Bitstellen entsprechend der Anzahl der Adressleitungen. Beim Betaetigen der RESET-Taste wird dieses Register auf Null gesetzt. Damit liest die CPU den ersten abzuarbeitenden Befehl auf dem Platz Null.

Aus diesem Grund muss ein Programm ab dieser Stelle beginnen. Um ein solches Programm ab Null nach dem Einschalten zur Verfuegung zu stellen, ist ein nicht fluechtiger Speicher, z. B. ein EPROM notwendig. Befindet sich in diesem Speicherbereich kein Programmspeicher, so findet die CPU zufaellige Bitkombinationen, die als Befehl aufgefasst und abgearbeitet werden.
Es kann also immer nur ein Programm nach dem Einschalten gestartet werden. Das wird im Normalfall ein Steuerprogramm sein, mit dem andere Programme aktiviert werden koennen. Soll ein anderes Steuerprogramm verwendet werden, ist der entsprechende Programmspeicher auszuwechseln. Da Programme auch in den Schreib-Lese-Speicher (RAM) geladen werden koennen, kann der Speicherbereich ab Null als RAM ausgelegt werden. Dann muss aber durch die Hardwareschaltung das Erreichen eines Steuerprogramms sichergestellt werden, welches in einem beliebigen Speicherbereich stehen kann. Es koennen nun beliebige Betriebsprogramme in den Bereich ab Null geladen und verwendet werden, ohne jedesmal den Speicher auswechseln zu muessen. Damit ist ein solches System jeder Aufgabenstellung anpassbar.
Der Bereich ab Null ist noch aus einem anderen Grunde besonders fuer Betriebsprogramme geeignet. Er enthaelt einige ausgewaehlte Adressen, die sowohl von Programmen (sogenannte RESTART-Befehle) als auch im Resultat von externen Ereignissen (sogenannten Programmunterbrechungen) benoetigt werden.
Das Lesen der Befehle oder auch anderer Informationen geschieht durch Aussenden einer Adresse, begleitet von bestimmten Steuersignalen.
Durch eine Speicherverwaltung werden aus bestimmten Stellen dieser Adresse die Auswahl der entsprechenden Speichereinheit sowie eines Speicherbereiches vorgenommen. Der niederwertige Teil der Adresse wird verwendet, um in dem betreffenden Speicherbereich den konkreten Platz zu adressieren.
War die dort vorgefundene Information ein Befehl fuer die CPU so wird automatisch der Befehlszaehler entsprechend der Befehlslaenge erhoeht (inkrementiert) und damit die neue Befehlsadresse bereitgestellt. Wurde der Befehl als ein Verzweigungsbefehl erkannt, wird im Befehlszaehler die neue Adresse bereitgestellt und dann erneut durch Aussenden dieser Adresse ein bestimmter Speicherplatz ausgewaehlt.

## 2.3.3. Die Darstellung von Informationen im Speicher

Bisher wurde immer nur allgemein von "Informationen" gesprochen, die in einem "Speicher" zu finden sind. Diese Informationen waren sowohl Daten als auch Befehle, die in unterschiedlichen Speichertypen aufbewahrt wurden (ROM bzw. EPROM oder RAM). Hinsichtlich ihrer Darstellung im Speicher unterscheiden sich diese Informationen auch nicht; es waere auch meeglich, Daten als Befehle zu betrachten und umgekehrt. Bei einer Abarbeitung durch die CPU kommen dabei selten sinnvolle Ergebnisse zustande.

Es wurde bereits der Speicher mit einer endlichen Anzahl Faecher eines Schranken verglichen, in denen Informationen abgelegt werden koennen. Durch eine Adresse wird die Nummer eines konkreten Faeher bereitgestellt.
Wenn Informationen sowohl gelesen als auch abglegt werden koennen, entspricht das dem Prinzip des Schreib-Lese-Speichers. Als Information kann das Vorhandensein eines Zeichens, einer Markierung oder dergleichen gedeutet werden. Ist diese Markierung dauerhaft (eingraviert), so handelt es sich um einen Nur-Lese-Speicher. In einem Kanten koennen auch mehrere Informationen enthalten sein. Analog dazu sind die Speicher im Mikrorechner aufzufassen.
Eine Funktionseinheit "Speicher" besteht aus einer bestimmten Anzahl adressierbarer Plaetze, wobei jeder Platz zwei verschiedene Zustaende annehmen kann. Diese beiden Zustaende werden durch die Dualziffern "0" und "1" repraesentiert. Die Auswahl dieser Plaetze erfolgt ueber sogenannte Adressleitungen, die Anzahl der Leitungen richtet sich nach der Kapazitaet des Speichers. Der einfachste Aufwand ergibt sich bei der Festlegung der Speicherkapazitaet, d. h. der Anzahl der adressierbaren Speicherplaetze, als ein Vielfaches einer Potenz zur Basis 2. Eine Adressleitung kann zwei Zustaende annehmen, entweder hohen Spannungspegel, sogenannten "H-Pegel" (logisch "1"), als auch niedrigen Spannungspegel, sogenannten "L-Pegel" (logisch "0"). Damit waeren zwei verschiedene Speicherplaetze adressierbar. Zwei Adressleitungen koennen zusammen bereits 4 Zustaende annehmen, damit sind 4 verschiedene Speicherplaetze (mit den Adressen 00, 01, 10 und 11) adressierbar. Demzufolge werden bei 10 Adressleitungen 2 hoch 10=1024=1K Speicherplaetze adressiert.
Damit ergibt sich:

Kapazitaet = 2 hoch n (n = Anzahl der Adressleitungen)

Die CPU U880 hat 16 Leitungen fuer die Bildung von Adressen zur Verfuegung, d. h. sie kann maximal 2 hoch 16 = 65536 ^= 64K Speicherplaetze adressieren.
Eine weitere Eigenschaft einer Speichereinheit ist die Aufrufbreite. Hierunter wird verstanden, wieviel Speicherplaetze gleichzeitig mit einer Adresse angesprochen werden koennen, um die Information parallel zu verarbeiten. Diese Aufrufbreite ist verschieden: bei ROM's und EPROM's betraegt sie 8 Stellen, bei statischen RAM's 1, 4 oder 8 Stellen und bei dynamischen RAM's im allgemeinen eine Stelle. Um die moegliche Verarbeitungsbreite des Mikroprozessors U880 mit 8 Datenleitungen zu nutzen, muessen in einer Speichereinheit mehrere Speicherschaltkreise kombiniert werden, um damit die gewuenschte Aufrufbreite zu realisieren. Das geschieht, indem die Adressanschluesse von acht Speicherschaltkreisen mit den jeweiligen Adressleitungen der CPU verbunden werden. Jeder der Speicherschaltkreise wird an einer Datenleitung angeschlossen, die Auswahl erfolgt fuer alle acht Schaltkreise mit einem gemeinsamen Auswahlsignal.

## 2.4. Grundbegriffe der Software

### 2.4.1. Darstellung von Zahlen

Das Wesentliche bei der Programmabarbeitang besteht in der Veraenderung der eingegebenen Zahlen, um die gewuenschten Ergebnisse zu erhalten. Das gewohnte Dezimalsystem ist fuer die Zahlendarstellung im Mikrorechner nicht geeignet; die zwei moeglichen Zustaende fuehren auf ein anderes Zahlensystem, das sogenannte Dualsystem. Dieses kennt nur die Ziffern 0 und 1, welche mit dem "L"- und "H"-Pegel der Informationsspeicherung identisch sind.
Da aber die Bildung von Zahlen sowohl im Dezimalsystem als auch im Dualsystem nach gleichen Gesetzmaessigkeiten verlaeuft, ist eine Umrechnung unproblematisch und kann durch entsprechende Programme vom Mikroprozessor vorgenommen werden.
Diese Zahlenbildung kann mit folgender Gleichung beschrieben werden:

$$Z = d * x^{n} + d * x^{n-1} + \ldots + d * x^{1} + d * x^{0}$$

wobei bedeuten:

- Z = Zahlenwert
- d = Ziffern innerhalb des Wertebereichs im Zahlensystem
- x = Basis des Zahlensystems
- n = ganzzahliger Exponent

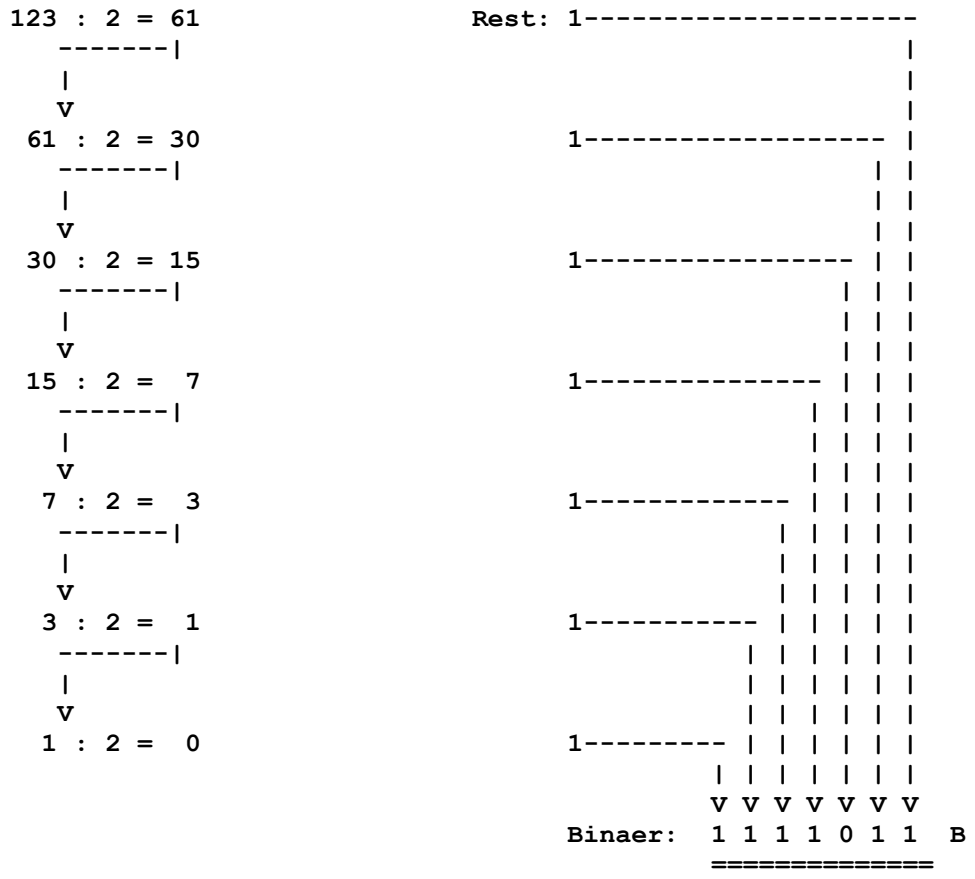
Im Dezimalsystem kann d die Ziffern 0 ... 9 annehmen, x ist dann gleich 10. Im Dualsystem ist d entweder 0 oder 1, die Basis ist gleich 2.
Ein Beispiel soll das verdeutlichen.

$$123 = 1 \times 10^{2} + 2 \times 10^{1} + 3 \times 10^{0} \quad \text{(dezimal)}$$

$$= 1 \times 2^{6} + 1 \times 2^{5} + 1 \times 2^{4} + 1 \times 2^{3} + 0 \times 2^{2} + 1 \times 2^{1} + 1 \times 2^{0}$$

$$= 11111011B \quad \text{(dual)}$$

Das "B" hinter der Dualzahl soll zur Unterscheidung zur Dezimalzahl, die ohne Kennzeichnung geschrieben wird, dienen. "B" bedeutet "binaer", abgeleitet von den zwei Zustaenden. Nun waere eine solche Umrechnung per Hand kompliziert. Es gibt jedoch ein einfaches Umrechnungsverfahren, das am deutlichsten durch ein Beispiel wird.




Die Speichereinheiten in U880 Systemen, wie dem Z1013, besitzen in der Regel eine Aufrufbreite von 8 Bit. Das heisst, auf einem Speicherplatz sind gleichzeitig 8 Bit, die zu einem Byte zusammengefasst werden, gespeichert. Ein Byte kann demzufolge 2 hoch 8 = 256 verschiedene Werte annehmen.
Fuer ein Byte ergeben sich die folgenden Wertigkeiten fuer die einzelnen Bits:
Byte:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Wertigkeit oder Exponent zur Basis 2 |
|---|---|---|---|---|---|---|---|---|
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | Zahlenwert |
| hoeherwertiges | | | | niederwertiges | | | | Halbbyte (BCD-Ziffer) |

Die in der Bytedarstellung eingetragenen Ziffern geben die Numerierung der einzelnen Bit's an. Das Bit 0 besitzt die niedrigste Wertigkeit, das Bit 7 die hoechste.

Eine vorzeichenlose ganze Zahl mit der Bitfolge 01001010B kann auch in der Form:
Z = 0*128 + 1*64 + 0*32 + 0*16 + 1*8 + 0*4 + 1*2 + 0*1 = 74
geschrieben werden.
Sollen auch negative Zahlen dargestellt werden, besitzt das Bit 7 die Funktion des Vorzeichens.
Eine Dualzahl 10110110B kann als
Z = -1*128 + 0*64 +1*32 + 1*16 + 0*8 + 1*4 + 1*2 + 0*1 = - 74
aufgefasst werden, diese Art bezeichnet man als Zweierkomplement.

Damit ergibt sich ein Zahlenbereich fuer ganze vorzeichenlose Zahlen von 0 bis 255 und fuer vorzeichenbehaftete Zahlen von -128 ueber 0 bis +127.
Sollen greessere Zahlen dargestellt werden, muessen 2 und mehr Byte dafuer genutzt werden. Die Zusammenfassung von 2 Byte wird als Wort bezeichnet, analog dazu 4 Byte als Doppelwort.
Die einzelnen Byte des Maschinenkodes werden als Dualzahlen, d. h. als Ziffemfolgen von "0" oder "1" dargestellt. Insbesondere bei grossen Programmen ergibt sich damit ein sehr grosser Schreibaufwand, um diese Dualzahlen zu notieren.
Deshalb hat sich ein anderes Zahlensysten, das sogenannte Hexadezimalsystem fuer die Darstellung von Zahlen und Programmen bei Mikrorechnern durchgesetzt. (Die Bezeichnung Hexadezimalsystem ist umgangssprachlich, exakt heisst es Sedezimalsystem.) Im Hexadezimalsystem werden 4 benachbarte Dualziffern zusammengefasst und durch eine Hexadezimalziffer dargestellt. Mit vier Dualziffern koennen 16 verschiedene Zustaende dargestellt werden. Die Zahlen "0" bis "9" sind gleich den Dezimalzahlen, groesser als "9" werden die ersten Buchstaben des Alphabets verwendet. Die folgende Tabelle enthaelt eine Gegenueberstellung von Dual-, Dezimal- und Hexadezimalziffern.

| DUAL | DEZ | HEX | DUAL | DEZ | HEX |
|---|---|---|---|---|---|
| 0 0 0 0 | 0 | 0 | 1 0 0 0 | 8 | 8 |
| 0 0 0 1 | 1 | 1 | 1 0 0 1 | 9 | 9 |
| 0 0 1 0 | 2 | 2 | 1 0 1 0 | 10 | A |
| 0 0 1 1 | 3 | 3 | 1 0 1 1 | 11 | B |
| 0 1 0 0 | 4 | 4 | 1 1 0 0 | 12 | C |
| 0 1 0 1 | 5 | 5 | 1 1 0 1 | 13 | D |
| 0 1 1 0 | 6 | 6 | 1 1 1 0 | 14 | E |
| 0 1 1 1 | 7 | 7 | 1 1 1 1 | 15 | F |

Da in einem Byte (mit 8 Bit) zwei sogenannte Halbbyte zu je 4 Bit enthalten sind, kann ein Byte mit 2 Hexadezimalziffern dargestellt werden.

Die binaere Darstellung der Dezimaizahlen von 0 bis 9 nennt man auch BCD-Zahlen, Auch mit dieser Zahlendarstellung kann gerechnet werden. Dabei muss aber eine Dezimalkorrektur vorgenommen werden. Warum und wie, wird bei der Erlaeuterung des DAA-Befehls im Befehlssatz genauer erklaert. Zur besseren Unterscheidung zu den Dezimalzahlen verden die Hexadezimalzahlen in Protokollen oder Drucklisten durch ein nachgestelltes Zeichen "H" gekennzeichnet.
Nehmen wir z. B. ein Byte in Binaerdarstellung:

```
     0111   1011B   =  7BH   = 123
      1.     2.     Halbbyte
```

Dabei sind die Wertigkeiten der einzelnen Bits in einem Halbbyte:

```
     3 2 1 0 Wertigkeit
     -------
     8 4 2 1 Zahlenwert
```

Die Umwandlung einer Hexadezimalzahl in die entsprechende Dezimalzahl geschieht am einfachsten auf folgende Weise:

```
                1      0
     7BH  = 7x16 + Bx16
                1       0
          = 7x16 + 11x16
          = 123
```

Die Umrechnung Dezimal- in Hexadezimalzahl erfolgt nach einem analogen Schema wie die Umrechnung Dezimal- in Dualzahl, z. B.

```
                                        Dez.   Hex.
  45 346  :  16  =  2 834      Rest      2      2 -----------
   2 834  :  16  =    177                 2      2 --------- |
     177  :  16  =     11                 1      1 ------- | |
      11  :  16  =     11                11      B ----- | | |
                                                       | | | |
                                                      V V V V
                                  Hex.-Zahl:          B 1 2 2  H
                                                      =======
```

Um den Vorteil dieser Schreibweise deutlich werden zu lassen, hier zum Vergleich diese Zahl in Binaerdarstellung:
1011 0001 0010 OO1OB.

## 2.4.2. Logische Operationen

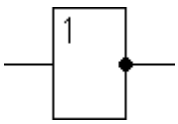
Mit den Dualzahlen lassen sich verschiedene logische Operationen durchfuehren. Bei den logischen Verknuepfungen werden die Dualzahlen als vorzeichenlose, ganze Zahlen aufgefasst.

Die wichtigsten dieser Operationen sind:

- Komplementbildung (NEGATION):
  Eine Dualzahl wird in ihr Komplement ueberfuehrt, indem alle Bitstellen einzeln auf den entgegengesetzten Wert gebracht werden.

```
Zahl:          0 1 0 0 1  0 1 0
--------------------------------
Ergebnis:      1 0 1 1 0  1 0 1
```

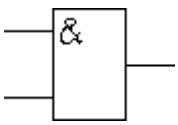
  Diese Operation wird nur mit einer Dualzahl durchgefuehrt. In Stromlaufplaenen finden Sie dafuer das folgende Sinnbild:




- UND-Verknuepfung (KONJUNKION, AND)
  Eine Konjunktion wird mit zwei Dualzahlen durchgefuehrt. Dabei bleibt nur in der Bitposition eine "1" stehen, in welcher in der ersten und in der zweiten Dualzahl eine "1" stehen.

```
1. Zahl:      0 1 0 0 1 0 1 0
2. Zahl:      0 0 0 1 1 1 1 1
-------------------------------
Ergebnis:     0 0 0 0 1 0 1 0
```

  Sinnbild:




  Zur besseren Darstellung der logischen Operationen ist es ueblich, sich eine beliebige Bitposition auszuwaehlen und in einer Wertetabelle alle moeglichen Kombinationen und deren Ergebnisse zu erfassen. Die Wertetabelle der Konjunktion besitzt danach folgendes Aussehen (gleiche Bitposition vorausgesetzt):

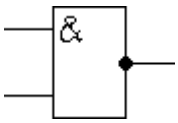
| 1. Zahl | 2. Zahl | Ergebnis |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |

```
    1          |    0           | 0
    1          |    1           | 1
```

Besonders bei komplizierten Verknuepfungen stellt die Wertetabelle ein sehr einfaches Hilfsmittel dar.

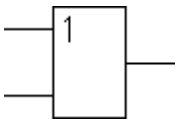
- NICHT-UND-Verknuepfung (NAND)
  Diese Verknuepfung stellt eine Konjunktion mit anschliessender Negation dar.

```
  1. Zahl    |  2.        Sinnbild:
Zahl     |  Ergebnis
  ----------------------
--------------
     0       |
0        |  1
     0       |
1        |  1
     1       |
0        |  1
     1       |
1        |  0
```



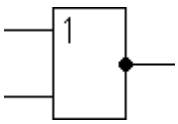

- ODER-Verknuepfung (DISJUNKION, OR)
  Zwei disjunktiv verknuepfte Dualzahlen liefern im Ergebnis eine "1", wenn in der ersten oder zweiten Dualzahl in der jeweiligen Bitposition eine "1" steht.

```
  1. Zahl    |  2.        Sinnbild:
Zahl     |  Ergebnis
  ----------------------
--------------
     0       |
0        |  0
     0       |
1        |  1
     1       |
0        |  1
     1       |
1        |  1
```




- NICHT-ODER-Verknuepfung: (NOR)
  Diese Verknuepfung stellt eine Disjunktion mit anschliessender Negation dar.

```
  1. Zahl    |  2.        Sinnbild:
Zahl     |  Ergebnis
  ----------------------
--------------
     0       |
0        |  1
     0       |
1        |  0
     1       |
0        |  0
     1       |
1        |  0
```




- Exklusiv- ODER bzw. (ANTIVALENZ, EXOR)
  In der jeweiligen Bitposition der Ergebnisse wird eine "1" eingetragen, wenn sich in dieser Bitposition die beiden Dualzahlen unterscheiden.

| 1. Zahl | 2. Zahl | Ergebnis |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Sind beide Dualzahlen gleich, wird das Ergebnis auf Null gesetzt. Das wird besonders verwendet, um einen bestimmten Zwischenspeicher, z. B. das A-Register der CPU, zu loeschen, indem der Inhalt des A-Registers mit sich selbst durch einen XOR-Befehl verknuepft wird (X0R A).

## 2.4.3. Arithmetische Verknuepfungen

Zu den arithmetischen Operationen gehoeren Addition und Subtraktion. Die Multiplikation zweier Dualzahlen kann durch fortlaufende Addition einer Dualzahl bei gleichzeitiger Verringerung der anderen Dualzahl, bis diese Null ist, vorgenommen werden. Auch eine teilweise Addition, kombiniert mit Verschiebung von Ergebnis und Operand ist ueb]ich. Die Division kann analog dazu als eine fortlaufende Subtraktion einer Dualzahl von einer anderen durchgefuehrt werden. Dabei wird der Dividend solange vom Divisor subtrahiert und der Quotient jeweils um 1 erhoeht, bis der Divisor kleiner als der Dividend geworden ist. Der Quotient als Ergebnis enthaelt damit die Anzahl der benoetigten Subtraktionsschritte, im Divisor ist der Rest enthalten.
Nachfolgend die arithmetischen Operationen im einzelnen:
Es empfiehlt sich, die Beispiele mit anderen Zahlen selbst noch einmal nachzuvollziehen.

- ADDITION:
  Die Addition zweier Dualzahlen liefert folgendes in der Wertetabelle sichtbare Ergebnis. Dabei wird der Uebertrag in der letzten Spalte in der naechsthoeheren Bitposition ausgewertet.

```
0   +   0       =   0
0   +   1       =   1
1   +   0       =   1
1   +   1       =   0  Uebertrag 1
```

  Sollen z. B. die Zahlen 26 und 43 miteinander addiert werden, ergibt das folgende Rechnung:

```
26:                0 0 0 1 1 0 1 0
43:                0 0 1 0 1 0 1 1
Uebertraege:           1 1 1   1
---------------------------------
Ergebnis:          0 1 0 0 0 1 0 1  = 69
```

  Werden zwei Zahlen addiert, deren Ergebnis den Zahlenbereich ueberschreitet, kommt es zum Üeberlauf, d. h. das ermittelte Ergebnis ist falsch. An der Addition der Zahlen 69 und 73 soll das im Rechenschema gezeigt werden.

```
69:                0 1 0 0 0 1 0 1   Zahlenbereich:
73:                0 1 0 0 1 0 0 1   -128 <= x <= 127
Uebertraege:       1           1
---------------------------------
Ergebnis:          1 0 0 0 1 1 1 0   = -114
```

  Im Ergebnis entsteht die Zahl -114, obwohl die Addition dieser Zahlen zu dem Ergebnis 132 fuehren muesste. Dieser Ueberlauf ist dadurch charakterisiert, dass ein Uebertrag in die Vor-

zeichenstelle ein-, aber kein Uebertrag aus der Vorzeichen-
stelle herauslaeuft.

- SUBTRAKTION

Die Subtraktion zweier Dualzahlen verlaeuft aehnlich der der Dezimalzahlen, d. h. wenn die Subtraktion einen negativen Wert in der Bitposition ergibt, muss von der hoeherwertigen Stelle etwas "geborgt" werden, es entsteht ein Uebertrag. Daraus resultiert folgende Wertetabelle:

```
1. Zahl 2. Zahl Ergebnis
   0   -   0       =   0
   0   -   1       =   1   (0-1 => 10-1 => 1+Uebertrag)
   1   -   0       =   1           |
   1   -   1       =   0           '->geborgte 1'
```

Subtrahiert man die Dualzahl 26 von der 43, so ergibt sich folgendes Rechenschema:

```
43:                0 0 1 0 1 0 1 1
26:                0 0 0 1 1 0 1 0
Uebertraege:           1
---------------------------------
Ergebnis:          0 0 0 1 0 0 0 1 =  17
```

Subtrahiert man die Zahlen in anderer Weise, d. h. die Dualzahl 43 von der 26, so kann man auch einen Vorzeichenwechsel beobachten.

```
26:                0 0 0 1 1 0 1 0
43:                0 0 1 0 1 0 1 1
Uebertraege: 1<=   1 1   1 1 1 1
---------------------------------
Ergebnis:          1 1 1 0 1 1 1 1 = -17
```

Da hier aber sin Uebertrag sowohl in die Vorzeichenstelle hinein als auch ein Uebertrag aus der Vorzeichenstelle heraus erfolgt, handelt es sich um keinen Ueberlauf und das Ergebnis ist korrekt. Dieser herauslaufende Uebertrag wird bei Zahlen im Wort- oder Doppelwortformat weiterverwendet.

- ZWEIERKOMPLEMENT:

Eine Ergebnisdarstellung wie im vorangegangenen Subtraktionsbeispiel wird Zweierkomplement genannt. Jede Zahl kann in ihr Zweierkomplement ueberfuehrt werden, wenn diese Zahl zuerst in ihr Komplement umgewandelt (negiert) wird und anschliessend zur niederwertigsten Bitposition eine "1" addiert wird.

```
-17:               1 1 1 0 1 1 1 1
Negation:          0 0 0 1 0 0 0 0
Addition:                        1
---------------------------------
Ergebnis:          0 0 0 1 0 0 0 1 = 17
```

Das Zweierkomplement wird verwendet, um eine Subtraktion auf eine Addition zurueckzufuehren.
Die Addition einer Zahl und ihres Zweierkomplements liefert als Ergebnis immer eine Null.
Abschliessend noch ein Beispiel zur Multiplikation, die hier als eine fortlaufende Addition betrachtet werden soll.
Es werden die Dualzahlen 9 und 5 miteinander multipliziert:

```
   Ausgangawerte:  5: 0 0 0 0 0 1 0 1
                   9: 0 0 0 0 1 0 0 1
                                       Multiplika-
                                       tor 5
                   9: 0 0 0 0 1 0 0 1      5
                  +9: 0 0 0 0 1 0 0 1      4
                  -------------------
                   =  0 0 0 1 0 0 1 0      3
                  +9: 0 0 0 0 1 0 0 1
                  -------------------
                   =  0 0 0 1 1 0 1 1      2
                  +9: 0 0 0 0 1 0 0 1
                  -------------------
                   =  0 0 1 0 0 1 0 0      1
                  +9: 0 0 0 0 1 0 0 1
                  -------------------
   Ergebnis:       =  0 0 1 0 1 1 0 1 = 45
```

Die Multiplikation mit teilweiser Addition und Verschiebung kann analog zur Multiplikation von Dezimalzahlen dargestellt werden:

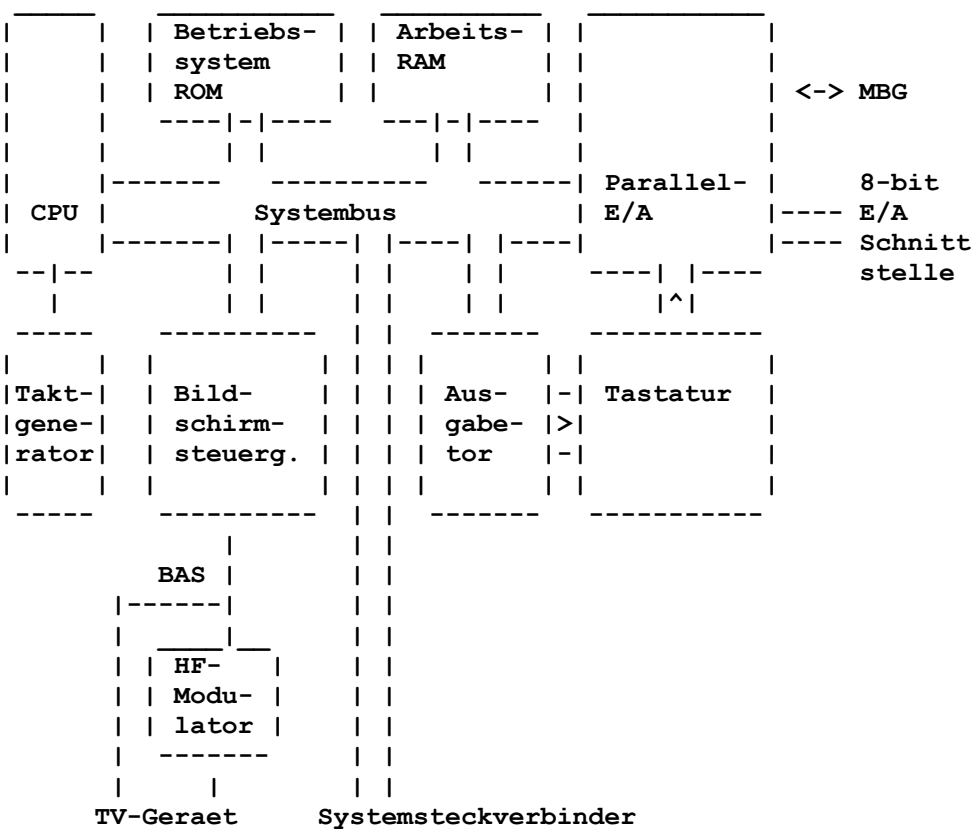
```
Ausgangswerte:     0 0 0 0 1 0 0 1  *  0 1 0 1
-----------------------------------------------
                   0 0 0 0 1 0 0 1  |  1
                 0 0 0 0 0 0 0 0    |  0 = 5
               0 0 0 0 1 0 0 1      |  1
             0 0 0 0 0 0 0 0        |  0
-----------------------------------------------
Ergebnis:    0 0 0 0 0 1 0 1 1 0 1       = 45
```

Bei der teilweisen Addition kann ebenfalls ein Uebertrag auftreten, d. h. der Zahlenbereich ueberschritten werden. Wenn die Kontrolle nicht in jedem Zwischensohritt vorgenommen wird, ist mit fehlerhaften Ergebnissen zu rechnen.

## 3. Hardware des Z1013

Am konkreten Beispiel des MRB Z1013 soll in diesem Kapitel die Arbeitsweise eines Mikrorechners erlaeutert werden. Grundlage dafuer bilden Stromlaufplaene des Z 1013, die Sie in der, Anlage 16 finden.

### 3.1. Blockschaltbild




### 3.2. Steuerung des Mikroprozessors

#### 3.2.1. Beschreibung der Steuersignale

Um den ordnungsgemaessen Betrieb der CPU zu gewaehrleisten, sind bestimmte Steuersignale notwendig. Andere Signale werden von der CPU gebildet und kennzeichnen bestimmte Zustaende waehrend der Abarbeitung von Befehlen. Im folgenden werden alle Steuersignale der CPU und des Systembusses beschrieben. Signale, die mit einem Schraegstrich beginnen, sind sogenannte LOW-aktive Signale, die normalerweise H-Pegel fuehren und bei ihrer Aktivierung L-Pegel zeigen.
Nach dem Signal steht in Klammern ein "A" fuer von der CPU ausgesandte, ein "E" fuer von der CPU empfangene und ein "B" fuer Signale, die sowohl von der CPU empfangen als auch ausgesandt werden koennen.
(A-Ausgabe, E-Eingabe, B-bidirektional, d. h. sowohl Ein- als auch Ausgabe)

- A0 bis A15 (A)
  Sie bilden den 16-Bit-Adressbus. Sie werden in der CPU gebildet und bei der Arbeit mit den Speichern als Speicheradresse sowie die Leitungen A0 bis A7 bei der E/A-Arbeit als E/A-Adresse verwendet.
- A0 bis A6 (A)
  Sie dienen zum Auffrischen dynamischer Speicher.
- D0 bis D7 (B)
  Diese Leitungen stellen den 8 Bit-Datenbus dar. Die Signale koennen sowohl von der CPU gebildet werden (bei Ausgabe oder Speicherschreiben) oder sie werden von den ausgewaehlten Funktionseinheiten erzeugt (bei Eingaben oder Speicherlesen).
- /MREQ (A)
  Dieses Signal wird benoetigt, um eine auf dem Adressbus ausgesandte Adresse zur Speicheradresse zu erklaeren und einen Speicherzugriff durchzufuehren.
- /IORQ (A)
  Das Signal kennzeichnet die auf dem Adressbus anliegende Adresse als Adresse einer E/A-Funktionegruppe. Dabei werden nur die Adreseleitungen A0 bis A7 in die Auswahl einbezogen.
- /M1 (A)
  Es charakterisiert den Maschinenzyklus 1. Dieses Signal wird von der CPU ausgesendet und kennzeichnet in Verbindung mit dem Signal /MREQ, dass vom Speicher ein Befehl geholt wird. In Verbindung mit dem Signal /IORQ wird gekennzeichnet, dass von einem interrupterzeugenden Baustein (s. 4.4.) der sogenannte Interruptvektor gelesen wird (Vektorlesen).
- /RD (A)
  Es wird in den angeschlossenen Funktionseinheiten ausgewertet und legt die Richtung des Datentransportes als "Lesen", d. h. zur Eingabe in die CPU fest.
- /WR (A)
  Es kennzeichnet die Richtung des Datentransportes fuer die angeschlossenen Funktionseinheiten als "Schreiben", d. h. die CPU sendet Daten aus.

- /RFEH (A)
  Zeigt den angeschlossenen Speichereinheiten in Verbindung mit /MREQ, dass auf dem Adressbus eine Refresh-Informaticn verfuegbar ist. Diese Refresh-Information besteht aus einer 7-Bit-Adresse (A0 bis A6), die festlegt, welche Speicherzellen in den dynamischen Speichern; aufgefrischt werden sollen. Das Adressbit 7 kann durch den Programmierer gesetzt oder rueckgesetzt werden und ist Bestandteil der Refresh-Information. Auf dem hoeherwertigen Teil des Adressbusses wird der Inhalt des I-Registers ausgesandt.
- /HALT (A)
  Wird von der CPU ausgesandt, wenn der soeben gelesene Befehl den Operationskode 76H hatte. Die Abarbeitung wird unterbrochen, der Befehlszaehler zeigt auf den naechsten Befehl. Die Refresh-Steuerung wird aufrechterhalten, eine Fortsetzung der CPU-Arbeit ist nur nach Reset oder Interrupt moeglich.
- /WAIT (E)
  Wird von der CPU zu bestimmten Zeiten abgetastet. Ist dieses Signal Low, wird die Arbeit der CPU angehalten, die Informationen auf dem Systembus bleiben erhalten. Anwendung findet dieses Signal vor allem bei der Anpassung der Verarbeitungsgeschwindigkeit von langsamen Funktionseinheiten, indem die Arbeitsgeschwindigkeit der CPU durch solche WAIT-Zyklen der entsprechenden Funktionseinheit angepasst

wird. Waehrend des WAIT-Zustandes findet kein Refresh-Zyklus statt.

- /INT (E)
  Wird von der CPU am Ende eines Befehls abgetastet und signalisiert, dass eine angeschlossene Funktionseinheit das gerade abzuarbeitende Programm unterbrechen moechte, damit von der CPU die Ursache dieser Unterbrechung analysiert und bearbeitet werden kann. Die Ursachen dieser Unterbrechung koennen ein notwendiger Datentransport zwischen CPU und Interface-Baustein sein oder eine Ereignismeldung aus einem zu ueberwachenden Prozess. Die Funktionseinheiten sind untereinander ueber eine sogenannte Prioritaetskette miteinander verbunden, um die jeweils wichtigste Unterbrechung vorrangig zu behandeln. Die CPU kann ihrerseits die Annahme einer Unterbrechung sperren, um z. B. bestimmte Programmabschnitte stoerungsfrei abzuarbeiten. Nach Freigabe des Unterbrechungseinganges wird die dort eventuell gespeicherte Unterbrechung ausgewertet.
- /NMI (E)
  Dieser Eingang stellt aequivalent zum INT-Signal eine Unterbrechungsmoeglichkeit der laufenden CPU-Arbeit dar,

- /NMI (E)
  Dieser Eingang stellt aequivalent zum INT-Signal eine Unterbrechungsmoeglichkeit der laufenden CPU-Arbeit dar, die allerdings nicht gesperrt werden kann. Die Abarbeitung des Unterbrechungsbehandlungsprogramms beginnt ab der Adresse 66H, nachdem zuvor die Fortsetzungsadresse des gerade laufenden Programmes gerettet wurde.
- /RESET (E)
  Unterbricht jede weitere Arbeit der CPU, stellt einen Anfangszustand ein und gibt mit dem Uebergang nach H-Pegel die CPU wieder frei. Da das Reset-Signal meist manuell erzeugt wird, wird durch die Schaltung eine Verkuerzung dieses Signals vorgenommen, um angeschlossenen dynamischen Speichern die Refresh-Informationen zu garantieren.
- C (E)
  Stellt den der CPU zugefuehrten Systemtakt dar. Dieser Takt ist gleichzeitig in allen Funktionseinheiten verfuegbar und sichert die Synchronitaet aller Baugruppen.
- /BUSRQ (E)
  Diese Leitung wird am Ende eines Befehls durch die CPU abgetastet. Dieses Signal kennzeichnet, dass eine angeschlossene Funktionseinheit den Systembus benoetigt, um ihrerseits die Vorgaenge im Mikrorechner zu steuern. Die CPU unterbricht das laufende Programm und setzt ihre Ausgaenge in den hochohmigen Zustand.
  Gleichzeitig wird ein Quittungssignal von der CPU aktiviert, welches den hochohmigen Zustand anzeigt. Dieser bleibt solange bestehen, wie das Signal BUSRQ aktiv ist, d. h. L-Pegel fuehrt. Danach wird das Quittungssignal von der CPU abgeschaltet, alle Ausgaenge nehmen wieder ihr erforderhohes Potential ein und die Abarbeitung wird fortgesetzt. Waehrend des hochohmigen Zustandes kann die CPU keine Refresh-Informationen aussenden.
- /BUSAK (A)
  Ist das Quittungssignal der CPU, welches den hochohmigen Zustand kennzeichnet und damit der den Systembus anfordernden Funktionseinheit den Zugriff erlaubt.

Weiterhin umfasst der Systembus folgende Signale, die nicht von der CPU ausgesandt oder empfangen werden:

- /MEMDI
  Stellt ein Systemsignal dar, mit dem angeschlossene Funktionseinheiten den Zugriff auf Speichereinheiten auf der Leiterplatte der Grundausbaustufe verhindern koennen. Dieses Signal wird erzeugt, wenn Speichererweiterungen die festgelegten Speicheradressen des Grundgeraetes ebenfalls verwenden. Es wird verhindert, dass nicht mehr als eine Speichereinheit den Datenbus benutzen kann.
- /IODI
  Stellt analog zum MEMDI-Signal eine Moeglichkeit dar, bestimmte Adressbereiche auszublenden und Konflikte auf dem Datenbus bei der E/A-Arbeit zu verhindern.
- /IEI und /IEO
  Werden zur Bildung der Prioritaetskette der interrupterzeugenden Funktionseinheiten benoetigt. Jeweils der Ausgang (IEO) der hoeheren Prioritaet wird dem Eingang (IEI) der naechstfolgenden Prioritaetsstufe zugefuehrt (vergleiche auch Abschnitt 4.4 Interruptbehandlung).
  Ein Interrupt kann von einer Funktionseinheit nur ausgeloest werden, wenn das zugefuehrte Signal IEI H-Pegel fuehrt. Gleichzeitig wird das abgegebene Signal IEO auf L-Pegel gehalten. Damit wird sichergestellt, dass immer nur die in der Prioritaetskette am weitesten am Anfang eingereihte Funktionseinheit eine Unterbrechung ausloesen kann.
- /BAI und /BAO
  Stellen analog zu den Signalen IEI und IEO die Signale einer Prioritaetskette dar, die alle Funktionseinheiten verbindet, die eine Anforderung auf den Systembus (BUSRQ) stellen keennen. Fuer die Benutzer des MRB Z1013 werden diese Signale kaum Bedeutung haben.
- RDY
  Stellt ein aehnliches Signal wie WAIT dar, um langsame Funktionseinheiten an die CPU anzupassen. Es kennzeichnet die Kommunikationsbereitschaft einer Funktionseinheit und kann mit der WAIT-Leitung verbunden werden. Im Gegensatz zu den meisten anderen Steuersignalen ist es nicht Low-aktiv.

### 3.2.2. Takterzeugung

Der Taktgenerator wird durch drei Gatter von A6, dem Kondensator C7.1 und den Widerstaenden R38 und R39 gebildet. Stabilisiert wird die Taktfrequenz durch den Schwingquarz Q1. Dieser schwingt mit einer Frequenz von 8 MHz. Der Takt wird dem Binaerteiler A3 zugefuehrt, an dessen Ausgaengen die Taktfrequenzen von 4 MHz, 2 MHz und 1 MHz anliegen. Der Z 1013.01 arbeitet standardmaessig mit 1 MHz Systemtakt, der Z 1013.12 mit 2 MHz. Hinweis: Das Umruesten des Z 1013.01 auf 2 MHz fuehrt zum Erloeschen der Garantie. Die Taktfrequenz 4 MHz ist nicht zugelassen!

Je nach Lage von El erhaelt die CPU den Takt mit der Frequenz entsprechend folgender Zuordnung:

| Lage | Systemtakt |
|---|---|
| E1.1 | 1 MHz |
| E1.2 | 2 MHz |

Mittels des Widerstandes R52 erfolgt noch die erforderliche Pegelanpassung zur Speisung der CPU (A7) und des E/A-Schaltkreises A45.
Dieser Takt realisiert die Synchronitaet aller Zeitablaeufe.

### 3.2.3 RESET-Logik

Um einen definierten Anfangszustand der CPU zu erreichen, ist die RESET-Steuerung erforderlich. RESET kann von 3 Stellen ausgeloest werden:

1. Taste TAl auf der Leiterplatte (RESET-Taste)
2. Externe Tastatur ueber den Steckverbinderanschluss X2:A02
3. A20 des Systemsteckverbinders X1

Eine spezielle Schaltung sorgt dafuer, dass der Datenbustreiber A1 inaktiv wird, d. h. er wird vom Prozessor getrennt. Unmittelbar an der CPU werden die Datenleitungen ueber die Widerstaende R44 ... R51 auf Masse, d. h. L-Pegel gelegt. Da die CPU nach aktiven RESET den Befehlszaehler auf die 0000H einstellt, werden nun auf dieser Adresse die Daten 00H gelesen. Das bedeutet fuer den Prozessor die Ausfuehrung eines sogenannten Leerbefehls (NOP, s. 4.3.15). Bei dessen Ausfuehrung wird der Befehlszaehler um eins erhoeht. Auf diese Art und Weise zaehlen die Adressen hoch, bis die Adresse des Betriebssystems erreicht wird und das Signal /CS aktiviert wird, das den Datenbus mit Hilfe der Logik wieder frei gibt. Als naechstes wird jetzt der erste Befehl des Betriebssystemprogrammes gelesen und dieses wird abgearbeiet.
Damit die Laenge des Reset-Impulses ven der Laenge der Betaetigung unabhaengig wird, wurde ein Monoflop verwendet. Damit wird eine zeitgerechte Aufffrischung der dynamischen Speicher gewaehrleistet. Einige periphere Schaltkreise besitzen keinen Reset-Anschluss. Sie werten das alleinige Auftreten des Signale /M1 als Resetimpuls. Damit auch diese Schaltkreise in einen definierten Anfangszuetand versetzt werden koennen, wurden die Signale /RESET und /M1 zum Signal /PM1 verknuepft, welches die Ruecksetzfunktion ausfuehrt.

## 3.3 Speichereinheiten

### 3.3.1. Anschluss

Der Anschluss der Speicherschaltkreise ist abhaengig vom Typ. Im MRB Z1013 werden drei Arten verwendet
In einem PROM U 2616 bzw. ROM U 2316 (A14) ist das Monitorprogramm enthalten. Dieser Schaltkreis besitzt eine Kapazitaet von 2048 (=2K) Speicherplaetzen, wobei bei jedem Zugriff acht Bit parallel gelesen werden. Um diese 2 KByte zu adressieren, sind 11 Adressleitungen (A0 ... A9) notwendig.
Die verwendeten statischen Schreib-Lese-Speicher besitzen eine Kapazitaet von 1024 (=1K) Plaetzen, wobei jeweils 4 Bit gleichzeitig angesprochen werden. Erst zwei dieser Schaltkreise besitzen deshalb eine Kapazitaet von 1 KByte, wobei 10 Adressleitungen (A0 bis A9) ausreichen.
Mit Hilfe dieser 11 bzw. 10 Adressbits wird jeweils nur ein Byte ausgewaehlt. Die verbleibenden Adressleitungen werden nun dazu verwendet, um einen oder mehrere Speichereschaltkreise auszuwaehlen, damit nur eine Information, und zwar die richtige, bearbeitet werden kann. Die Auswahl des betreffenden Speicherschaltkreisee erfolgt mit dem Adressdekoder A23, der aus einem Bereich von 8 KByte fuer jeden einzelnen 1 KByte-Bereich eine Auswahlleitung bereitstellt. Mit dem Gatter A 24/25 wird dieser Bereich auf den oberen Adressraum eingestellt. Dazu werden mit A25 die betreffenden Adressleitungen mit dem Speicherauswahlsignal MREQ ver-

knuepft und damit der Adressdekoder frei gegeben, d. h.
konkret

```
MREQ  ADR
      15 14 13  12 11 10     9 8 7 6 5 4 3 2 1 0
   0  1  1  1                Diese Leitungen werden an alle
                              Speicherschaltkreise gelegt
                0  0  0  --->/DK10 = E000H = RAM
                0  0  1  --->/DK11 = E400H
                  ...
                0  1  1  --->/DK13 = EC00H = Bildwieder-
                                             holspeicher
                1  0  0  --->/DK14 = F000H = Mit Dioden D9
                1  0  1  --->/DK15           ODER verknuepft
                                             fuer 2K-Monitor
```

Die so gebildeten Leitungen zur Bausteinauswahl (chip-select, CS) werden an den CS-Eingang der Speicherschaltkreise gefuehrt und geben diese frei.

Die Bildung der Auswahlsignale kann ueber das Signal MEMDI am Steckverbinder X1 von ausserhalb verhindert werden. Das wird dann sinnvoll sein, wenn der MRB Z1013 als Bestandteil eines Mikrorechnersysteme betrieben wird und in diesen Adressbereichen bereits Speichereinheiten angeschlossen sind.
Bei der Verwendung der dynamischen 16 KByte Speicher U 256 bzw. K 565 RU3 oder K 565 RU6 (A33 bis A40) ergibt sich folgende Adressauswertung:

```
ADR: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
       0  0 Diese Adressbits werden intern zur
            Auswahl des Speicherplatzes ausgewertet
     Legt den Bereich ab Adresse 0000H fest.
```

Da diese Speicher pro Platz nur 1 Bit speichern, muessen hier 8 Schaltkreise parallel an den Datenbus angeschlossen werden. Der Anschluss ist ausserdem komplizierter, weil diese Schaltkreise nur 6 Adresseingaenge haben. Die Uebernahme der 14-stelligen Adresse erfolgt deshalb zeitlich gestaffelt. Zuerst werden bei Auswahl dieses Speicherbereiches die sieben niederwertigen Adressbits in den Speicherschaltkeis uebernommen. Dazu wird das Signal RAS (ROW ADRESS STROBE, Reihenadressuebernahmeimpuls) am Schaltkreis aktiviert. Anschliessend werden die sieben hoeherwertigen Adressbits auf die Schaltkreisanschluesse geschalten und mit dem Signal CAS (COLUMN ADRESS STROBE, Spaltenadressuebernahmeimpuls) diese in die Schaltkreise eingetragen.
Die Erzeugung des Signals /RAS ist durch das Speicheranforderungssignal /MREQ gegeben. Das negierte Signal MREQ gibt ein FlipFlop (A17) frei. Mit der naechsten steigenden Flanke des Systemtaktes wird das FlipFlop umgeschalten  steuert den Adressleitungsumschalter A28/41 (Multiplexer) und gibt ein zweites Flip-Flop frei, das mit der absteigenden Flanke des Systemtaktes das Signal CAS erzeugt und, sofern ein Zugriff in diesen Adressbereich (A14=A15=0) erfolgt, das Signal /CAS an den Schaltkreisanschluessen aktiviert. War der Zugriff zum Speicher nicht in dem Bereich der dynamischen RAM's oder wurde das Signal /REFRESH als Zeichen der Speicherauffrischungszeit aktiv, so wird zwar das Signal /RAS gebildet, aber kein Signal /CAS. Damit werden in den dynamischen RAM's nur Auffrischungsaktivitaeten ausgeloest.
In Verbindung mit den Signalen /RD oder /WR, die ebenfalls an die Speicherbausteine gefuehrt werden, werden entweder Daten oder Befehle gelesen und auf den Datenbus geschaltet oder die auf dem Datenbus vorhandenen Daten im Speicher eingetragen.
Im Anhang ist ein Schema der Speicherverteilung innerhalb des gesamten Adressraumes zu finden (Anlage 2).

### 3.3.2. Zusammenarbeit mit der CPU

Wenn die CPU auf den Speicher zugreifen moechte, sei es, um Befehle oder Daten zu holen oder um etwas abzuspeichern, ist das durch folgende Signale gekennzeichnet:

- /MREQ (Speicheranforderung) wird Low, d. h. aktiv und zeigt damit den Zugriff auf den Speicher an.
- A0 bis A15 (Adressen) geben den konkret adressierten Speicherplatz an.
- /WRt (Schreiben) wird aktiv, wenn die CPU etwas in den Speicher schreiben moechte.
- /RD (Lesen) wird beim Lesen von Daten oder Befehlen aktiv.
- /M1 (Befehlsholezyklus) kennzeichnet in Verbindung mit /MREQ und /RD das Holen des Operationskodes eines Befehls (s. Kapitel 4)
- D0 bis D7 enthalten entweder die abzuspeichernde oder gelesene Information.

Die detaillierten Zeitablaeufe koennen der Anlage 10 entnommen werden, wo die Taktdiagramme fuer den Speicher-Schreib- und Speichen-Lese-Zyklus angegeben werden.

## 3.4. Ein- und Ausgabebaugruppen

### 3.4.1. Parallel E/A-Baustein U 855 PIO

#### 3.4.1.1. Beschreibung der Steuersignale

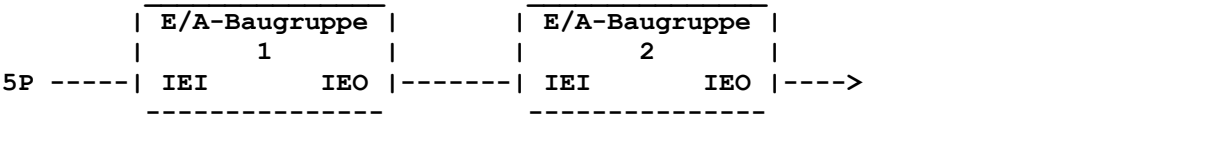
Aus der Bezeichnung des Bausteins geht eigentlich seine Verwendung bereits hervor. Er dient bevorzugt zur parallelen Ein- bzw. Ausgabe, d. h. zum Beispiel, dass alle acht Bit des Datenbusses gleichzeitig ausgegeben werden koennen.
Man kann natuerlich auch Daten seriell, d. h. bitweise nacheinander aus- oder eingeben. Dazu ist aber ein gesondertes Programm notwendig.
Im MRB Z1013 kommt ein Baustein U 855 zum Einsatz. Ein Teil davon wird von den E/A-Baugruppen des Z1013 selbst genutzt (s. 3.4.2., 3.4.3.). Ueber den anderen Teil koennen Sie frei verfuegen. Dazu muessen Sie allerdings die Anschluss- und Funktionsweise einer PIO kennen. Das soll Inhalt dieses Abschnittes sein.
Die Anschlussbelegung des U 855 finden Sie in der Anlage 9.
Es ist zu erkennen, dass die PIO rechnerseitig an den Datenbus angeschlossen wird und prozesseitig zwei Kanaele A und B, auch Tore oder Ports genannt, besitzt.
Ausserdem verfuegt er ueber eine Reihe von Steuersignalen, deren Bedeutung hier kurz erlaeutert werden soll:
Es gelten die gleichen Vereinbarungen wie im Abschnitt 3.2.1.

- B/A SEL (E)
  Liegt dieser Eingang auf "L", so wird das Tor A, liegt er auf "H", dann das Tor B freigegeben. Ueblicherweise wird hieran die Adressleitung der CPU A1 gefuehrt.
- C/D SEL (E)
  Der U 855 ist ein programmierbarer E/A-Baustein, d. h. es muss vor der eigentlichen Nutzung fuer den Datentransfer zwischen Rechner und Prozess mitgeteilt werden, was er machen soll. Dazu gibt es eine Reihe von Steuerwoertern, die die Befehle der PIO darstellen. Diese Programmierung der PIO wird im allgemeinen als Initialisierung bezeichnet. Lesen Sie dazu den Abschnitt 3.4.1.2.
  Erhaelt dieser Eingang "L"-Begel, so sind die auf dem Datenbus befindlichen Informationen Daten, bei "H"-Pegel Steuer-

woerter. Ueblicherweise liegt C/D SEL an der Adressleitung A0.

- /CS (E)
  Hiermit wird die PIO fuer den Datentransfer freigegeben. Die Bildung dieses Bausteinauswahlsignals erfolgt analog zur CS-Dekodierung fuer die Speichereinheiten, nur dass fuer die E/A-Dekodierung die Adressen AO bis A7 (Niederwertiger Teil des Adressbusses) ausgewertet werden. Es koennen also maximal 256 (2 hoch 8) E/A-Tore angeschlossen werden.
- /IORQ (E)
  Dient in Verbindung mit den anderen Signalen zur Kennzeichnung der E/A-Anforderung. Dieser Eingang wird direkt an den entsprechenden Ausgang der CPU gelegt.
- /M1 (E)
  Mit aktiven M1 bei nicht aktiven RD und IORQ wird die PIO in einen definierten Anfangszustand zurueckgesetzt. Geschieht dies nicht, arbeitet die PIO unkontrolliert. Anschliessend muss die Initialisierung erfolgen.
  Ausserdem synchronisiert dieses Signal in Verbindung mit IORQ die Interruptbehandlung durch die CPU. Damit beide Funktionen gewaehrleistet werden koennen, muss dieses M1 aktiv bei aktivem RESET der CPU oder bei Aussendung des CPU-M1 sein. Diese ODER-Verknuepfung wird durch die Bildung des /PM1 realisiert, welches an das PIO-M1 angeschlossen wird.
- RD (E)
  Wird die Datenbusinformation in den PIO geschrieben, muss RD inaktiv sein. Ist RD auf "L", legt die PIO die vom Prozess gelesenen Daten, entsprechend den mit B/A SEL ausgewaehlten Tor, auf den Datenbus.

- C (E)
  Systemtakt analog zur CPU,
- /ASTB (E), /BSTB (E)
  Diese Steuerleitungen werden zur Quittung des erfolgten Datenaustausches verwendet. Zur Ausgabe wird diese Leitung (Low-aktiv) vom angeschlossenen Geraet aktiviert und damit die Daten uebernommen. Bei der Eingabe wird mit dieser Leitung angezeigt, dass die anstehenden Daten in die PIO uebernommen werden koennen. Der Uebergang des Signals /STB aus dem aktiven Zustand in den H-Pegel kann zur Bildung des Interruptsignals verwendet werden.
- ARDY (A), BRDY (A)
  Diese Steuerleitungen teilen dem angeschlossenen Geraet mit, dass bei der Ausgabe Daten bereitstehen, waehrend bei der Eingabe dieses Signale dem Geraet die Bereitschaft zur Datenuebernahme signalisiert.
- /INT (A)
  Dieses Signal liegt parallel zu allen anderen interruptausloesenden E/A-Baugruppen am INT-Eingang der CPU und meldet der CPU, dass eine Unterbrechung des aktuell laufenden Programms erwuenscht wird. Ursache dafuer kann eine Meldung vom Prozess sein, da hier eine Warnung ausgegeben wurde, die unbedingt eine Behandlung erfordert.
- /IEI (E), /IEO (A)
  Hiermit werden die Prioritaeten bei der Behandlung von Unterbrechungsanforderungen durch Bildung einer Prioritaetskette (daisy chain).



Die in einer solchen Kaskade am weitesten links stehende Baugruppe hat den groessten Vorrang. Wird an dieser E/A-Einheit eine Unterbrechung angemeldet, dann wird diese Kette unterbrochen (der Schalter oeffnet), so dass fuer die nachfolgenden Einheiten ein Interrupt gesperrt ist.
Intern besitzt das Tor A gegenueber Tor B hoehere Prioritaet.

### 3.4.1.2. Programmierung

Am Beispiel der im MRB Z1013 verwendeten E/A-Tore soll die Bildung der Auswahladresse erlaeutert werden. Fuer die Ergaenzung der Chip-select Signale wird ein Dekoder A27 eingesetzt, der mit dem E/A-Anforderungssignal die ersten acht Ausgaenge freigibt. Die Festlegung der jeweiligen aktiven IOSEL-Leitung erfolgt dann mit den Adressen A2, A3 und A4.

Mit dem im vorigen Abschnitt zu den 0/13 SEL- ;4nd B/ASEL-Signalen gesagten ergibt sich folgende Adreseverteilung:

```
ADR:    7  6  5  4  3  2  1  0
                              C/D SEL
                              0, wenn Information Daten
                              1, wenn Information Steuerworte
                           B/A SEL
                            0  , wenn Tor A
       beliebig, z.B.       1  , wenn Tor B
       0  0  0  0  0  0  ==>IOSELO, PIO
                0  1  0  ==>IOSEL2, Tastaturspaltentreiber
```

Damit ergeben sich die Adressen:

```
    Tor A (Anwenderport) - Daten:      00H
                         - Steuerwort: 01H
    Tor B (Systemport)   - Daten:      02H
                         - Steuerwort: 03H
```

Im Z1013 sind diese Adressen nicht eindeutig, da die Adressbits A7, A6, A5 auch 111 sein koennten. Da diese nicht ausgewertet werden, spielt das aber keine Rolle,
Die Arbeitsweise der PIO wird durch die Steuerworte festgelegt, die im folgenden erlaeutert werden sollen.

1. Betriebsartenauswahl

```
   Bit:        7   6   5   4   3   2   1   0
   Belegung:                   1   1   1   1  Kennzeichen
                       beliebig
               0   0   Betriebsart    0 :    Byteausgabe
               0   1                   1 :    Byteeingabe
               1   0                   2 :    Byteein/ausgabe
               1   1                   3 :    Bitein/ausgabe
```

   Betriebsart 0:
   Die durch die CPU bereitgestellten Daten werden waehrend des durch sie veranlassten Ausgabezyklus in das angesprochene Ausgaberegister geschrieben. Mit RDY zeigt die PIO dem Prozess an, dass Daten zur Uebernahme in PIO bereitstehen. Dieses RDY wertet das periphere Geraet aus, uebernimmt daraufhin die Daten und teilt mit STB der PIO die Datenuebernahme mit. Anschliessend loest die PIO ein Interrupt aus, um der CPU das Ende der Datenausgabe zu melden.
   Betriebsart 1:
   Die PIO teilt mit H-Pegel an RDY dem externen Geraet die Bereitschaft zur Datenuebernahne mit. Mit STB=L schreibt das Geraet die Daten in das entsprechende Eingaberegister. RDY=L sperrt eine weitere Eingabe bis die Daten durch eine Interruptbehandlung von der CPU uebernommen werden.

Betriebsart 2:
Diese bidirektionale Betriebsart ist nur mit dem Kanal A moeglich. Mit Kanal B ist dann nur noch Betriebsart 3 moeglich, da fuer die Abwicklung des Datentransfers alle vier Quittungssignare ARDY, ASTB, BRDY und BSTB benoetigt werden. ARDY und ASTB steuern die Ausgabe, die beiden anderen die Eingabe.
Betriebsart 3:
In dieser Betriebsart kann innerhalb eines Tores jedem Bit eine beliebige Datenflussrichtung zugeordnet werden. Auf diese Weise koennen Stellsignale und Statusmeldungen fuer Prozess-Steuerungen aus- bzw. eingegeben werden.

2. Ein-/Ausgabe Maskenwort
Soll die Bitstelle eine Eingabeleitung sein, muss an dieser Stelle eine 1 stehen, bei Ausgabe eine 0. Da dieses Steuerwort kein eigenes Kennzeichen besitzt, muss es unmittelbar auf das Betriebsauswahlsteuerwort folgen. Ist in diesem Betriebsart 3 festgelegt worden, liest die PIO das naechste Byte immer als E/A-Maskenwort.

3. Interruptvektor
Bit: 7 6 5 4 3 2 1 0
Belegung: 0 Kennzeichen
niederwertiger Teil des Interruptvektors
(s. 4.4.)

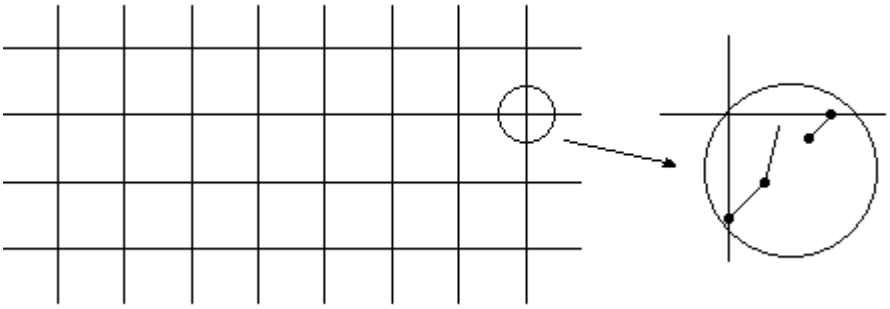
4. Interruptsteuerwcrt
Bit: 7 6 5 4 3 2 1 0
Belegung: 0 1 1 1 Kennzeichen
0 , naechstes Steuerwort ist kein Maskenwort
1 , naechstes Steuerwort wird als Maskenwort erkannt
0 , Interrupt bei H -> L Flanke
1 , Interrupt bei L -> H Flanke
0 , die im folgenden Steuerwort festgelegten Interrupt ausloesenden Bit sind ODER-verknuepft, d. h. eine dieser Leitungen kann bereits Interrupt ausloesen
1 , UND verknuepft, d. h. alle festgelegten Stellen muessen gleichzeitig die mit Bit 5 festgelegte Interruptbedingung erfuellen
0 , Interrupt freigeben
1 , Interrupt gesperrt

5. Interruptmaskenwort
Die Bitztelle der Eingabeleitungen, die Interrupt ausloesen sollen, werden durch eine 0 gekennzeichnet, die kein Interrupt ausloesen sollen durch eine 1.

6. Interrupt Ein/Aus
Bit: 7 6 5 4 3 21 0
Belegung: beliebig 0 01 1 Kennzeichen
0 ,Interrupt gesperrt
1 ,Interrupt freigegeben

## 3.4.2. Tastaturanschluss

Elektrisch stellt die Tastatur nichts anderes als eine Matrix von Schaltern in folgender Anordnung dar:



Die Zeilen dieser Anordnung sind mit den Widerstaenden R11 bis R14 auf "H"-Pegel gelegt. Diese Leitungen sind mit dem Tor B, Bit 0 bis 3, des PIO verbunden, welche fuer Eingabe programmiert sind. Wird keine Taste gedrueckt, liest die PIO auf allen vier Leitungen eine 1.
Die acht Spaltenleitungen der Tastatur sind an ein separates Ausgabetor, das durch die Bausteine A47 (Speicher fuer Spaltennummer) und A46 (1 aus 8 Spaltenleitungen) gebildet wird, angeschlossen. Die Adresse dieses Tores ist 08H. Die Spaltennummer steht im niederwertigen Halbbyte des Datenbusses binaer verschluesselt. Bei einer Ausgabe werden diese vier Bit entschluesselt und legt so eine Spalte auf "L"-Potential. Wird in dieser aktivierten Spalte nun eine Taste betaetigt, wird der L-Pegel auf die entsprechende Zeilenleitung durchgereicht. Der Rechner liest jetzt eine 0 in der entsprechenden Bit stelle.
Aus der ausgegebenen Spaltennunmer und der eingelesenen Zeilennummer ermittelt das Tastaturbedienprogramm des Betriebssystems den rechnerinternen Kode der gerade betaetigten Taste. Der Z1013 benutzt den sogenannten ASCII-Kode (s. Anlage 7).

### 3.4.3. Magnetbandanschluss

Von der auf der Leiterplatte installierten PIO wird eine Bitleitung (PB 7) zur Ausgabe eines seriellen Datenstromes genutzt. Die erforderliche Parallel/Serienwandlung wird softwarenaessig realisiert. Das ausgegebene Signal wird ueber einen Spannungsteiler R27/28 zur Pegelanpassung abgeschwaecht; mit einem Kondensator C1.9 werden die Flanken verrundet, damit ein etwa sinusfoerniges Signal in Magnetbandgeraet aufgezeichnet werden kann.
Das Ausgangssignal eines Magnetbandgeraetes wird gleichspannutigsfrei einem Operationsverstärker A48 zugefuehrt. Das auf TTL-Pegel verstaerkte Signal wird an einen Anschluss der PIO (PB 6) geleitet, Durch entsprechende Software wird dieser Anschluss staendig abgefragt und aus dem ankommenden seriellen Datenstrom durch Serien/Parallelwandlung die urspruengliche Information wieder zurueckgeweonnen.

### 3.4.4. Bildschirmsteuerung

Die Bildschirmsteuerung wandelt die vom Rechner auszugebende Information in ein CCIR-kopatibles Fernsehsignal, indem sie zusaetzlich die notwendigen Synchron- und Dunkeltastimpulse erzeugt. Um diesen Vorgang prinzipiell zu verstehen, sind einige Bemerkungen ueber den Aufbau des Fernsehsignals notwendig.
Beim Schreiben eines Fernsehbildes laeuft ein Elektronenstrahl, auf den die Bildinformation aufmoduliert wurde, ueber einen flueresziererenden Schirm. Fuer eine Zeile benoetigt er eine Zeit von 64 µs.

Das entspricht einer Zeilenfrequenz
von 15,625 kHz. Ein Zeilensynchronimpuls veranlasst den
Strahlruecklauf, wobei der Strahl dunkelgesteuert wird.
Um ein Flimmern der Anzeige zu vermeiden, muss das ganze Bild
mit einer Frequenz von mindestens 25 Hz wechseln.
Da beim Fernsehen in dieser Zeit zwei Halbbilder geschrieben
werden, im Z1013 aber ein Bild zweimal, ergibt sich hier
eine Bildwechselfrequenz von 50 Hz.
Ein sogenannter Bildsynchronimpuls loest dann jeweils einen
Strahlruecklauf zum oberen Bildrand aus. Die Bildschirmsteue-
rung des MRB Z1013 arbeitet nach folgendem Prinzip:
Die gesamte Erzeugung des fernsehgerechten Signals, des soge-
nannten BAS-Signals, wird durch die Zaehlkaskade ohne Mitar-
beit der CPU gesteuert. Die Kaskade A3, A4, A5 und A12 wird
mit dem 8 MHz-Takt des Taktgenerators gespeist. Eine Teilung
durch 2 hoch 9 liefert z. B. die Zeilenfrequenz.
Aus dem Bildaufbau wissen wir bereits, dass eine Zeile aus
32 (=2 hoch 5) Zeichen besteht. Um diese abzuzaehlen, werden
die 5 niederwertigen Adressen des Bildwiederholspeichers (BWS)
A30/31 genutzt. Die hoeherwertigen Adresseingaenge zaehlen die
Zeichenzeilen eines Bildes. Da die Zaehlkaskade immer zyklisch
durchzaehlt, wird auch der BWS zyklisch ausgelesen.
Das aus dem BWS gelesene Byte, das den ASCII-Kode entsprechend
Anlage 7 des darzustellenden Zeichens enthaelt, steht als
hoeherwertiger Adressteil am Zeichengemerator A44. Mit den
drei Ausgaengen des Linien pro Zeichenzaehlers, die an die
niederwertigen Adresseingaenge von A44 gehen, werden nacheinan-
der die Bildpunktzeilen an den nachfolgenden Parallel/Serien-
Wandler A21/22 uebergeben. Hier wird das uebernommene Bitmu-
ster mit dem 8 MHz-Takt seriell herausgeschoben. Dieser seriell
Datenstrom bildet die Bildinformation des Bild-, Austast- und
Synchronsignals (BAS-Signal).
Mit den Gattern der Schaltkreise A9, A10, A13 und A20 werden
aus dem Zaehlfolgen entsprechend der Fernsehnorm  die Synchron-
impulse dekodiert.
Ausserdem wird durch diese Schaltung gesichert, dass fuer der
Strahlruecklauf das Signal dunkelgesteuert wird, da dieser
sonst auf dem Bildschirm sichtbar waere. Diese Impulse werden
mit der Bildinformation gemischt und ergeben so das BAS-Sig-
mal.
In einem HF-Modulator wird das BAS-Signal auf eine HF-Traeger-
frequenz, die auf den Fernsehkanal 3 abgestimmt ist, aufmodu-
liert. Der Ausgang dieses Modulators kann nun direkt mit dem
Antenneneingang des Fernsehgeraetes verbunden werden.
Wie gelangen aber nun in diese selbstaendig arbeitende Einhel
die darzustellenden Daten? Ueber die Adroesmultiplexer (A29,
A42, AlS) kann die CPU einen Platz im BWS adressieren. Dazu
wird mit einem Speicherbereichauswahlsignal der Multiplexer
umgeschaltet. Ueber den Datentreiber A43 kann die CPU den
BWS beschreiben oder lesen.
Damit ist auch deutlich gemacht, dass der BWS wie ein norma-
ler Speicher behandelt werden kann. Die Anfangsadresse ergibt
sich analog zu dem ROM-Auswahlsignal zu EC00H. Welche Position
die einzelnen Speicherplaetze auf dem Bildschirm einnehmen,
ist in der Anlage 8 schematisch dargestellt.

## 3.5. Stromversorgung

Fuer den Betrieb des MRB Z1013 sind drei verschiedene Versorgungsspannungen noetig.
Zur Versorgung aller Logikschaltkreise wird eine Spannung von + 5 V, die im folgenden mit 5P bezeichnet wird und etwa mit 1 A belastbar ist, verwendet. Die beiden anderen Spannungen von + 12 (12P) und - 5 V (5N) werden fuer die Speichereinheiten sowie einige Spezialfaelle benoetigt. Sie werden nicht so stark belastet.
Um diese Spannungen zu erzeugen, besitzt der MRB Z1013 ein eigenes Netzteil. Eine zugefuehrte Wechselspannung von ca. 12 V wird mittels Dioden in Einweggleichrichtung gleichgerichtet. An den Ladekondensatoren C2.1, C3.1 und C5.1 sind jeweile Rohspannungen verfuegbar. Eine Ausnahme bildet die Erzeugung der Rohspannung fuer die 12P. Hier wird mit einer Spannungsverdopplerschaltung gearbeitet.
Die Erzeugung der 5P wird mit einem integrierten Festspanmungsregler A2 vorgenommen, der auf einem Chip alle benoetigten Bauteile enthaelt und kaum eine Aussenbeschaltung benoetigt. Lediglich ein Kondensator am Ausgang ist erforderlich. Da eine starke Belastung dieses Bauelementes erfolgt, wird eine angemessene Kuehlung benoetigt.
Die Spannung 5N wird mittels einer Z-Diode D4 stabilisiert. Diese einfache Widerstands/Z-Dioden-Kombination ist bei dem geringen Leistungsbedarf ausreichend.
Um die Spannung 12P zu erzeugen, wird eine verdoppelte und anschliessend mit einer Widerstands/Z-Dioden-Kombination stabilisierte Spannung der Basis eines Transistors V2 zugefuehrt. Dadurch ist am Emitter dieses Transistors eine stabilisierte Spannung verfuegbar, die staerker belastet werden kann.

## 3.6. Bussystem

Die wichtigsten Signale des Mikrorechners Z1013 sind an den Rand der Leiterplatte gefuehrt und dort fuer den Anschluss von Steckverbindern vorbereitet. Dabei haben diese Anschluesse folgende Bedeutung:

X1: Systembus (Steckverbinder: StL 304-58 TGL 29331/03)
Enthaelt alle Signale des Systembusses und ist elektrisch kompatibel zum K1520-Systembus. (Anlage 6)

X2: Pruefkamm und Tastaturanschlusspunkte (hier wird entsprechend den Hinweisen von Pkt.1.2.4.1. und 1.4. der Bedienungsanleitung das Tastaturbandkabel oder die Buchsenleiste BuL 202-26 TGL 29331/04 angeloetet)

X3: Wechselspannungszufuehrung (Flachsteckverbinder)

X4: PIO Kanal A (Steckverbinder: BuL 402-15 TGL 29331/04)
Hier werden die Anschluesse des Kanals A der PIO herausgefuehrt. Ausser den Steuerleitungen ARDY und /ASTB des Kanals A wurden auch die des Kanals B (BRDY und /BSTB) auf den Steckverbinder gelegt, um die Betriebsart bidirektionale E/A realisieren zu koennen.

X5: Anschluss Magnetbandgeraet (Diodenbuchse)

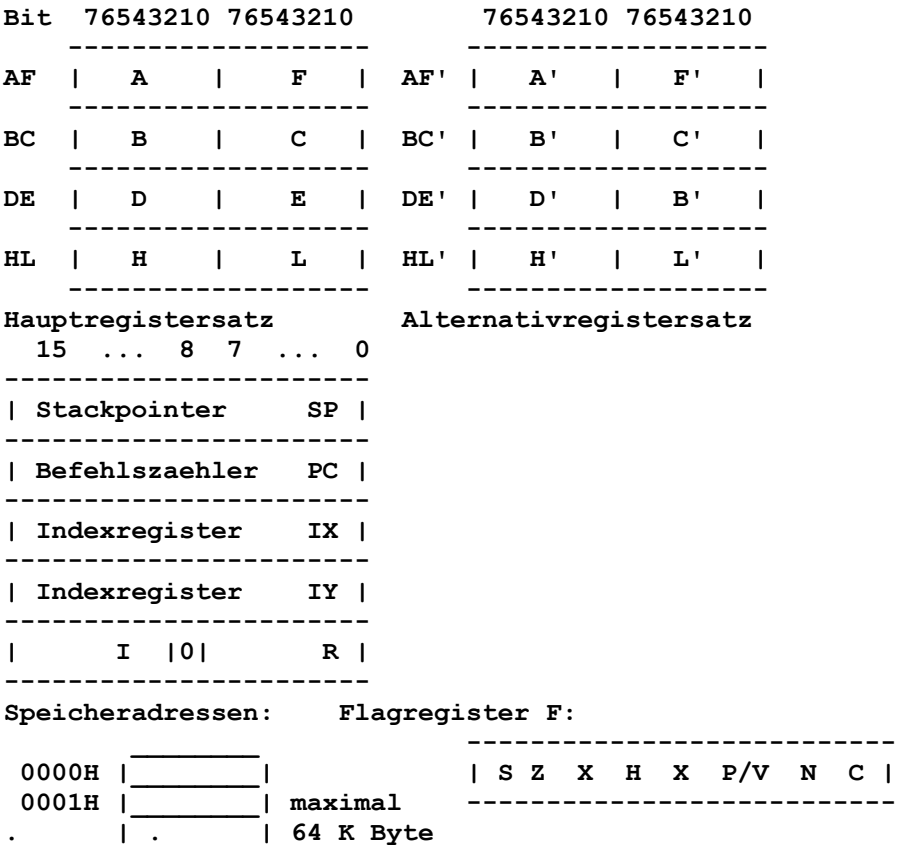
X6: HP-Ausgang des Modulators (Koaxialbuchse)

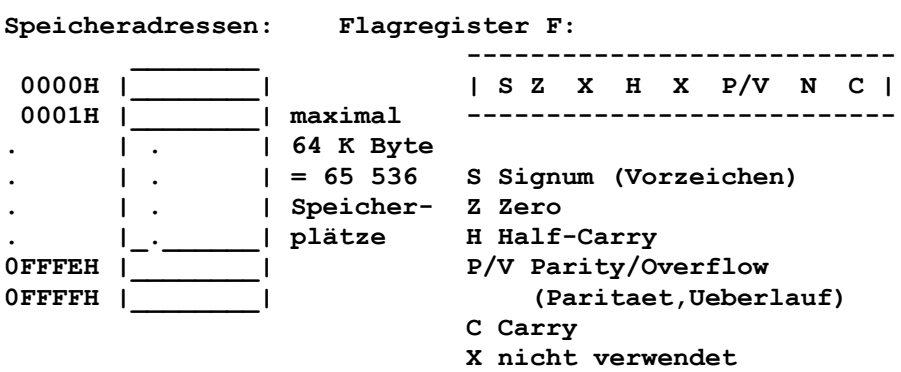
Die genaue Zuordnung der einzelnen Signale zu den jeweiligen Anschluessen ist der Anlage 6 zu entnehmen.

## 4. Der Befehlssatz des Mikroprozessors U880

Dieser Abschnitt soll das Verstaendnis der Arbeitsweise und der Programmierung des Mikrorechnerbausatzes Z1013 erleichtern. Anhand von Beispielen erfolgt eine Erlaeuterung der verschiedenen Mikroprozessor-Befehle und deren Wirkungsweise und Anwendungsmoeglichkeiten. Der folgende Ueberblick soll prinzipielle Eigenschaften und Besonderheiten des Mikroprozessors U 880 aufzeigen:

- 64 K Byte Adressraum fuer Speicher
- 256 Ein-/Ausgabekanaele
- 3 Doppelregister mit Alternativregistersatz
- 2 Indexregister mit je 16 Bit Breite
- 1 Refreshregister (ermoeglicht das automatische Auffrischen externer dynamischer RAM-Speicher)
- 1 maskierbarer, 1 nichtmaskierbarer Interrupt
- Architektur des Mikroprozessors U 880:




Kanaladressen:
0...255 256 Eingaenge
oder
0.. .255 256 Ausgaenge

Befehlsvorrat:
158 Grundbefehle

## 4.1. Befehlsschluessel

Der Mikroprozessor erhaelt seine Befehle vom Speicher ueber den 8 Bit-Datenbus binaer verschluesselt zugefuehrt. Fuer den Programmierer ist diese Darstellung im Binaercode meistens zu detailliert und erschwert die Programmierung; es werden deshalb Hexadezimalkodes mit entsprechend zugehoeriger Mnemonik des Maschinenbefehls verwendet. Als Mnemonik bezeichnet man Pseudonamen der Befehle. Diese Pseudonamen bzw. Abkuerzungen geben gleichzeitig Auskunft ueber die Funktion der Befehle und erleichtern damit die Programmierung. Die Uebersetzung in den Maschinenkode erfolgt dann anhand der Tabelle der Befehlsliste oder durch ein Programm, genannt Assembler.
Bemerkung: Bei hexadezimaler Verschluesselung muss der fuehrenden Ziffer, falls diese ein Buchstabenzeichen ist, eine Null vorangestellt werden. Dadurch wird eine Verwechselung mit Bezeichnern vermieden.
Bei der Programmierung des U 880 werden die Maschinenbefehle im allgemeinen hexadezimal verschluesselt, wobei das 'H' und die fuehrende '0' vor Buchstabenzeichen innerhalb des Maschinenkodes weggelassen werden.

```
z.B.   00 = NOP
       40 = LD C,H
       FF = RST 38H
```

Wie bereits erwaehnt, koennen Maschinenbefehle aus einem oder mehreren Bytes bestehen. Man spricht dann von 1-Byte-, 2-Byte-Befehlen und so weiter. Jedem Byte ist in Abhaengigkeit seines Platzes im Befehl und seiner Kodierung eine bestimmte Bedeutung zugeordnet worden.

### 4.1.1. 1-Byte-Befehle

Diese Befehle bestehen nur aus dem Operationskode (im weiteren als OPC bezeichnet). Da es sich fast ausschliesslich um Registeroperationen, also Operationen innerhalb der Prozessorregister handelt, ist in diesem Byte auch die notwendige Registeradresse enthalten. Bei der Behandlung der Adressierung wird noch einmal naeher darauf eingegangen.

```
 1.  Byte
     OPC
```

Beispiel:

```
Befehls- Maschinen- Quellkode   Kommentar
zaehler  kode
-----------------------------------------------
1000     47         LD B,A      ;Lade B mit A
1001     23         INC HL      ;HL:=HL+1
1002     40         LD B,B
1003     81         ADD C
```

Die Form des Beispiels soll als Normativ eines Programmprotokolls dienen. Es empfiehlt sich, diese uebersichtliche Darstellung bei der Erstellung von Programmen zu nutzen, da sich in dieser tabellenartigen Zusammenstellung alle Angaben widerspiegeln, die zu einem Programm gehoeren. Die letzte Spalte bietet die Moeglichkeit, einen Kommentar unterzubringen, um noch nach laengerer Zeit den Inhalt eines Programms nachvollziehen  zu koennen. Kommentarzeilen werden mit einem Semikolon gekennzeichnet. Der Inhalt sollte kurz, aber eindeutig sein.

Wird nur ein Byte als Operationskode verwendet, wuerden sich 256 Moeglichkeiten ergeben. Man nutzt davon aber nur 252. Die verbleibenden vier Kombinationen sind fuer folgende Aufgaben reserviert worden:
Beim U 880 werden zwecks Erweiterung und Ausbau des Befehlsumfangs vier Hexadezimalkodes (0CBH, 0DDH, 0EDH, 0FDH) der Befehlsliste als sogenannte "Signalbytes" festgelegt. Diese Signalbytes stehen grundsaetzlich an erster Stelle des Befehls (1.Byte). Sie kennzeichnen aber keinen konkreten Befehl, sondern kuendigen eine spezielle Gruppe von Befehlen an. Die Konkretisierung des Befehle erfolgt durch zusaetzlich ein oder mehrere Bytes. Auf diese Art ist es moeglich, durch ein Signalbyte weitere 256 Befehle zu kennzeichnen und somit den Befehlsumfang stark zu erweitern. Der U 880 bietet folgende Erweiterungen des Befehlsschluessels mit den Signalbytes:

| | | |
|---|---|---|
| Signalbyte | CB: | Bitmanipulationen, Verschiebebefehle |
| Signalbyte | DD: | Umschaltung von HL nach IX |
| Signalbyte | ED: | Blocktransport- und Suchbefehle |
| Signalbyte | FD: | Umschaltung von HL nach IY |

## 4.1.2. 2-Byte-Befehle

Diese Befehle koennen jetzt einen zweiten OPC, einen Direktwert oder eine Sprungweite (s. auch Sprungbefehle) im 2. Byte des Befehle enthalten.

| 1. Byte | 2. Byte | |
|---|---|---|
| OPC | OPC | |
| OPC | n | n=Direktwert |
| OPC | c | c=Sprungweite |

Ist das 2. Byte ein OPC, so stellt das 1. Byte das Signalbyte dar.

Beispiele:

```
Befehls- Maschinen- Quellkode   Kommentar
-------------------------------------------
1000     DD 23      INC IX      ;IX:=IX+1
1002     FD 23      INC IY      ;IY:=IY+1
1004     CB 40      BIT 0,B
1006     3E 10      LD A,10H    ;Lade A mit 16
1008     36 FF      LD(HL),0FFH ;Lade den von HL adres-
                         ;sierten Speicherplatz mit 255
100A     28 04      JRZ 06       ;Springe, wenn Z=1 ist,
                    ;um 6 Byte nach vorn, PC: PC + 6 - 2
```

Bei der Berechnung der Sprungweite wird die aktuelle Position des Befehlszaehlers (PC), der ja bereits auf den naechsten Befehl zeigt, durch Subtraktion einer 2 beruecksichtigt. Im Quellkode beziehen sich die Sprungweiten immer auf den Befehlsanfang, also den PC-Stand, der zum betreffenden Befehl gehoert. Es empfiehlt sich, bei der Programmerstellung im Quellkode symbolische Sprungmarken zu verwenden, denen bei der Uebersetzung in Maschinenkode ein konkreter Wert zugewiesen wird. Wie das gemacht wird, zeigen spaetere Beispiele.

## 4.1.3. 3-Byte-Befehle

Diese Befehle enthalten einen Operationskode und einen 16-Bit-Direktwert (nn). Dieser Direktwert stellt einen normalen Datenwert oder eine Adresse dar, wie er z.B. in Lade- oder Sprungbefehlen benoetigt wird. Ebenfalls ist eine Kombination von Signalbyte und OPC sowie von Signalbyte und 8-Bit-Direktwert (n) fuer einige Befehle moeglich.

```
1. Byte 2. Byte 3. Byte
OPC     N(nn)   H(nn)      (bei Ladebefehlen)
OPC     N(nn)   H(nn)      (Sprungadresse)
OPC     OPC       n        (wobei der 1. OPC ein Sig-
                            nalbyte  DD oder FD
                            sein kann)
```

Die Angabe "N(nn)" bezeichnet den niederwertigen Teil, d.h. die letzten 8 Bit des 16-Bit-Direktwertes, "H(nn)" demgemaess den hoeherwertigen Teil, d.h. die ersten 8 Bit.

```
Wert:  |   H  |  N  |
        15...8 7...1
```

Beispiele:

```
Befehls- Maschinen- Quellkode    Kommentar
zaehler  kode
----------------------------------------
1000     21 00 F7   LD HL,0F700H ;Lade HL mit 0F7OCH
1003     C3 2D F0   JMP 0F02DH   ;Springe zur Adresse
                                 ;0F02DH
1006     CD B7 E1   CALL OF1B7H  ;Sprung ins Unterprogr.
1009     DD 46 3F   LD B,(IX+3FH)
100C     FD 72 00   LD (IY+0),D
```

## 4.1.1. 4-Byte-Befehle

In den 4-Byte-Befehlen sind im wesentlichen Kombinationen bereits bekannter Befehle mit einem oder zwei Signalbyte enthalten. Eine Uebersicht ueber diese Befehle ist der Anlage zu entnehmen. Zwei Beispiele sollen hier genuegen.

Beispiele:

```
Befehls- Maschinen-   Quellkode  Kommentar
zaehler  kode
----------------------------------------
1000     DD 21 00 00  LD IX,0    ;Lade IX mit 0
1004     DD CB 0F 1E  RR (IX+0FH)
```

Die im Anhang enthaltene Befehlsliste fuer den Mikroprozessor U 880 enthaelt uebersichtlich alle Befehlsschluessel hexadezimal kodiert mit Hinweisen fuer die Verwendung des Signalbytes bzw. der Verschiebung 'd' (siehe auch 'Indexierte Adressierung'). Diese Befehlsliste dient bei der manuellen Programmierung im Maschinenkode als rationelles Hilfsmittel.

## 4.2. Adressierung

### 4.2.1. Registeradressierung

Die Angabe von Registeroperanden (also von Registern des Notizblockspeichers im U 880) erfolgt implizit im Operations-Byte durch sogenannte Kurzadressen. Fuer die Adressierung von maximal acht allgemeinen Registern sind im OPC-Byte zweimal 3-Bit-Stellen (sowohl fuer das Quellregister als auch fuer das Zielregister) notwendig.

Bei der Angabe der Kurzadressen fuer U880-Register gilt also:

| Register | A | B | C | D | E | H | L | (HL) |
|---|---|---|---|---|---|---|---|---|
| Kurz-adresse | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| (binaer) | 111 | 000 | 001 | 010 | 011 | 100 | 101 | 110 |

| Doppelregister | BC | DE | HL | AF oder SP |
|---|---|---|---|---|
| (binaer) | 00 | 01 | 10 | 11 |

Die Angabe von Operanden, die sich im Hauptspeicher befinden, kann auf verschiedene Weise erfolgen. Der Zugriff auf eine bestimmte Speicherstelle (beim U 880 ist eine 16-Bit-Adresse erforderlich) erfolgt durch Bereit stellen der entsprechenden Adresse zu dem Zeitpunkt, wo der Maschinenbefehl diese benoetigt.

### 4.2.2. Direktwertadressierung

Der Zugriff zum Speicher erfolgt mittels der im Befehl komplett angegebenen Speicheradressen. Diese steht im mnemonischen Befehl immer in Klammern.
z. B.: Der unbekannte 8-Bit-Wert K soll mittels direkter Adressierung vom Speicherplatz mit der Adresse 3000H in das A-Register geholt werden. Danach soll diese Konstante auf den Speicherplatz mit der Adresse 3010H gebracht werden.

```
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------
1000     3A 00 30     LD A,(3000H) ;A:=(3000)
1003     32 10 30     LD (3010H),A ;(3010H):=A=K
```

### 4.2.3. Registerindirektadressierungng

Bei der indirekten Adressierung erfolgt der Speicherzugriff durch Angabe eines Doppelregisters im Maschinenbefehl, wobei im Doppelregister die Adresse fuer den Speicherzugriff geladen sein muss (Registerpaare HL, BC, DE sind moeglich).
z. B.: Der unbekannte 8-Bit-Wert K soll wie im oben genannten Beispiel mittels indirekter Adressierung von Speicherplatz 3000H in den Speicherplatz 3010H umgespeichert werden. Zunaechst laedt man die Adresse des Speicherplatzes, von der der Wert K geholt werden soll, in das Registerpaar BC und die Zieladresse in das Registerpaar HL.

```
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------
1000     01 00 30    LD BC,3000H ;Quelladresse
1003     21 10 30    LD HL,3010H ;Zieladresse
1006     OA          LD A,(BC)   ;A:=(3000H)
1007     77          LD (HL),A   ;(3010H):=A=K
```

Anmerkung: Zur indirekten Adressierung wird gern das Doppelregister HL zur Adressenbereitstellung eingesetzt; man schreibt auch:
(HL)= M (englisch: Memory = Gedaechtnis, Speicher)
Dann wuerde die letzte Zeile des obigen Beispiels lauten:

```
1007     77          LD M,A      ;= LD (HL),A
```

### 4.2.4. Indexierte Adressierung

Soll der Speicherzugriff mittels indexierter Adressierung erfolgen, so wird im Maschinenbefehl ein sogenanntes Indexregister (IX, IY) und eine Verschiebung 'd' als vorzeichenbehaftete Konstante angegeben. Die Adressenbildung fuer den Speicherzugriff erfolgt durch Summierung der im jeweiligen Indexregister enthaltenen Grundadresse und der Verschiebung 'd'. Die Verschiebung 'd' kann Werte von -128 bis +127 annehmen. Bei Verwendung einer negativen Verschiebung wird diese von der Grundadresse subtrahiert.
z. B.: Der 8-Bit-Wert K soll wie im oben genannten Beispiel mittels indexierter Adressierung umgeladen werden. Ins Indexregister IX wird die Adresse 3000H geladen.

```
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------
1000     DD 21 00 30  LD IX,3000H
1004     DD 7E 00     LD A,(IX+0)    ;A:=(3000H),d=0
1007     DD 77 10     LD (IX+16),A   ;(3010H):=A,
                      ;da IX=3000H und d=10h=16
```

Die indexierte Adressierung wird verwendet, um einen einfachen Zugriff zu in Tabellenform gespeicherten Daten zu erhalten. Dazu wird der Anfangspunkt der Tabelle in das Indexregister geladen, die Verschiebung 'd' entspricht dann der konkreten Tabellenposition.

## 4.3. Maschinenbefehle und ihre Bedeutungen

Im folgenden Abschnitt werden die entsprechenden Befehlsgruppen naeher erlaeutert und an Beispielen die Funktionsweise untermauert.

### 4.3.1. Ladebefehle

Bei den Ladebefehlen werden prinzipiell die Byte- und Doppelbyte-Ladebefehle unterschieden. Die Datenbewegung erfolgt stets nur zwischen Speicher und Prozessor bzw. innerhalb des Prozessors zwischen den Registern. Die Flage werden nicht beeinflusst (Ausnahmen bilden nur die Befehle LD A,I und LD A,R).

Allgemeiner Aufbau der Ladebefehle:
[MARKE:] LD Ziel , Quelle [;KOMMENTAR]
Die eingeklammerten Angaben sind wahlfrei, sie koennen vorhanden sein, muessen aber nicht.
Beispiele fuer Byte-Ladebefehle:
(d.h. die transportierten Daten umfassen ein Byte)

```
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------
1000     3E 3A        LD A,3AH    ;A: =3AH
1002     57           LD D,A      ;D: =3AH
1003     46           LD B,(HL)
1004     1A           LD A,(DE)
1005     02           LD (BC),A
1006     DD 70 00     LD (IX+0),B ;B wird auf
         ;IX+d adressierten Speicherplatz gela-
         ;den
```

```
1009     DD 4E 7F     LD C,(IX+127)
100C     ED 57        LD A,I      ;das Interruptre-
         ;gister I wird in A geladen, Stand
         ;von IFF 2 in das Flag P/V
100E     ED 5F        LD A,R       ;das Refreshre-
         ;gister wird in das A-Register geladen
```

Beispiele fuer Doppelbyte-Ladebefehle (d. h. die transportierten Daten umfassen 2 Byte):

```
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------
1000     01 34 12     LD BC,1234H  ;B:=12H, C:=34H
1003     2A 34 12     LD HL,(1234H)
1006     ED 43 34 12  LD (1234H),BC
100k     DD F9        LD SP,IX
```

Es ist verstaendlich, dass Doppelbyte-Ladebefehle auf Grund des 8-Bit-Batenbusses nur byteweise abgearbeitet werden, wobei zunaechst das niederwertige und danach das hoeherwertige Byte geladen wird.
Einen Sonderfall der Boppelbyte-Ladebefehle bilden die sogenannten Kelleroperationen, diese werden aber spaeter beschrieben.

## 4.3.2 Byte- und Doppelbyte-Zaehl-Befehle

Diese Befehlsgruppe dient dem Erniedrigen bzw. Erhoehen von Registerinhalten oder Speicherinhalten um jeweils den Wert 1.

Beispiele fuer Zaehlbefehle:

```
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------
1000     3C           INC A        ;A:=A+1
1001     34           INC (HL)     ;(HL):=(HL)+1
1002     03           INC BC       ;BC:=BC+1
1003     DD 23        INC IX
1005     0D           DEC C        ;C:=C-1
1006     DD 35 00     DEC (IX+0)   ;(Ix+0):=(1X+0)-1
1009     DD 2B        DEC IX       ;IX:=IX-1
```

Die Byte-Zaehlbefehle beeinflussen das Z-Flag. Ist das Resultat des Befehls im behandelten Byte identisch 0, so wird das Z-Flag auf "1" gesetzt, sonst bleibt es "0". Die Doppelbyte-zaehlbefehle boeinflussen keine Flags.
Beispiel: Laden von 3 Speicherstellen mit 00,01,02 ab Adresse 3000H

```
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------
1000     3E 03        LB A,3       ;Anzahl Speicherpl.
1002     21 00 30     LD HL,3000H  ;1. Adresse laden
1005     75       M1: LD (HL),L    ;Speicherpl. laden
1006     23           INC HL       ;Adresse und Wert
                                   ;um 1 erhoehen
1007     BD           CMP L        ;Vergleich L mit A
1008     20 FB        JRNZ M1-#    ;Ruecksprung zu M1,
                                   ;wenn L noch nicht 3
```

## 4.3.3. Arithmetische Befehle

Beim U 880 ist nur die Addition und die Subtraktion von Bytes und Doppelbytes (16-Bit-Worte) moeglich.
Beispiele fuer arithmetische Befehle:

| Befehls-zaehler | Maschinen-kode | Quellkode | Kommentar |
|---|---|---|---|
| 1000 | 80 | ADD B | ;A:=A+B |
| 1001 | 66 3E | ADD 3EH | ;A:=A+3EH |
| 1003 | 86 | ADD (HL) | ;A:=A+(HL) |
| 1004 | 88 | ADC B | ;A:=A+B+CY |
| 1005 | CE 0F | ADC OFH | ;A:=A+0FH+CY |
| 1007 | 92 | SUB B | ;A:=A-D |
| 1008 | DD 9E 04 | SBC (IX+4) | ;A:=A-(IX+4)-CY |



---

Die arithmetischen Operationen laufen prinzipiell in folgender Form ab:
A := A + Operand s
Hierbei ist die Konstruktion ":=" als "ergibt sich aus" zu interpretieren: A ergibt sich aus A plus Operand s. Diese Befehlsgruppe beeinflusst das Flagregister vollstaendig. Die Wirkungsweise der Flag-Bits laesst sich guenstig bei Zahlenbereichsueberschreitungen nach Ausfuehrung der Operationen zeigen:
Beispiel:

```
            A:= -128
            ADD B     ;(B=127,-128,-127)
            B = 127   B = -128    B = -129

A alt   = 1000 0000   1000 0000   1000 0000
+s      = 0111 1111   1000 0000   1000 0001
-----------------------------------------
A alt+s = 1111 1111  10000 0000  10000 0001
CY-F lag
neu     = 0          1           1
Z neu   = 0          1           0
-----------------------------------------
A neu   =  1111 1111  0000 0000   0000 0001
```

Beim Ueberlauf wird das CY-Flag=1, d. h. das Ergebnis der Addition ist falsch.
Fuer die Doppelbytearithmetik gilt mit gewissen Besonderheiten das oben genannte.

| Befehls-zaehler | Maschinen-kode | Quellkode | Kommentar |
|---|---|---|---|
| 1000 | 09 | ADD HL,BC | ;HL:=HL+BC |
| 1001 | ED 7A | ADC HL,SP | ;HL:=HL+SP+CY |
| 1003 | 19 | SBC HL,DE | |
| 1004 | DD 09 | ADD IX,BC | |

Falls bei der ADD-Operation ein Ueberlauf entsteht, wird das Carry-Flag gesetzt (auf "1"). Bei ADC und SBC wird das Flagregister vollstaendig neu bestimmt. Anwendungsgebiet dieser Befehlsgruppe ist insbesondere die sogenannte Adressarithmetik, d. h. wenn mit Adressen gerechnet wird.

## 4.3.4. Vergleichsbefehle

Einen Sonderfall der Arithmetik-Befehle bilden die Vergleichsbefehle. Es wird eine Subtraktion des A-Registerinhaltes mit dem jeweiligen Operanden ausgefuehrt, allerdings werden im Ergebnis der Operation nur die Flags geaendert, der A-Register inhalt bleibt unveraendert.
Beispiele fuer Vergleichsbefehle:

| Befehls-zaehler | Maschinen-kode | Quellkode | Kommentar |
|---|---|---|---|
| 1000 | B8 | CMP B | ;A-B=? |
| 1001 | FE 38 | CMP 38H | ;A-38H=? |
| 1003 | FD BE 05 | CMP (IX+5) | ;A-(IX+5)=? |

Die Flagbeeinflussung funktioniert nach folgender Tabelle:

| | Z | CY |
|---|---|---|
| ist A>s, dann ist A-s>0 : | 0 | 0 |
| ist A=s, dann ist A-s=0 : | 1 | 0 |
| ist A<s, dann ist A-s<O : | 0 | 1 |

(s ist der entsprechende Operand)
In der Praxis schliessen sich i.a. an Vergleichsbefehle entsprechende Verzweigungsoperationen oder bedingte Unterprogrammaufrufe in Abhaengigkeit vom Z- oder/und vom CY-Flag an (siehe unter 'Bedingte Sprungbefehle').

## 4.3.5. Logische Befehle

Die logischen Befehle des U 880 umfassen das UND, das ODER und das EXKLUSIV-ODER. Die Wirkungsweise wurde im Punkt 2.4 ausfuehrlich erlaeutert. Die logische Verknuepfung erfolgt stets mit dem A-Register und kann mit B, C, D, E, H, L, (HL), A, n, (IX+d) und (IY+d) erfolgen. Das Ergebnis der logischen Operation steht nach deren Ausfuehrung immer im A-Register.
Nachfolgend einige Beispiele:

| Befehls-zaehler | Maschinen-kode | Quellkode | Kommentar |
|---|---|---|---|
| 1000 | A0 | AND B | ;A:=A UND B |
| 1001 | AE | XOR (HL) | ;A:=A XOR (HL) |
| 1002 | F6 55 | OR 55H | ;A:=A OR 55H |
| Spezialfall: | | | |
| 1004 | AF | XOR A | ;CY:=O, Z:=O, |
| | | | ;P:=1, A:=00H!!! |

Das Flag-Register wird bei diesen Operationen neu bestimt:
Carry-Flag ist stets 0
Zero-Flag entsprechend dem A-Registerinhalt
Sign-Flag entsprechend dem Bit 7 des A-Registers
Paritaets-Flag entsprechend der Anzahl der 1-Bits des Ergebnisses (P=1, wenn die Anzahl der 1-Bits geradzahlig ist)

## 4.3.6. Spezielle arithmetische Hi1fsoperationen

```
DAA  ; Dezimalkorrektur bei BCD-Verarbeitung
CPL  ; Komplementieren des A-Registers (Einerkomplement)
       A:=/A entspricht einer bitweisen Negation
NEG  ; Negieren des A-Registers (Zweierkomplement>
       A:=-A entspricht A:=/A+1
CCF  ; Komplementieren des Carry-Flags, CY:=/CY
SCF  ; Setzen des Carry-Flags, CY:=1
```

Fuer eine effektive Verarbeitung von Dezimalzahlen koennen diese direkt durch Addition und Subtraktion verarbeitet werden. Dabei koennen, wie dem nachfolgenden Beispiel zu entnehmen ist, unkorrekte Ergebnisse auftreten (Pseudo-ergebnisse). Deshalb wurde der DAA-Befehl zur Erkennung von Pseudoergebnissen und zur anschliessenden Korrektur nach arithmetischen Operationen bereitgestellt. Der DAA-Befehl kann naoh folgenden Befehlen verwendet werden:
ADD, ADC, INC, SUB, SBC, DEC, NEG
Fuer die Funktion des DAA-Befehls uebernimmt das Carry-Flag CY fuer das hoeherwertige Halbbyte des A-Registers {H(A)} und das Half-Carry-Flag H fuer das niederwertige Halbbyte des A-Registers {N(A)} die Ueberwachungsfunktion, d. h. ein Uebertrag aus dem niederwertigen Halbbyte steht im H-Flag, der aus dem hoeherwertigen im CY-Flag.
Ein Beispiel soll die Wirkungsweise des DAA-Befehls verdeutlichen: In jedem Halbbyte steht eine Dezimalzahl im zugelassenen Wertebereich von 0.. .9.
Die Korrektur nach Additionen erfolgt in der Form:

```
N(A) > 9 oder H =  1 ===> A:=A+06H
H(A) > 9 oder CY=  1 ===> A:=A+60H
```

Bei Subtraktionen erfolgt die Korrektur in folgender Form:
(N-Flag = 1)

```
N(A) > 9 oder H = 1 ===> A:=A-06H
H(A) > 9 oder CY= 1 ===> A:=A-60H
```

Zahlenbeispiel: Addition zweier zweistelliger Dezimalzahlen
A:=99, B:=39, A:=A+B ?

```
     A:   1001 1001 = 99        99
     B:   0011 1001 = 39       +39
----------------------------------
A:=A+B:   1101 0010 = D2 falsch !!
          CY=0 H=1!
DAA: +06H 0000 0110 = 38
     +60H 0110 0000 = 60
----------------------------------
     A:   0011 1000 = 38
          CY=1  H=0          = 138
```

Ergebnis: Es entsteht eine dreistellige BCD-Zahl, die sich zu zwei Stellen aus dem A-Registerinhalt und als dritte Stelle aus dem Carry-Bit (CY=1) ergibt:
( CY, H(A), N(A) ) = (138)
Auf die anderen Hilfsoperationen soll hier nicht weiter eingegangen werden. Ihre Bezeichnung erlaeutert die Funktion genuegend.

## 4.3.7. Befehle zur Bit-Manipulation

Diese Befehlsgruppe erlaubt den Einzelbit-Test, das definierte Setzen bzw. Ruecksetzen ausgewaehlter Bits von Registern des Prozessors oder der durch den Inhalt der Register HL, IX und IY adressierten Speicherplaetze.
Der Aufbau und die Funktion eines Bit-Test-Befehls laesst sich wie folgt allgemein beschreiben:

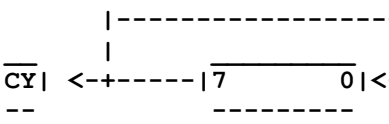
BIT i,s;  Bit-Test-Operation, Testergebnis ist ueber
          Z-Flag auswertbar; Z:=1, wenn das Bit i des
                                   Registers 's' = 0 ist,
                             Z:=0, wenn das Bit i = 1 ist
          0 <= i <= 7

Operanden 's' = A, B, C, D, E H, L, (HL), (IX+d), (IY+d)
Befehle zum Ruecksetzen einzelner Bits haben den Aufbau 'RES i,s' und Befehle zum Setzen 'SET i,s' . Sie beeinflussen keine Flags. Mit dem Befehl 'RES 7,A' wird das Bit 7 des A-Registers auf den Wert 0 gesetzt, mit 'SET 4,B' das Bit 4 auf den Wert 1.

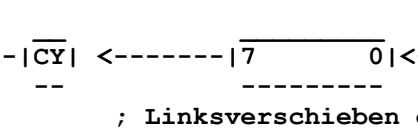
## 4.3.8 Verschiebebefehle

Mit den Verschiebebefehlen erfolgt im entsprechenden Register entweder eine Rechts- oder eine Linksverschiebung um eine Dualstelle, d. h. um ein Bit. Die Besonderheiten der verschiedenen Verschiebebefehle gehen aus den Darstellungen der einzelnen Befehle hervor.

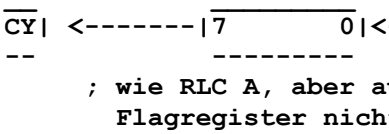
Fuer alle Operanden 's' gilt:
s = A, B, C, D, E, H, L, (HL), (IX+d), (IY+d)
Linksverschieben um eine Dualstelle
RLC s ; Linksverschieben mit Setzen des Carry-Flag's



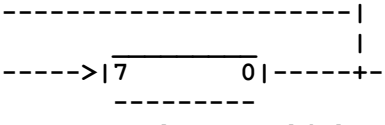

RL s       ; Linksverschieben ueber das Carry-Flag



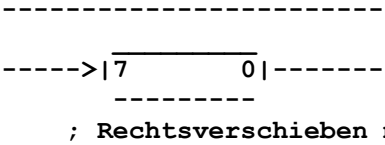

SLA s      ; Linksverschieben durch Carry-Flag



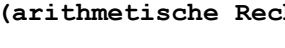

RLCA       ; wie RLC A, aber ausser dem Carry-Flag wird das
             Flagregister nicht beeinflusst.
RLA;       ; wie RL A, aber ausser dem Carry-Flag wird das
             Flagregister nicht beeinflusst.
Rechtsverschiebung um eine flualstelle
RRC s      ; Rechtsverschiebung mit Setzen des Carry-Flags

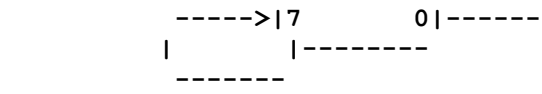



RR s       ; Rechtsverschiebung ueber das Carry-Flag

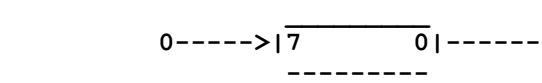



SRA s      ; Rechtsverschieben mit Setzen des Carry-Flags
             (arithmetische Rechtsverschiebung)





```
SRL s       ; Rechtsverschieben mit Setzen des Carry-Flags
              (logische Rechtsverschiebung)
```

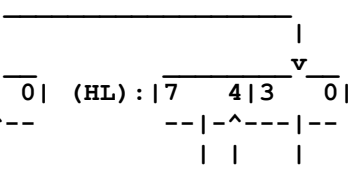


```
RRCA        ; wie RRC A, ausser Carry wird aber das Flag-
              register nicht beeinflusst.
RRA         ; wie RR A, ausser Carry wird aber das Flag-
              register nicht beeinflusst.

Beispiel:     Ein beliebiger Inhalt des A-Registers soll nach
              rechts verschoben werden, bis Bit 0 des A-Regi-
              sters = 1 ist. Wenn das A-Register geloescht
              war, sollen keine Verschiebungen stattfinden.
```

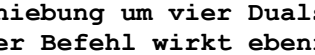
| Befehls-zaehler | Maschinen-kode | | Quellkode | Kommentar |
|---|---|---|---|---|
| 1000 | B0 | | OR A | ;Z = 1, wenn A = 0 |
| | | | | ;ist |
| 1001 | 28 0D | | JRZ END-# | ;Abbruch des |
| | | | ;Programms, wenn A = 0 | |
| 1003 | CB 3F | M1: | SRL A | ;Rechtsverschiebung |
| | | | | ;durch CY |
| 1005 | 30 FC | | JRNC M1-# | ;Ruecksprung zu M1 |
| | | | ;wenn CY=0 ist, also das herausge- | |
| | | | ;schobene Bit eine 0 enthielt | |
| 1007 | CB 17 | | RL A | ;wenn Bit 1, |
| | | | ;muss es aus CY wieder in das A-Regi- | |
| | | | ;ster zurueck | |
| ... | | | | |
| 1010 | 76 | END: | HALT | |

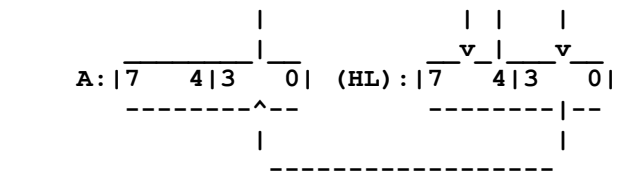
```
Mit einer Verschiebung laesst sich z. B. ein Steuerbit aus dem
Bit 0 oder Bit 7 ins CY-Flag schieben und abtesten, ohne die
anderen Bits zu zerstoeren.
Weitere Anwendungsmoeglichkeiten ergeben sich fuer die Arith-
metik. Beispielsweise laesst sich der SRA-Befehl als Division
durch 2 oder der SLA-Befehl als Multiplikation mit 2 einset-
zen.
Ein Sonderfall der Verschiebungsoperationen ist die Verschie-
bung um vier Dualstellen.
Linksverschiebung um vier Dualstellen
RLD   ;     Der Befehl wirkt zwischen A-Register und dem durch
            HL indirekt adressierten Speicheroperanden.
```



```
Zahlenbeispiel:
   A: 3F                A: 33
         ==> RLD ==>
(HL): 3C             (HL): CF

Rechtsverschiebung um vier Dualstellen
RRD     ; der Befehl wirkt ebenfalls nur zwischen A und (HL)
```






## 4.3.9 Sprungbefehle

Grundsaetzlich werden bedingte und unbedingte Spruenge unterschieden.
- Unbedingte Spruenge:
Bei Erkennen des Sprungbefehls wird vom Prozessor der Befehlszaehler entweder mit der im Adressteil des Befehl angegebenen Sprungadresse geladen (absolute Sprungadresse) oder der Befehlszaehler wird um eine Konstante e veraendert (realtive Sprungweite). Die Konstante e wird im Befehl mit angegeben und liegt im Wertebereich von -128 <= 127. Die Angabe der absoluten Sprungadresse kann indirekt oder indiziert erfolgen. Relative Sprungbefehle bzw. indirekt adressierte Spruenge benoetigen weniger Speicherraum, sind aber bei manueller Programmierung ohne Assembler schwer zu beherrschen.
Allgemein:

```
JMP sadr  ; Befehlszaehler PC nimmt den Wert 'sadr' an.
JR e      ; Befehlszaehler PC wird um den Wert e veraendert.
            PC:=PC+e (Vorwaerts- und Rueckwaertsspruenge
            sind moeglich)
JMP (xx)  ; Befehlszaehler PC nimmt den Wert an, der im
            Registerpaar xx enthalten ist.
```

Registerpaare 'xx' = HL, IX, IY
Die Berechnung der Sprungweite ist mit entsprechender Sorgfalt durchzufuehren.
Die Sprungweite ist eine Differenz von Adressen: Zieladresse minus die Adresse des Befehls, der dem Sprungbefehl folgt. Dabei ist zu beachten, dass ein Relativsprungbefehl immer 2 Byte lang ist. Von der CPU wird prinzipiell nach dem Lesen des Befehls der PC auf den Beginn des physisch naechsten, abzuarbeitenden Befehls gestellt, es sei denn, bei der Abarbeitung des eben gelesenen Befehls stellt sich heraus, dass der PC geaendert werden muss (bedingter Sprung). Relativspruenge beziehen sich dann dabei auf die Adresse des nachfolgenden Befehls. Dieser Umstand ist bei der Sprungweitenberechnung unbedingt zu beachten.

```
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-------------------------------------------------
1000     21 00 10     LD HL,1000H  ;Zaehler laden
1003     75       M1: LD M,L
1004     2C           INC L
1005     20 FC        JRNZ M1-#
1007     24           INC H
...                                ;Programmfort-
...                                ;setzung
```

Sprungweitenberechnung fuer den Sprungbefehl auf PC = 1005H:

```
     Zieladresse:   1003H
 Absprungadresse: - 1007H
 ------------------------
       Differenz:  0FFFCH
```

Es wird nur das niederwertige Byte der Differenz verwendet, also in diesem Falle 0FCH. Man erkennt ausserdem noch folgendes:

Der Betrag der Sprungweite ist groesser 7FH (Bit 7 = 1), das heisst, dass rueckwaerts gesprungen werden muss. Rueckwaerts in dem Sinne, dass der PC um den Absolutbetrag der Sprungweite 's' vermindert wird. Die Sprungweite 's' ist eine vorzeichenbehaftete Konstante.

- Bedingte Spruenge:
Bedingte Spruenge beziehen sich auf den Zustand der durch vorangegangene Operationen entsprechend veraenderten Flags. Auch hier gibt es die Moeglichkeit der absoluten und relativen Sprungmarkenangabe. Der Befehlszaehler wird wie bereits oben genannt veraendert.

```
JPcc adr   ; wenn die Bedigung 'cc'erfuellt ist, nimmt
             der PC den Wert 'adr'an, PC:=adr
  cc kann sein:  C wenn Carry    gesetzt   C=1 ==> JPC  adr
                NC wenn No Carry   "       C=0 ==> JPNZ adr
                 Z wenn Zero       "       Z=1 ==> JPZ  adr
                NZ wenn No Zero    "       Z=0 ==> JPNZ adr
                 P wenn Plus       "       S=0 ==> JPP  adr
                 M wenn Minus      "       S=1 ==> JPM  adr
                PO wenn No Parity  "       P=0 ==> JPPO adr
                PE wenn Parity     "       P=1 ==> JPPE adr

JRcc       ;  wenn die Bedingung 'cc' erfuellt ist, wird der
              PC um den Wert 'e' veraendert (PC:=PC+e)
              cc  = C , ==> JRC e
                    NC, ==> JRNC e
                    Z , ==> JRZ e
                    NZ, ==> JRNZ e
```

Ein Sonderbefehl ist der Zyklusbefehl mit relativer Adressierung.

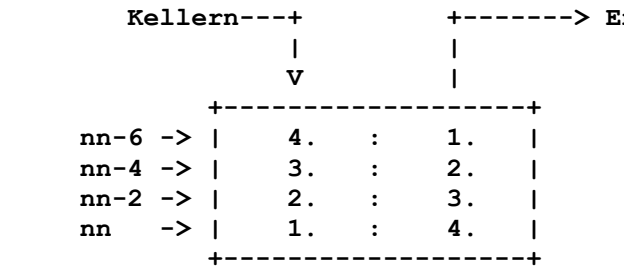
```
DJNZ e       ;Das B-Register wird dekrementiert (um 1 vermindert).
              Solange dessen Inhalt groesser Null ist, erfolgt
              der relative Sprung (PC:=PC+e), sonst wird der
              diesem Befehl folgende aufgerufen.
```

| Befehls-zaehler | Maschinen-kode | Quellkode | Kommentar |
|---|---|---|---|
| 1000 | AF | TIME: XOR A | ;A:=00H |
| 1001 | 47 | LD B,A | ;B:=00H |
| 1002 | 10 FE | DJNZ 2 | ;B:=B-1,B=0 ? |
| | | ;Ruecksprung zum Befehl DJNZ, | |
| | | ;also um 2 Byte zurueck | |
| 1004 | C9 | RET | ;B:=0, Zeit- |
| | | ;schleife abgelaufen | |

## 4.3.10. Kelleroperationen

Die Kelleroperationen sind eine spezielle Gruppe von Transportoperationen. Die Inhalte der Doppelregister werden in einen Speicherbereich gebracht oder von dort geholt, der durch ein spezielles Register (Stackpointer) indirekt adressiert wird. Beim U 880 ist der Kellerspeicherbereich im gesamten Adresebereich sowohl von der Lage als auch von der Groesse frei waehlbar. Allerdings kann nur ein RAM verwendet werden. Zu Beginn eines Programmes muss der Stackpointer (SP) festgelegt werden. Das geschieht mit dem Ladebefehl 'LD SP,nn'. Dieser Zeiger ist also eine Adresse und damit maximal 16 Bit lang.

Sie stellt die Anfangsadresse (den Kontrollboden) des gewuenschten Kellerspeicherbereiches dar.




Die Anwendung der Kelleroperationen liegt vor allem im Retten von Doppelregistern und anschliessendem "Zurueckholen" aus dem Keller zwecks Weiterverwendung in anderen Programmteilen. Das heisst, die geretteten Register koennen bei der Abarbeitung eines Programmteiles anders verwendet werden, denn ihre Inhalte wurden im Keller "sichergestellt".
Auf Grund des 8-Bit-Datenbusses wird deutlich, dass eine Kelleroperation immer in zwei Schritten ablaeuft. Beim Kellern wird zuerst der Stackpointer dekrementiert, das hoeher wertige Byte gekellert, der SP wieder dekrementiert und dann das niederwertige Byte in den Keller gebracht. Das Entkellern erfolgt derart, dass zuerst das niederwertige Byte gelesen und der SP inkrementiert wird. Anschliessend wird das hoeherwertige Byte gelesen und der SP erneut inkrementiert. Nach Kellern und anschliessendem Entkellern zeigt der Stackpointer wieder auf den gleichen Speicherplatz.
Der Kellerbereich wird auch noch von Unterprogrammspruengen und der Interruptorganisation benoetigt, um die erforderlichen Rueckkehradressen aufzubewahren und bereitzustellen.

```
Kellern:      PUSH ss  ;ss= BC, DE, AF, HL, IX, IY
Entkellern:   POP  ss  ;ss= BC, DE, AF, HL, IX, IY
```

Beispiel:
Retten der Register BC, DB, AF und HL in einem Unterprogramm, da diese Register im Unterprogramm benoetigt werden und danach ihre alten Inhalte wieder erhalten sollen.

```
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-------------------------------------------------------
1000     31 00 00     LD SP,0      ;Kellerzeiger
                                   ;laden
1003     CD F1 10     CALL UP1     ;Unterprogr.-ruf
             ;PC wird gekellert:  (FFFFH):=H(PC)=10h
             ;                    (FFFBH):=N(PC)=06H
             ;SP: =FFFEH
1006  ...    ;Programmfortsetzung nach der Rueckkehr
             ;aus dem Unterprogrararn
             ;Unterprogramm UP1
10F1     C5      UP1: PUSH BC      ;SP:=FFFCH
                                    (FFFDH):=B
                                    (FFFCH):=C
10F2     D5           PUSH DE      ;SP=0FFFAH
                                    (FFFBH):=D
                                    (FFFAH):=E
...  ;Unterprogramm
1107     Dl           POP DE       ;SP:=FFFCH
1108     Cl           POP BC       ;SP:=FFFEH
1109     C9           RET          ;Ruecksprung zur
             ;Adresse, die jetzt im Keller oben
             ;an steht, in diesem Falle 1006H
             ;SP:=OOOOH
```

Man erkennt, dass das "Zurueckholen" in genau umgekehrter Reihenfolge erfolgen muss wie das "Retten". Die Ursache liegt darin begruendet, dass der zuletzt gerettete Wert zuerst wieder zurueckgeholt wird und demzufolge der zuerst gerettete Wert als letzter im Stack verfuegbar ist. Im Keller werden also alle geretteten Werte "aufeinandergestapelt" und von diesem Stapel in umgekehrter Reihenfolge wieder herausgeholt.

## 4.3.11. Unterprogrammoperationen

Unterprogramme sind Programmteile, die sich bei der Abarbeitung eines Programmes oft wiederholen. Um unnoetige Programmlaengen zu vermeiden, werden solche Teile nur einmal ins Programm eingefuegt und bei Bedarf als UP aufgerufen. Das heisst: Das Hauptprogramm wird an diesen Stellen verlassen, das Unterprogramm abgearbeitet und dann das Hauptprogramm mit dem Befehl fortgesetzt, der der Aufrufstelle folgt.
Der Aufruf von Unterprogrammen kann sowohl unbedingt als auch bedingt in Abhaengigkeit vom Flagregister erfolgen. Im wesentlichen ist mit dem Unterprogrammaufruf die definierte Aenderung des Befehlszaehlers aehnlich den Sprungbefehlen verbunden, allerdings muss bei Rueckkehr aus dem Unterprogramm der Befehlszaehler PC die Adresse des Befehls enthalten, der als unnimittelbarer Nachfolger des Unterprogrammaufrufes gilt, um genau an dieser Stelle im Programm weiterarbeiten zu koennen. Dies wird realisiert, indem vor dem Sprung ins Unterprogramm die alte PCAdresse, die auf den folgenden Befehl zeigt, gekellert wird und erst dann der Befehlszaehler PC die Unterprogrammadresse erhaelt. Bei Rueckkehr aus dem Unterprograrnin wird diese im Keller gesicherte Adresse wieder in den Befehlszaehler geladen. Das erledigt der Prozessor allein durch den Befehl 'RET'.

```
CALL adr ;Unterprogrammaufruf, es wird die Adresse des
          CALL-Befehls +3 in den Keller gebracht (also der
           nachher notwendige PC-Stand), der PC selbst nimmt
           den Wert adr an (16-Bit-Adresse). Der Stackpointer
           wird um 2 erniedrigt.
RET      ;Unterprogrammrueckkehr, aus dem Keller wird die
          sichergestellte Adresse in den Befehlszaehler PC
          geholt, das Hauptprogramm kann weitergehen; der
          SP wird wieder um 2 erhoeht.
CAcc adr ;bedingter Unterprogrammaufruf, wenn die Bedingung
          cc erfuellt ist, sonst nachfolgender Befehl.
Rcc      ;bedingte Unterprogrammrueckkehr, wenn die Bedin-
          gung cc erfuellt ist, sonst nachfolgender Befehl.
```

Moegliche Bedingungen: cc= NZ, Z, C, NC, P, M, PO, PE
Die Bedingungen der Unterprogrammrufe sind die gleichen wie bei den bedingten Sprungbefehlen.

Einen Sonderfall des Unterprogrammrufes bilden die Restart-Befehle. Hier entfaellt die Angabe einer 2-Byte Sprungadresse. Es erfolgen Spruenge zu festen Adressen (0000H, 0008H, 0010H, 0018H, 0020H, 0028H, 0030H, 0038H).

```
RST hh   ;Unterprogrammruf, der aktuelle PC des RST-Be-
          fehle (konkret der PC-Stand nach dem RST-Befehl)
          wird gekellert. Der PC nimmt den Wert 00hhH an.
          hh=OH, 8H, 10H, ... , 30H, 38H
```

```
Beispiel zur Unterprogrammtechnik:
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------------------
1000     3E FF START: LD A,0FFH    ;Start des Haupt-
                                    programms
1002     77           LD (HL),A
...
100E     CD 34 13     CALL UP1     ;1. Unterprogramm-
                                    ;ruf
                                    ;UP1 beginnt bei
                                    ;Adr.1334H
1012     23           INC  HL      ;Fortsetzung des
                                    ;Hauptprogramms
1013     7E           LD A,(HL)
...
103F     88           ADC  B
...
125E     CD 34 13     CALL UP1     ;2. Unterprogramm-
                                    ;ruf
1261     23           INC  HL      ;Fortsetzung des
                                    ;Hauptprogrammes
...
126A
126B     DD CB 00 DE  SET 3,(IX+0)
...
1333     76           HALT
1334     C5      UP1: PUSH BC      ;Beginn des UP
1335     CB 4F        BIT 1, A
..
1353     Cl           POP BC
1354     C9           RET          ;Rueckkehr ins
                                    ;Hauptprogramm
```

Hinweis:
Es ist unbedingt zu beachten, dass im Unterprogramm die Anzahl der PUSH-Befehle identisch der Anzahl der POP-Befehle sein muss, da sonst bei der Entkellerumg bei Unterprogrammrueckkehr ein falscher Wert in den Befehlszaehler kommt und

Programm undefiniert und nicht ab der Aufrufstelle weiterlaeuft.
Einde Verschactelung mehrerer Unterprogramme (ein UP ruft ein weiteress UP auf ) ist moeglich, die Anzahle der Schachtelungen ist theoretisch unbegrenzt.

## 4.3.12 Ein- und Ausgabebefehle

Wie bereits erwaehnt, sind beim U 880 maximal 256 Ein- und Ausgabekanaele (Peripherieadressen) adressierbar. Zur Adressierun bei Ein- und Ausgaben wird der niederwertige Teil des Adressbusses verwendet. Im Befehl selbst wird die Geraeteadresse (Kanaladresse) entweder direkt ausgegeben oder indirekt ueber das C-Register adressiert. Bei direkter Kanaladresse erfolgt der Datentransport immer zwischen dem A-Register und der Peripherie, bei indirekter Kanaladresse (C-Register muss vorher mit der Kanaladresse geladen werden) kann der Datentransport zwischen einem der Register A, B, C, D, E, H, L, und der Peripherie erfolgen.

```
IN n      ;Eingabe des Kanals n in das A-Register, A:=(n)
IN r      ;Eingabe dec Kanals, dessen Adresse im C-Register
           enthalten ist, nach dem Register 'r'.
```

```
OUT n     ;Ausgabe des A-Registers nach Kanal 'n', (n):=A
OUT r     ;Ausgabe des Registers 'r' nach dem Kanal, dessen
           Adresse im C-Register enthalten ist.
Register 'r' = A, B, C, D, E, H, L
Einen Sonderfall stellt der Eingabebefehl 'INF' dar. Der In-
halt des von Register C adressierten Kanals wird gelesen und
danach entsprechend die Flags gesetzt.
Beispiel: Das Ausgaberegister mit der Adresse 10H soll defi-
          niert mit 00, 01 und 02 geladen werden.
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------------------
1000     0E 10        LD  C,10H     ;C:=10H Kanaladres-
                                    ;se des Registers
1002     A8           XOR A         ;A:=00H
1003     ED 79        OUT A         ;(C):=00H
1005     3C           INC A         ;A:=A+1 = 01H
1006     ED 79        OUT A         ;(C):=01H
1008     3C           INC A         ;A:=A+1 = 02H
1009     ED 79        OUT A         ;(C):=02H
...
1178     76      END: HALT
```

## 4.3.13. Gruppenoperationen fuer Lade-, Vergleichs- und Ein-/ Ausgabe-Befehle

Gruppenoperationen in diesem Sinne sind hardwaremaessig im Prozessor U 880 installierte Befehlsablaeufe, die vom Programmierer lediglich die Generierung bestimmter Register als Parameter verlangen. Diese Befehle vermeiden nicht nur das umstaendliche Programmieren mit einzelnen Befehlen, sondern fuehren besonders zur Programmbeschleunigung und Einsparung von Programmspeicherplaetzen.

- Einfache Gruppenoperationen

```
LDI   ;inkrementierendes Laden
      - die Zieladresse ist in DE zu laden
      - die Quelladresse ist in HL zu laden
      - die Blocklaenge in BC (=Anzahl der im Speicher auf
        einanderfolgenden 'zu transportierenden Byte's).
      Befehlsablauf:  (DE):=(HL)
                      DB:=DE+1
                      HL:=HL+1
                      BC:=BC-1
                      P/V=0 bei BC-1=0
                          1 bei BC-1<>0
LDD   ;dekrementierendes Laden
       wie LDI, nur DE:=DE-1 und HL:=HL-1
CPI   ;inkrementierendes Vergleichen (Suchbefehl)
      - die Anfangsadresse ist in HL zu laden
      - die Blocklaenge ist in BC zu laden
      Befehlsausfuehrung: A-(HL)=?
                          HL:=HL+1
                          BC:=BC-1
                          Z=1 bei A=(HL)
                            0 bei A<>(HL)
                          P/V=0 bei BC-1=0
                              1 bei BC-1<>O
CPD   ;dekrementierendes Vergleichen
       wie CPI, nur HL:=HL-1
INI   ;inkrementierende Eingabe
      - Zieladresse ist in HL zu laden
      - Kanaladresse ist in C zu laden
```

```
      - Blocklaenge in B (max. 256 Byte)
      Befehlsausfuehrung: (HL):=(C)
                          HL:=HL+1
                          B:=B-1
                          Z=1 bei B=0
                            0 bei B<>O

IND   ;dekrementierende Eingabe
       wie INI, nur HL:=HL-1
OUTI  ;inkrementierende Ausgabe
      - Zieladresse ist in C zu laden
      - Kanaladresse ist in Hl zu laden
      - Blocklaenge in B
      Befehlsausfuehrung: (C):=(HL)
                          HL: =HL+1
                          B:=B-1
                          Z=1 bei B=0
                            0 bei B<>O
OUTD  ;dekrementierende Ausgabe
      wie OUTI, nur HL:=HL-1
Bei den einfachen Gruppenoperationen kann der Programmierer
gezielt die Flags testen und entsprechende Ruecksprueng bzw.
die weitere Programmbearbeitung fordern.
Als moegliche Anwendungen ergeben sich:
LDI,  LDD  zum Blocktransport
CPI,  CPD  zur Byte- oder Blocksuche
INI,  IND  Blockeingabe
OUTI, OUTD Blockausgabe
Die Blocklaenge ist dabei in BC bzw. B anzugeben.
- Repetierende Gruppenoperationen (sich automatisch wiederho-
  lende Gruppenoperationen)
Diese Befehle basieren auf den einfachen Gruppenoperationen.
Es werden lediglich die Testung der Flags und entspechende
Ruecksprueng zur Wiederholung des Befehls automatisch ausge-
fuehrt. Allerdings ist bei den Ein- und Ausgabegruppenopera-
tionen zu beachten, dass die peripheren Geraete eine gleiche
Verarbeitungsgeschwindigkeit haben muessen wie der Prozessor,
oder hardwaremaessig ueber die WAIT-Leitung an der CPU eine
Angleichung der Verarbeitungsgeschwindigkeit erfolgen muss.
LDIR  ; wie LDI, Befehl wird wiederholt bis BC:=00H
LDDR  ; wie LDD, Befehl wird wiederholt bis BC:=00H
OPIR  ; wie CPI, Befehl wird wiederholt bis BC:=00H oder A=(HL)
CPDR  ; wie CPD, wird wiederholt bis BC:=00H oder A=(HL)
INIR  ; wie INI, wird wiederholt bis B:=0
INDR  ; wie IND, wird wiederholt tis B:=0
OTIR  ; wie OVIl, wird wiederholt bis B:=0
OTDR  ; wie OUTD, wird wiederholt bis B:=0
Beispiel: Ein Speicherbereich von Adresse 0D00H bis Adresse
          0FFFH soll definiert mit "0E3H" geladen werden.

Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-------------------------------------------------------
1000     21 00 0D    LD HL, 0D00H  ;Quelladresse
1003     3E E3       LD A, 0E3H
1005     77          LD  (HL),A    ;(HL):=0E3H,
                     ;1. Speicherplatz geladen
1006     11 01 OD    LD  DE,0D01H  ;1. Zieladresse
1009     01 FE 02    LD  BC,2FEH   ;Blocklaenge -1
                     ;da ein byte schon geladen wurde -
1000     ED B0       LDIR          ;Befehl bewirkt
                     ;das sogenannte Durchschleifen des Wer-
                     ;tes '0E3H' bis zur Adresse 0FFFH
```

## 4.3.14. Austauschbefehle

```
EXAF      ; Wechsel des Doppelregisters AF gegen AF'
EXX       ; Wechsel des Doppelregistersatzes:
            BC gegen BC'
            DE gegen DE'
            HL gegen HL'
Angewendet wird der Registerwechsel beispielsweise zur Ret-
tung bei Interrupt oder Unterprogramaufruf, da diese Variante
schneller ist als das Kellern bzw. Entkellern der Register.
EX DE, HL ; Tausch der Doppelregisterinhalte
             E-->L und L-->E
             D-->H und H-->D
EX (SP), xx; xx=HL, IX, IY
              Tausch der jeweils obenanstehenden Stackinhalte
              gegen die Inhalte der Doppelregister xx.
              (SP)   gegen L, N(IX) oder N(IY)
              (SP+1) gegen H, H(IX) oder K(IY)
Beispiel:
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-------------------------------------------------------
1000     08     INT1: EXAF       ;Register-
                ;wechsel, da im Interruptprogramm ein
                ;Grossteil der Register benoetigt wird
1001     D9           EXX
1002     3E E3        LD   A,3FH
1004     ...
101B     D3 04        OUT  04
101D     D9           EXX
101E     08           EXAF
101F     FB           EI
1020     ED 4D        RETI
```

## 4.3.15. CPU-Steuerbefehle

```
NOP  ; Es wird keine Operation ausgefuehrt, nur der Befehle-
       zaehler um ein Byte weitergestellt.
HALT ; Prozessor fuehrt intern NOP-Operationen zur Aufrecht-
       erhaltung des Refresch-Zyklus durch.
       Der Befehlszaehler erhaelt die Adresse des nachfolgen-
       den Befehls. Dieser Befehl wird aber erst nach der
       vollstaendigen Ausfuehrung einer Interruptroutine aus-
       gefuehrt. Eine Fortsetzung ist auch mit Reset moeglich.
```

### 4.3.16. Bedeutung der Flags

Ein Grossteil aller Mikroprozessorbefehle beeinflusst definiert entweder das gesamte Flag-Register bzw. nur einzelne Flag-Bits. Die genaue Kenntnis der Bedeutung der Flag-Bits bzw. die Art der Beeinflussung durch verschiedene Befehle ist die wichtigste Voraussetzung fuer die fehlerfreie bzw. optimale Programmerstellung. Die allgemeine Bedeutung der Flag-Bits ist folgender Zusammenstellung zu entnehmen:

S: Vorzeichen (signum)
Ist eine Kopie des Bits A7;
es repraesentiert das echte Vorzeichen des Resultates nach arithmetischen Operationen mit Operanden in Zweierkomplementdarstellung.
S=1, wenn das Ergebnis <0 ist, Abfrage mit cc=M (JPM)
S=0, wenn das Ergebnis >=0 ist, Abfrage mit cc=P (JPP)

Z:  Null (zero)
    Bei einer Null im A-Register (alle 8 Bitpositionen = 0)
    nach arithmetischen und logischen Operationen (auch nach
    CMP) wird Z=1.
H:  Halbbyteuebertrag (half-carry)
    Wird bei einem Uebertrag von Bit A3 nach Bit A4 auf "1"
    gesetzt. (Wird vom DAA-Befehl intern ausgewertet.)
N:  Subtraktion
    N=1, wenn der Befehl eine Subtraktion (auch CMP) war.
    Wird vom DAA-Befehl ebenfalls herangezogen.
CY: Uebertrag (Carry)
    Uebertrag aus dem A-Register nach einer Addition,
    Mittels SO? und CO? ist das OY-Flag auch setz- und
    komplement ierbar.
P/V: Paritaets/Ueberlauf (parity/overflow)
    Dieses Flag hat verschiedene Funktionen.
1.) Es zeigt nach logischen und Verschiebeoperationen und
    nach dem Befehl 'IN r' die Paritaet P an:
    P=1 bei gerader Anzahl vorhandener Bit-Einsen, die Abfra-
    ge kann mit JPPE erfolgen
    Abfrage einer ungeraden Anzahl mit JPPO

2.) Es zeigt im Gegensatz zu Carry den echten Ueberlauf V
    des Resultates nach einer arithmetischen Operation mit
    Operanden an.
    Abfrage mit JPPE
3.) Das Flag sichert nach LD A,I und LD A,R das Bit aus IFF2,
    damit es mit 'LD A,I' wieder in IFF2 gesetzt werden kann.
    In IFF2 wird eine angemeldete Unterbrechung gespeichert,
    bis sie vom Prozessor bearbeitet wird.
4.) Es fixiert nach LD- und CMP-Gruppenoperationen die Aussa-
    ge, ob der Bytezaehler in BC nach Erniedrigung ungleich
    Null ist.
    D. h. P/V=0 bei BC-1=0
          P/V=1 bei BC-1<>O
Die Abfrage der Flag-Bits bei bedingten Operanden erfolgt
nach den im jeweiligen Befehl anzugebenden Bedingungen (siehe
Sprungbefehle) wie folgt:

| Bedingung | Frage | |
|---|---|---|
| NZ : | Z=0? | ===> JPNZ , JRNZ |
| Z : | Z=1? | ===> JPZ , JRZ |
| NC : | C=0? | usw. |
| C : | C=1? | |
| PO : | P/V=0? | |
| PE : | P/V=1? | |
| P : | S=0? | |
| M : | S=1? | |

Die komplette Analyse aller Flag-Bits ist jederzeit moeglich
durch Kellern von AF und anschliessendem Laden der durch den
SP angezeigten letzten zwei Speicherplaetze in ein Doppelre-
gister.
Beispiel:

| Befehls-zaehler | Maschinen-kode | Quellkode | Kommentar |
|---|---|---|---|
| 1000 | F5 | PUSK AF | ;Kellern von AF |
| 1001 | El | POP HL | |
| | | ;H:=(SP+1)=A,L:=(SP)=F | |
| 1002 | LD A,L | | |

Die im Anhang beigefuegte Befehlsliste enthaelt Hinweise auf
die Beeinflussung der Flag-Bits durch die jeweiligen Befehle
sowie die Kodierung der einzelnen Befehle zur manuellen Pro-
grammuebersetzung.

## 4.4. Unterbrechungsorganisation

Um auf Signale aus der Umwelt des Prozessors reagieren zu koennen, kann mit einem Programm eine Leitung (also ein Signal) zyklisch abgefragt werden. Das setzt aber voraus, dass dieses Programm immer dieses Signal abfragt. Eine Reaktion des Rechners erfolgt erst, wenn das Signal durch die Abfrage erkannt wird. Der U 880 verfuegt ueber die Moeglichkeit eines Interrupts. Das heisst, dass ein Signal von einem externen Geraet eine Unterbrechung des laufenden Programms zu einem beliebigen Zeitpunkt verursachen kann, um den Prozessor zur sofortigen Abarbeitung eines Programms zu zwingen, das diese Unterbrechung entsprechend behandelt (Interruptbehandlungsroutine).
Fuer die effektive und schnelle Bearbeitung von Unterbrechungswuenschen der Peripherie stehen beim U 880 ein maskierbarer Interrupt und ein nichtmaskierbarer Interrupt zur Verfuegung (der maskierbare Interrupt kann verboten werden). Wird von der Peripherie ein Signal fuer den nichtmaskierbaren Interrupt erzeugt (L-Pegel am Eingang NMI), so fuegt der Prozessor eine RST-Operation mit der festen Adresse 0066H ein. Es liegt beim Programmierer, inwieweit er in diesem Unterpregramm fuer eine Rettung bestimmter Registerinhalte sorgt. Es gibt keinen Befehl, der den nichtmaskierbaren Interrupt verbietet. Die Anwendung liegt meist bei der Datenrettung vor Erkennen eines Spannungsausfalis des Rechnersystems. Der mankierbare Interrupt kann in 3 unterschiedlichen Betriebsarten auftreten, die in der CPU durch einen Befehl eingestellt werden koennen.
Fuer das Sperren bzw. Freigeben aller drei maskierbaren Interruptarten steht jeweils ein Befehl zur Verfuegung.
Interrupt sperren: DI
Interrupt erlauben: EI
Trifft eine Anforderung an den maskierbaren Interrupt ein (L-Pegel am Eingang INT), wenn er durch 'DI' gesperrt ist, so nimmt der Prozessor diese Anforderung zur Kenntnis und arbeitet diese Anforderung sofort nach dem Wiederfreigeben der Interrupterlaubnis durch den Befehl 'EI' ab.
Maskierbare Interruptarten
Diese 3 Interruptarten werden durch einen der Befehle 'IM0', 'IM1' oder 'IM2' gesetzt.

IM0: In dieser Betriebsart wird nach akzeptiertem Unterbrechungswunsch der Interruptquelle der naechste abzuarbeitende Befehl vom Datenbus geholt und in den Programmlauf eingeschoben.
(Es bietet sich hier an, RST-Befehle zu verwenden. Das sind 1-Byte-Befehle, die einem Unterprogrammsprung entsprechen.) Das heisst aber auch, dass durch das interruptanfordernde Geraet der Befehl auf den Datenbus gelegt wird.

IM1: Nach dem Akzeptieren des Unterbrechungswunsches wird der Befehl 'RST 38H' automatisch ausgefuehrt. Gegebenenfalls muss im Interruptbehandlungsprogramm eine Abfrageroutine eingeleitet werden, welche den "Interrupt-Anmelder" feststellt und danach entsprechende Programme aktiviert.

IM2: Diese Betriebsart stellt fuer den Prozessor die leistungsfaehigste Unterbrechungsbehandlung dar. Man nennt sie auch Vektorinterrupt, weil die Organisation der Behandlungsroutinen ueber Zeiger auf einen Adressvektor realisiert wird.

Bei Unterbrechungsanforderung stellt jede Interruptquelle den fuer die gewuenschte Interruptroutine notwendigen niederwertigen "Zeigerteil" durch Aussenden eines 8-Bit-Wertes auf den Datenbus bereit (sogenanntes Vektorlesen).
Dieser Wert bildet den niederwertigen Teil, der Inhalt des I-Registers den hoeherwertigen Teil einer Adresse einer bestimmten Speicherzelle. Diese und die nachfolgende Speicherzelle beinhalten dann die Adresse des Interruptbehandlunnsprogramms. Dies setzt aber natuerlich das vorbereitende Laden des I-Registers und der entsprechenden Register der Peripheriebausteine voraus (sogenannte Initialisierung).

Beispiel: 3 Peripheriebausteine verlangen jeweils eine andere Interruptroutine. Die Startadressen fuer die jeweiligen Unterprogramme lauten:
INT1:=0FF0H
INT2:=0FFFH
INT3:=2F00H
Aufbau der Startadressentabelle:

```
Zeiger1: 0C00 F0H
         0C01 OFH
Zeiger2: 0C02 FFH <===   N(INT2):=0FFH
         0C03 0FH <===   H(INT2):=0FH
Zeiger3: 0C04 00H ===>   0FFFH=Startadresse
         0C05 2FH
         0C06
```

- Das I-Register muss mit "0CH" geladen werden. Es stellt den absoluten Zeigerteil dar.
- Dem Peripheriebaustein 1 muss fuer das entsprechende Register der variable Zeigerteil mit dem Wert "00" mitgeteilt werden. Entspechend gilt "02H" und "04H" fuer Baustein 2 und 3.

Durch Aneinandersetzen des I-Registers als High-Teil und des variablen Zeigerteils als Low-Teil entsteht der gesamte Zeiger. Das heisst zum Beispiel:
Peripheriebaustein 2 meldet einen Interrupt an. Der Interrupt 2 wird zu gegebener Zeit akzeptiert und der Peripheriebaustein muss jetzt den variablen Zeigerteil im Falle des Zahlenbeispiels 02H, auf den Datenbus legen (Vektorlesen der CPU). Jetzt erfolgt die Bestimmung des gesamten Zeigers.

```
I-Register  0CH (High-Teil)
var. Zeiger   02H (Low-Teil)
---------------------------
   Zeiger 2 0C02H
```

In der Startadresstabelle ist unter der Adresse 0C02H ein 0FFH und unter 0C03H ein 0FH eingetragen, d. h. die Startadresse der zum Peripheriebaustein 2 gehoerenden Interruptbehandlungsroutine lautet 0FFFH.
Auf diese Art, laesst sich nicht nur sehr schnell aus vielen Interruptroutinen die fuer den Peripheriebaustein notwendige herausfinden und aktivieren, es wird auch moeglich, fuer ein und denselben Peripheriebaustein durch Umprogrammieren des "variablen Zeigerteils" eine andere Interruptroutine aufzurufen. Das Akzeptieren einer Interruptanforderung setzt stets das Retten der Fortsetzungsadresse voraus. Dies geschieht wie beim Unterprogrammruf durch Kellern des Befehlszaehlers. Bei Rueckkehr aus der Interruptroutine wird der Befehlszaehler wieder entkellert und es erfolgt somit die Fortsetzung der Programmabarbeitung an der zuvor verlassenen Stelle. Das Kellern und Entkellern fuehrt der Prozessor automatisch aus.

Auch hier gilt zu beachten, dass im Interruptprogramm die Anzahl der Keller- und Entkelleroperationen gleich sein muss. Problematisch ist es, wenn mehrere Peripheriebausteine gleichzeitig eine Unterbrechung anmelden. Vorrang hat stets der Baustein, welcher in der Prioritaetskette an "weitesten vorn" steht. Diese Kette wird durch die Signale 'IEI' und 'IEO' gebildet, die durch alle Bausteine hindurchgefuehrt werden. Waehrend der Abarbeitung einer Interruptbehandlungsroutine bleibt es dem Programmierer ueberlassen, ob er durch Setzen des EI-Befehls die Unterbrechung der Interruptbehandlungsroutine vorzeitig durch einen Peripheriebaustein hoeherer Prioritaet zulaesst oder nicht, da die Annahme einer Unterbrechung weitere Unterbrechungsannahmen ausschliesst. Am Ende einer Interruptbehandlungsroutine muss stets, sofern nicht schon erfolgt, mit dem Befehl 'EI' die Unterbrechungserlaubnis wieder freigegeben werden. EI wird immer erst nach Ausfuehrung des auf EI folgenden Befehls wirksam.

- Rueckkehrbefehle aus dem Interruptprogramm:

```
RETI    ; Rueckkehr aus dem maskierbaren Interruptprogramm
RETN    ; Rueckkehr aus dem nichtmaskierbaren Interruptpro-
          gramm
Hinweis:  RETI bewirkt beim Peripheriebaustein, welcher die
          Interruptroutine ausgeloest hat, das Wiederschlies-
          sen der Prioritaetskette (IEO--high). Somit koennen
          nach RETI auch die Peripheriebausteine niedriger
          Prioritaet einen Interrupt ausloesen.
```

R O B O T R O N

Mikrorechnerbausatz Z 1 0 1 3

Handbuch Teil II

Bestandteile des Handbuches:
---

Handbuch Teil I
Handbuch Teil II
Anlagenteil

# 5. Software des MRB Z1013

## 5.1 Monitor

### 5.1.1. Leistungen des Monitors

Nach dem Einschalten benoetigt der Mikroprozessor eine definierte Befehlsfolge fuer seine Arbeitsfaehigkeit. Ohne ein Betriebsprogramm, dem sogenannten Monitorprogramm, das in einem nichtfluechtigen Speicher stehen muss, sind keinerlei Aktionen der CPU wie Tastaturabfrage oder Ausgaben moeglich. Der Monitor des MRB Z1013 umfasst eine Groesse von 2K Byte und belegt den Adressbereich von F000H bis F700H. Mit diesem Monitor kann der Benutzer Speicherbereiche ansehen, eigene Programme von Hand oder von Magnetband eingeben, diese eingegebenen Programme austesten sowie auf Magnetband abspeichern. Weiterhin ist der Start eigener, ausgetesteter Programme moeglich.
Auch zum Austesten eigener Hardwareerweiterungen kann der Monitor verwendet werden. Um diese Moeglichkeiten umfassend nutzen zu koennen, wurden bestimmte Kommandos festgelegt, die bereits im Abschnitt 1.3. vorgestellt wurden.
Ausser dem unmittelbaren Aufruf bestimmter Monitorleistungen durch Kommandos, auf die hier nicht mehr eingegangen werden soll, enthaelt der Monitor noch eine Reihe weiterer, haeufig verwendeter Programmteile.
Im folgenden werden diese Monitorfunktionen beschrieben. Dazu wird ausserdem die Adresse, unter der dieses Unterprogramm aufrufbar ist, und ein Datenbyte (DB), ueber dessen Verwendung weiter unten noch etwas gesagt wird, angegeben. In verschiedenen Faellen muessen dem Programm bestimmte Parameter zur Verfuegung gestellt werden.
Das ist einmal ueber die CPU-Register und zum anderen ueber bestimmte Arbeitszellen des Monitors, die sich im RAM-Bereich befinden, moeglich. In die letztgenannten Zellen werden die erforderlichen Parameter mit dem M-Kommando geschrieben.
Die Adressen einiger ausgewaehlter Arbeitszellen des Monitors sind:

| Name | Adresse | Anzahl der Byte | Bedeutung |
|---|---|---|---|
| SOIL | 0016 | 2 | Anfangsadresse der Eingabezeile (Eingabepuffer) |
| ARG1 | 001B | 2 | 1. Parameter eines Kommandos |
| ARG2 | 001D | 2 | 2. Parameter |
| AR03 | 0023 | 2 | 3. Parameter |

CURSR 002B  2  Kursoradresse

Nachfolgend die Monitorfunktionen:
OUTCH (OUT CHARACTER)
Adr. F21BH  DB 00H
Ausgabe des im A-Register stehenden Zeichens ueber den Video-treiber. 4 Steuerzeichen werden vom Z1013-Video-Treiber spe-ziell verarbeitet:
08H - Kursor links
09H - Kursor rechts
OCH - Bildschirm loeschen
ODH - Neue Zeile; bei Erreichen des unteren Bildschirmrandes wird gerollt
INCH (IN CHARACTER)
Adr. F20CH  DB 01H
Mit dieser Routine wird die Eingabe eines Zeichens von der Tas-tatur in das A-Register realisiert. Dabei wird die Routine INKEY genutzt. Die Register BC, DE und HL werden gerettet. Der Rueck-sprurng aus INCH erfolgt nur bei (A) ungleich 0. Ansonsten be-findet sich der Monitor in einer Eingabewarteschleife.
PRST7 (PRINT STRING mit dem 7. Bit als Endzeichen)
Adr. F2A5H  DB 02H
Die der Datenbytedefinition (DB 2) folgende Bytekette wird aus-gegeben, bis das 7. Bit gesetzt ist. Graphikzeichen (80H..FFH) sind also mit dieser Routine nicht ausgebbar. Fuer diesen Zweck ist ein Unterprogramm mit OUTCH aufzubauen.
INHEX (Konvertierung einer max. 4-steillgen hexadezimalen Zeichenkette in das interne Format)
Adr. F2F4H  DB 03H
Die Routine realisiert die Konvertierung einer max. 4-stelligen hexadezimalen Zeichenkette in das Format eines Doppelregister-inhaltes. Die Anfangsadresse muss im DE-Register uebergeben wer-den. Fuehrende Leerzeichen werden ueberlesen. Das einer max. 4-stelligen hexadezimalen Zeichenkette folgende Leerzeichen bzw. jedes andere Zeichen, welches verschiede von den Hexa-Zeichen ist, wird als Trennzeichen interpretiert. Der konvertierte Wert steht im HL-Register. Bei laengeren Zeichenkettern erfolgt keine Fehlerausschrift, sondern im HL-Register befindet sich der ge-wandelte Wert der letzten 4 Hexa-Zeichen.
INKEY (IN KEYBOARD/Tastatureingabe)
Adr. F130H  DB 04H

Tabelle 1:
Shift-Ebene 0:

| Zeilen-Nr. | Spalten-Nr. | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | @ | A | B | C | D | E | F | G |
| HEX | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 1 | H | I | J | K | L | M | N | O |
| HEX | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F |
| 2 | P | Q | R | S | T | U | V | W |
| HEX | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |
| 3 | S1 | S2 | S3 | S4 | <- | SP | -> | Enter |
| HEX | | | | | 08 | 20 | 09 | 0D |

Shift-Ebene 1:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | X | Y | Z | [ | \ | ] | ^ | _ |

```
   HEX 58 59 5A 5B 5C 5D 5E 5F

1       0  1  2  3  4  5  6  7
   HEX 30 31 32 33 34 35 36 37

2       8  9  :  ;  <  =  >  ?
   HEX 38 39 3A 3B 3C 3D 3E 3F
------------------------------

Shift-Ebene 2:
        0  1  2  3  4  5  6  7
------------------------------
0       x  y  z  {  |  }  ~  _
   HEX 78 79 7A 7B 7C 7D 7E 7F
1      SP  !  "  #  $  %  &  '
   HEX 20 21 22 23 24 25 26 27
2       (  )  *  +  ,  -  .  /
   HEX 28 29 2A 2B 2C 2D 2E 2F
------------------------------

Shift-Ebene 3:
        0  1  2  3  4  5  6  7
------------------------------
0       '  a  b  c  d  e  f  g
   HEX 60 61 62 63 64 65 66 67

1       h  i  j  k  l  m  n  o
   HEX 68 69 6A 6B 6C 6D 6E 6F

2       p  q  r  s  t  u  v  w
  HEX  70 71 72 73 74 75 76 77
------------------------------
SHift-Ebene 4:
        0  1  2  3  4  5  6  7
------------------------------
0         UA                UG
   HEX 10 11 12 13 14 15 16 17

1              STOP
   HEX 00 01 02 03 04 05 06 07

2     C-L C-R     CLS ENT
   HEX 08 09 0A 0B 0C 0D 0E 0F
------------------------------
```

Mit Shift 4 und der Taste G (17H) wird im INKEY auf Graphik umgeschaltet, d. h. es wird zum ermittelten Hexa-Kode eine 80H aufaddiert. Die Umschaltung in die A-Ebene erfolgt mit Shift 4 und der Taste A (91H).

C-L : Kursor links
C-R : Kursor rechts
CLS : Bildschirm loeschen
ENT : ENTER
STOP : Programmabarbeitungen stoppen

INLIN (Eingabe einer Zeile mit fuehrendem Promptsymbol)
Adr. F2B3H DB 05H
Ausgabe eines Promptsymbols und anschliessende Eingabe einer Zeichenkette bis Enter (0DH). die Startadresse fuer die Eingabezeile wird in SOIL zwischengespeichert und kann nach Rueckkehr fuer die Auswertung, z. B. fuer INHEX, verwendet werden. Da jedes eingegebene Zeichen durch die Folge INCH/OUTCH bis 0DH sofort auf dem Bildschirm ausgegeben wird, erfolgt beim Rollen

am unteren Bildschimrand automatisch eine Korrektur von SOIL
um -20H.

OUTHX (OUT A-Register hexadezimal)
Adr. F301H   DB 06H
Ausgabe des A-Registers hexadezimal. Es werden pro Byte zwei
Zeichen ausgegeben.

OUTHL (OUT A-Register hexadeziamal)
Adr. F31AH   DB 07H
Ausgabe des HL-Registers hexadezimal. Es werden 4 Zeichen aus-
gegeben.

CSAVE (Save to Cassette)
Adr. F369H   DB 08H
Entspricht dem S-Kommando des Monitors, wobei die Anfangsadre-
se und die Endadresse vorher in ARG1 und ARG2 einzutragen sind.

CLOAD (Load from Cassette)
Adr. F3F8H   DB 09H
Die Routine entspricht dem L-Kommando des Monitors. Anfangs-
und Endadresse muessen vorher in ARG1 und ARG2 eingetragen
werden.

MEM (Modify Memory)
Adr. F325H     DB 0AH
Die Routine entspricht dem M-Kommando des Monitors. Vor An-
sprung ueber RST 20H muss die Anfangsadresse in ARG1 einge-
tragen werden.

WIND (Rollfenster fuer Bildschirmbereich)
Adr. F6D1H     DB 0BH
Entspricht dem W-Kommando des Monitors. In ARG1 und ARG2 sind
Anfangsadresse und Endeadresse+1 fuer das Rollfenster einzu-
tragen.

OTHLS (Ausgabe von 2 Byte hexadezimal entspr. der Adresse im
HL-Register)
Adr. F5C7H     DB 0CH
Entsprechend der Adresse im HL-Hegister werden 2 Byte = 4 Zei-
oben (erst High-Teil, dann Low-Teil) und anschliessend ein
Leerzeichen ausgegeben.

OUTDP (Ausgabe eines Doppelpunktes (:) und weiter wie OTHLS)
Adr. F5C4H     DB 0DH
siehe OTHLS

OUTSP (Ausgabe eines Leerzeichens)
Adr. F5CFH     DB 0EH
Ausgabe eines Leerzeiehens

TRANS (Transfer)
Adr. F51DH     DB 0FH
Die Routine entspricht dem T-Kommando des Monitors. In ARG1,
ARG2 und ARG3 sind vorher die Werte fuer "von Adresse", "auf
Adresse" und Byteanzahl einzutragen.

INSTR (Eingabe einer Zeichenkette)
Adr. F2B9H      DB 10H
Es wird die Eingabe einer Zeichenkette abgefordert, die mit
Enter abzuschliessen ist. Wie in INLIN steht in SOIL die An-
fangsadresse der Zeichenkette fuer eine anschliessende Auswer-
tung. Im INSTR wird kein fuehrendes Promptsymbol ausgegeben.

**KILL (Fuellen eines Speicherbereichen mit einem Byte)**
**Adr. F50DH DB 11H**
**Die Routine entspricht dem K-Kommando des Monitors. ARG1, ARG2 und ARG3 sind vorher mit "von Adresse", "bis Adresse" und dem zu fuellenden Byte zu laden.**

**HEXUM (Hexa-Umschaltung)**
**Adr. F6B8H DB 12H**
**Die Routine entspricht dem H-Kommando des Monitors. Umschaltung der Tastatur auf die Zeichen 0..7, 8..? in die Shift-Ebene Null. Diese Umschaltung ist z. B. vor Zifferneingabe sehr sinnvoll.**

**ALFA (Alpha-Umschaltung)**
**Adr. F6C5H DB 13H**
**Umschaltung der Tastencodetabelle auf die Zeichen H...W in der Shift-Ebene Null. Dieser RST 20H entspricht dem A-Kommando des Monitors.**
**Ihre Anwendung in eigenen, selbst gefertigten Programmen erleichtert die Programmierung sehr wesentlich und verkuerzt den sonst erforderlichen Programmumfang. Der Anwender muss nur wissen, wie diese Programmteile aufgerufen werden und auf welche Art die Uebermittlung der Parameter vorgenommen wird.**
**Diese aufgefuehrten Monitorfunktionen koennen ohne Einschraenkungen aus beliebigen Nutzerprogrammen aufgerufen werden.**
**Ihr Aufruf kann erfolgen in der Art: 'CALL adr', wobei als Adresse die Aufrufadresse der Monitorroutine angegeben wird.**
**Alle Routinen werden mit dem Befehl 'RET' beendet, so dass die Programnfortsetzung im aufrufenden Programm gewaerleistet ist.**
**In Anlage 4 sind wichtige Arbeitszellen des Monitors angegeben.**

**Allerdings muss bei einer eventuellen Aenderung des Monitors, die zwangslaeufig zu neuen Adressen der Rufe fuehrt, in allen Nutzerprogrammen, die diese Aufrufe benutzen, diese neuen Adressen in den 'CALL'-Befehlen geaendert werden.**
**Deshalb wurde noch eine andere Art des Aufrufs bereitgestellt, die unabhaengig von Aenderungen im Monitor stets den richtigen Anschluss garantiert Dazu wird anstelle des Aufrufes mit dem 'CALL'-Befehl eine der Restart-Adressen verwendet. Beim Abarbeiten des Restart-Befehls 'RST 20' (E7H) wird ein 'CALL'-Befehl zur Adresse 0020H ausgefuehrt, die Ruecksprungadresse wurde vorher in den Keller gerettet. Auf der Adresse 20H steht ein Sprungbefehl in den Monitor, der beim Programmstart oder nach Reset automatisch dort eingetragen wird.**
**Im Monitor befindet sich dann eine Auswertelogik, die anhand der gekellerten Ruecksprungadresse die konkret geforderte Monitorroutine ermittelt und deren Realisierung einleitet. Die benoetigte Auswahlinformation steht in einem dem RST-Befehl folgenden Byte, die Ruecksprungadresse kann also direkt als Zeiger auf dieses Byte verwendet werden, muss dann aber zur Programmfortsetzung auf den dem Byte folgenden Befehl gestellt werden.**
**Ein einfaches Beispiel soll das verdeutlichen:**
**In einem Programmstueck so zuerst der Bildschirm geloescht und anschliessend eine beliebige Anzahl von Zeichen eingegeben werden. Diese Zeichen sollen sofort wieder auf dem Bildschirm erscheinen, die Betätigung der Entertaste beendet das Programm und kehrt in den Monitor zurueck.**

Zum Vergleich wurde dieses Beispiel einmal unter Verwendung von 'CALL'- und einmal unter Verwendung von 'RST'-Befehlen programmiert.

```
Befehls-        Maschinen-
zahler          code
                ;
                ; BEISPIEL MIT CALL-BEFEHLEN
                ;
                OUTCH: EQU 0F21B  ; AUSGABE ZEICHEN
                INCH:  EQU 0F20C  ; EINGABE ZEICHEN
1000 3E 0C      BSP:   LD  A,0CH  ; LOESCHEN BILD-
                                    SCHIRM
1002 CD 1B F2          CALL OUTCH ; AUSGABE
1005 CD 0C F2   M1:    CALL INCH  ; EINGABE
1008 CD 1B F2          CALL OUTCH ; AUSGABE
100B FE 0D             CMP  ODH   ; ENTER ?
100D C2 05 10          JPNZ M1    ; NEIN
1010 C9                RET        ; JA-->MONITOR

                ;BEISPIEL MIT RST-BEFEHLEN
                ;

                OUTCH:  EQU 0     ; AUSGABE ZEICHEN
                INCH:   EQU 1     ; EINGABE ZEICHEN
1000 3E 0C      BSP2:   LD  A,0CH ; LOESCHEN BILD
                                    SCHIRM
1002 E7                 RST 20H   ; RUF
1003 00                 DB OUTCH  ; AUSGABE
1004 E7         Ml:     RST 20H   ; RUF
1005 01                 DB INCH   ; EINGABE
1006 E7                 RST 20H   ; RUF
1007 00                 DB OUTCH  ; AUSGABE
1008 FE 0D              CMP 0DH   ; ENTER ?
1OOA C2 04 10           JPNZ M1   ; NEIN
1OOD C9                 RET       ; JA -> MONITOR
```

Falls die Funktionen des Monitors durch eigene Programmteile realisiert werden sollen, muss der Sprungbefehl auf der Adresse 20H, der auf drei Auswerteroutinen im Monitor zeigt, durch einen Sprungbefehl in eine eigene Auswerteroutine ersetzt werden. Bleibt die Zuordnung der Auswahlinformationen erhalten, so sind auch in diesem Fall keinerlei Aenderungen in den eigenen Programmen notwendig die den Aufruf ueber den 'RST'-Befehl verwenden.
Damit ist es z. B. moeglich, eigene Ein- and Ausgaberoutinen, mit denen die Bedienung anderer Geraete realisiert wird, zu verwenden und damit die Ein- und Ausgaben ueber andere Geraete zu erreichen.
Abschliessend sei noch bemerkt, dass ein komplettes Assemblerlisting des Monitors mit Hilfe des im Anhang abgedruckten Reassemblers durch Reassemblieren des ROM-Inhaltes erzeugt werden kann.

## 5.1.2. Erweiterungen des Monitors

Falls der Kommandoumfang und die Leistungen des Monitors nicht ausreichen, ist die Nutzung eigener Programmteile zur Erweiterung moeglich. Damit diese auch aus der Kommandoschleife des Monitors nutzbar sind, muessen diese Kommandos mit dem Zeichen '@' eingeleitet werden. Statt des nun ueblichen Leerzeichens steht nun ein ASCII-Zeichen, das das aufzurufende Programm

spezifiziert. Der Anschluss zur Kommandoschleife wird in folgender Weise hergestellt.
Ab der Adresse 00B0H werden die hexadezimale Verschluesselung des ASCII-Zeichens sowie die Anfangsadresse des zugehoerigen Programmteiles eingetragen. Als Beispiel sollen durch die zusaetzlichen Kommandos '@B' der BASIC-Interpreter (siehe Abschnitt 5.2.) ab der Adresse 0100H gestartet werden bzw. durch '@C' ein Wiederstart ab der Adresse 0103H erfolgen.

Dazu wird ab Adresse 0B0H eingetragen:

```
.
00B0   42    Zeichen "B"
00B1   00    niederwertiger Teil der Startadresse
00B2   01    hoeherwertiger Teil der Startadresse
00B3   43    Zeichen "C"
00B4   03    niederwertiger Teil der Startadresse
00B5   01    hoeherwertiger Teil der Startadresse
```

Die Eingabe zusaetzlicher Progrmmteile kann z. B. im Adressbereich zwischen 0100H und 03FFH erfolgen, da die meisten Programme nicht unterhalb von 0400H arbeiten und die Arbeitszellen des Monitors unterhalb von 0100H liegen. Diese zusaetzlichen Programmteile muessen allerdings immer wieder in den Speicher gebracht werden, da sie mit Ausschalten des Gerätes verlorengehen. In dem nachfolgenden Abschnitt wird die Eingabe, Speicherung und Testung selbsterstellter Programme beschrieben.

## 5.2. Hinweise zum Aufbau einer Programmbibliothek

Der Monitor allein als nutzbares Programm wird bald in all seinen Moeglichkeiten ausgeschoepft sein, so dass der Wunsch nach weiteren Programmen geweckt wird.
Da aber nur der Monitor in einem nichtfluechtigen Speicher steht, gehen andere, meist sehr muehsam eingegebene Programmteile nach dem Ausschalten wieder verloren. Aus diesem Grund ist es notwendig, sich eine Programmbibliothek auf einem externen Datentraeger anzulegen, in die jedes neue Programm aufgenommen wird, um es fuer spaetere Nutzung immer wieder zur Verfuegung zu haben. Im Monitor sind zu diesem Zweck die Kommandos "L" (load from cassette) und "S" (Save to cassette) enthalten. Mit diesen Kommandos ist es moeglich, Programme aus dem Speicher auf ein Magnetband, auszulagern und wieder in den Speicher zu laden.
Die Eingabe der Programme erfolgt beim erstenmal meist mittels der Tastatur. Im allgemeinen wird die hexadezimale Zifernfolge, die in einem vorgegebenen Speicherbereich eingetastet werden soll, einer Liste entnommen (z. B. die in diesem Buch enthaltenen Testbeispiele).
Diese Listen entstanden entweder als Resultat eines Uebersetzungslaufes auf einer EDV-Anlage oder wurden durch manuelle Uebersetzung selbst gefertigt. In jedem Fall wird es sich um ein in sich abgeschlossenes Programmteil handeln. Eine Programmaenderung oder -erweiterung waehrend der Eingabe wird kaum zu brauchbaren Ergebnissen fuehren.
Einige Programmteile, die als Liste geliefert werden, sind nur ein Speicherabzug in der Art, wie er auch mit den D-Kommando auf dem Bildschirm zu sehen ist.
Damit koennen recht einfach Liste und tatsaechlicher Speicher inhalt miteinander verglichen werden. Zur Eingabe wird das M-Kommando des Monitors verwendet.
Mit dem Parameter wird die Anfangsadresse des Programmes angegeben. Es ist zweckmaessig, vorher die Tastatur mit dem H-Kommando in die hexadezimale Eingabe zu schalten. Die Zif-

fern sind jetzt ohne Benutzung der Shift-Taste direkt erreichbar. Anschliessend kann ein grosser Bereich der Liste eingegeben werden.

Es empfielt sich, nach einer bestimmten Anzahl von hexadezimalen Ziffern die Eingabe mit dem Semikolon ";" zu beenden und mit dem Kommando 'D :' den bis dahin eingetasteten Abschnitt zu kontrollieren. Die zeilenweise mitausgegebenen Pruefsummen (CS) koennen mit dem Listenausdruck verglichen werden. Die weiteren Bloecke werden dann wieder mit dem M-Kommando ab der Abbruchadresse eingegeben.
Nach endlicher Zeit steht das Programm im gewuenschten Speicherbereich und entspricht dem Original auf der Liste. Es waere nun grundfalsch, sofort mit der Abarbeitung zu beginnen, vielmehr sollte das Programm zuerst einmal auf Magnetband abgespeichert und somit gesichert werden. Das erspart wiederholtes Eingeben von Programmteilen ueber Tastatur und erleichtert die Arbeit wesentlich.
Meist wird man diese Abspeicherung nicht gleich auf dem Magnetband vornehmen, auf dem sich die anzulegende Programmbibliothek befindet, sondern definiert ein anderes als "Arbeitsband".
Auf jeden Fall sollte man sich sehr genau die Bandstellen zu Beginn und Ende sowie die im S-Kommando verwendeten Parameter notieren. Das erleichtert ein spaeteres Wiederauffinden der gespeicherten Informationen und gewaehrleistet das ordnungegemaesse Einlesen in den Speicher.
Moechte man das wiederholte Einlesen vom Band, das ja auch einen gewissen Aufwand erfordert, so gering wie moeglich halten und ist noch genuegend Speicherplatz frei, empfielt sich auch folgende Methode:
Mit dem T-Kommando (Transfer) wird eine gleichartige Kopie des Programms in einem anderen, nicht genutzten Speicherbereich erzeugt. Falls jetzt das Programm beim Testen zerstoert, wird ist es moeglich, wieder mit dem T-Kommanda die Kopie auf den urspruenglichen Speicherbereich zurueckzuspeichern und damit das Programm wieder herzustellen.
Im allgemeinen werden Programme nach dem Eingeben nicht sofort richtig und fehlerfrei arbeiten, dazu sind zu viele Fehlermoeglichkeiten vorhanden. Sie reichen von einfachen Eingabefehlern ueber falsche Anschlussstellen bishin zu Fehlern im Programmentwurf (insbesondere bei manuell assemblierten Programmen). Damit ergibt sich die Notwendigkeit, diese Programme auszutesten. Der Monitor stellt dazu einige Hilfen zur Verfuegung, von denen Haltepunkt und Schrittbetrieb bereits erwaehnt wurden.
Zum Testen von Programmen unter den fuer diese Verhaeltnisse typischen Bedingungen ergibt sich etwa nachfolgende Herangehensweise, die mit fortschreitendem Erfahrungsstand individuell modifiziert werden kann:

- Zu Beginn der Programmtestung wird mit dem I-Kommando der Registerrettebereich geloescht und damit ein definiertes Laden der CPU-Register gewaehrleistet.
- Der Haltepunkt wird mit dem B-Kommando auf den ersten zu testenden Befehl des Programmes gesetzt und mit dem E-Kommande gestartet.
- Mit Erreichen des Haltepunktes wird mit dem N-Kommando eine schrittweise Abarbeitung vorgenommen. Gegebenenfalls ist die Ausgabe der Registerinhalte auf Ausfuehrung der Befehle zu kontrollieren.
- Auftretende Programmschleifen sowie haeufig aufgerufene Routinen (Unterprogramme) sind mindestens einmal im Schrittbetrieb abzuarbeiten. Sofern man danach von der

Richtigkeit dieser Programmteile ueberzeugt ist, wird bei wiederholtem Aufruf die Haltepunktadresse auf den dem Aufruf oder dem Schleifenausgang folgenden Befehl gelegt und damit diese Programmteile direkt abgearbeitet.

- Alle im schrittbetrieb abgearbeiteten Befehle werden bei richtiger Funktion in der Liste abgehakt, damit ist jederzeit ein Ueberblick ueber das zu testende Programm vorhanden.
- Falls die Testung des Programmes nicht auf einmal durchgaengig erfolgen kann, sei es durch zu grossen Programmumfang oder durch Programmabsturz (undefiniertes Verhalten), kann mittels Aktivieren des letzten erfolgreich erreichten Haltepunktes oder der in Schrittbetrieb erreichten Adresse das zu testende Programm jederzeit wieder neu gestartet werden. Der weitere Ablauf ist dann genauso wie bereits beschrieben.
- Falls durch den Monitor nicht bereits die Protokollierung der Monitorbefehle bei Monitorrufen verhindert wird, sollte durch Setzen der Haltpunktadresse auf den dem Monitorruf folgenden Befehl diese Protokollierung ausgeschlossen worden.

Bei Beachtung dieser Hinweise ist die Austestung aller Programme moeglich. Sollten waehrend der Testung in Programm Fehler festgestellt werden, die eine Vernaenderung der Programmstruktur erforderlich machen, ist das bei den im Maschinenkode vorliegenden Programmen besonders schwierig. Entweder man bemueht sich um eine Aenderung mit anschliessender Neuuebersetzung oder es wird an der zu aendernden Stllen eine Maschinenkodeeinfuegung vorgenommen.
Dazu wird an Stelle von drei oder mehr Befehlsbyte ein 'CALL'-Befehl eingefuegt, dessen Adresse auf das erste freie Byte am Programmende oder in einen anderen freien Speicherbereich zeigt. Die dabei verdraengten Befehlsbyte muessen vollstaendige Befehle sein, erforderlichenfalls sind mehr als drei Byte zu ersetzen und die nicht benoetigten durch 'NOP'-Befehle aufzufuellen. Keinesfalls darf ein Relativsprungbefehl ersetzt werden, da dessen Sprungdistanz von seiner Position im Programm abhaengig ist.
An der durch den 'CALL'-Befehl adressierten Stelle stehen als erstes die durch die Einfuegung verdraengten Befehlsbyte sowie die geplante Erweiterung. Mit einem 'RET'-Befehl wird die Abarbeitung wieder an der urspruenglichen Stelle im Programm fortgesetzt.
Diese Art der Maschinenkodeerweiterung ist zu Testzwecken durchaus moeglich, sollte aber zu einem spaeteren Zeitpunkt durch Aenderung im Programm und Neuuebersetzung Beruecksichtigung finden.
Wenn die Programme vollstaendig ausgetestet sind, werden sie in die Programmbibliothek aufgenommen. Auf diesem Magnetband befinden sich eine Reihe von bereits ausgetesteten Programmen, die bei Bedarf in dcn Speicher geladen werden koennen. Allergroessten Wert ist auf ein genaues Inhaltsverzeichnis zu legen, damit diese Programmteile wiedergefunden und spaeter problemlos geladen werden koennen.
Dieses Laden erfolgt mit dem L-Kommando des Monitors, nachdem zuvor die benoetigten Parameter dem Inhaltsverzeichnis entnommen wurden. Von besonders wichtigen Programmteilen sollte man sich noch eine weitere Kopie anlegen, um bei einer eventuellen Zerstoerung der Programmbibliothek diese jederzeit wieder herstellen zu koennen.

## 5.3. BASIC

### 5.3.1. Programmiersprache BASIC

BASIC ist eine sogenannte hoehere Programmiersprache, eine Sprache also, die sich nicht direkt auf die Maschinensprache des Rechners bezieht. Diese Sprache wurde um 1965 von John G. Kemeny und Thomas E. Kurtz im Dartmouth College in den USA entwickelt. Sie hatten dabei die Entwicklung einer Computersprache vor Augen, die einerseits vom Anfaenger leicht zu erlernen ist und andererseits viele Moeglichkeiten bietet. Man sollte mit dier Sprache leicht numerische (Zahlen-) Probleme angehen koennen, aber ebenso sollten Verwaltungsaufgaben, bei denen die Textverarbeitung eine grosse Rolle spielt, in der Sprache zu behandeln sein. Deshalb ersann man fuer diese Sprache die Bezeichnung BASIC, ein Wort, das konstruiert wurde aus den Anfangsbuchstaben von: Beginner's All purpose Symboic Instruction Code (etwa: Anfaenger-Allzweck-Symbolik-Instruktions-Code).
Neben den Woertern "Beginner's" und "All purpose" (Allzweck), deren Bedeutung ohne weiterse einleuchtet, koennen die Woerter "Symbolic Instruction Code" vielleicht ein wenig verwirren. Damit wird nur gesagt, dass man mit einem sogenannten symbolischen Befehlekode arbeitet. Das sind Schluesselwoerter, mit denen bestimmte Verarbeitungsbefehle mit Hilfe von Symbolen angegeben werden, z. B. das Addieren oder Subtrahieren. Also nur wenige Woerter, deren Bedeutung bei der weiteren Beschaeftigung mit der Sprache ohne weiteres einleuchten wird.
Kemeny und Kurtz haben mit ihrem einfachen Aufbau genau ins Schwarze getroffen. Die Erwartungen haben sich erfuellt: BASIC wurde eine sehr populaere Programmiersprache. Sie hat sich im Unterricht bewaehrt, und auch im Laboratorien und in Betrieben gibt es zahllose Computer, die den Befehlen in BASIC gehorchen.

### 5.3.2. Der BASIC-Interpreter

Die eingegebenen Programmzeilen muessen ihrem Inhalt entsprechend bestimmte Verarbeitungsleistungen aufrufen. Diese Aufgabe uebernimmt der BASIC-Interpreter. Dieser Interpreter arbeitet Anweisung fuer Anweisung interpretativ ab, d. h. jedem entschluesselten Befehl oder Kommando wird ein entsprechendes Maschinenprogramm zugeordnet und mit den in der BASIC-Anweisung angegebenen Zahlenwerten abgearbeitet. Diese interpretative Abarbeitung nutzt ein Maschinenprogramm fuer alle im gesamten Programm vorkommenden gleichen Kommandos bzw. Befehle. Damit sind genau so viele verschiedene Maschinenprogramme notwendig, wie es verschiedene Kommandos und Befehle gibt. Damit ist die Laenge des BASIC-Interpreter festgelegt, unabhaengig von der Laenge der abzuarbeitenden Programme. Als Nachteil ist die geringere Rechengeschwindigkeit gegenueber uebersetzten Programmen anzusehen.
Ein Vorteil ist die einfache Dialogfaehigkiet, BASIC-Zeilen erscheinen so wieder auf dem Bildschirm, wie sie eingegeben wurden. Ausserdem koennen einzelne Anweisungen sofort ausgefuehrt werden. Auf diese Weise kann der Computer an Stelle eines Tischrechners verwendet werden. Auf diese Betriebsart wird noch genauer eingegangen.
An dieser Stelle sei auch auf einen Nachteil des Interpreters hingewiesen. Wie spaetere Beispiele zeigen werden, kann ein Teil des BASIC-Programmes haeufiger als nur einmal ausgefuehrt

werden, z. B. bei der Unterprogrammarbeit oder in Programmschleifen. Vom Interpreter wird aber jedesmal, wenn der entsprechende Abschnitt an der Reihe ist, Anweisung um Anweisung neu interpretiert. Das hat zur Folge, dass solche Programme laengere Verarbeitungezeiten gegenueber gleichen Programmloesungen in Maschinensprache erfordern. Im Extremfall kann das dazu fuehren, dass besonders zeitkritische Probleme nur in Maschinensprache loesbar sein werden. Diese Maschinenprogramme koennen dann innerhalb einer BASIC-Anweisung aufgerufen werden.

### 5.3.3. Laden des BASIC-Interpreters

Da der BASIC-Interpreter selbst ein Maschinenprogranun darstellt, benoetigt er im Speicher des Z1013 einen Speicherplatz von ca. 2.75 KByte. Dazu wird noch weiterer Speicherraum zum Ablegen der BASIC-Programme gebraucht.
Im Anhang ist eine Liste des Maschinenkodes des BASIC-Interpreter-Programmes enthalten. Diese Liste von Maschinenbefehlen muss beim ersten Mal per Hand in den Rechner eingetippt werden, wobei bei Adresse 0100H zu beginnen ist. Diese muehsame Arbeit ist aber nur beim erstem Mal notwendig. Deshalb soll an dieser Stelle darauf verwiesen werden, dass dabei entsprechende Sorgfalt von Noeten ist. Ein einziger Fehler kann das Interpreterprogramm funktionsunfaehig machen.
Das Eintippen erfolgt mit dem in 1.3. kennengelernten M-Kom mando. Es wird immer ein kleiner Abschnitt eingegeben und an schliessend mit dem D-Kommando wieder angezeigt. Durch Vergleich mit den Pruefsummen je Zeile auf dem Bildschirm und in der Liste wird die Fehlerfreiheit festgestellt. Muss die Ar beit unterbrochen werden oder ist der Interpreter vollstaendig eingegeben, kann er wie jedes andere Maschinenkode-Programm auf Magnetband gespeichert werden und steht ab jetzt immer zur Verfuegung. Soll jetzt mit dem MRB Z1013 ein BASIC-Programm bearbeitet werden, so muss vorher nur noch der BASIC-Interpreter von Magnetband in den Adressbereich 0100H bis 0BFFH geladen und mit dem J-Kommando ab Adresse 0100H gestartet werden.

### 5.3.4. Arbeit mit dem BASIC-Interpreter

Nach dem Start des BASIC-Interpreters erscheint auf dem vorher geloeschten Bildschirm die Meldung 'robotron Z 1013 BASIC 3.01', in der naechsten Zeile 'READY' und am Beginn der folgenden Zeile das Zeichen '>' (groesser als) als Promptsymbol. Immer wenn das Zeichen '>' auf dem Bildschirm auftaucht, befindet sich der Interpreter in einer Eingabeschleife, d. h., dass der Interpreter zur Eingabe von Befehlen, Kommandos oder Programmzeilen bereit ist. Eine solche Programmzeile hat folgenden Aufbau: [Zlnr] anweisung 1 [; anweisung 2, ... ]. Die eckige Klammer bedeutet, dass diese Eingaben nicht unbedingt getaetigt werden muessen.
Die Laenge einer Programmzeile darf 64 Zeichen nicht ueberschreiten. Jede Programmzeile und jede Kommandoeingabe ist mit der Enter-Taste abzuschliessen
Beginnt die eingegebende Zeile mit einer Zeilennummer (Zlnr), so wird diese Zeile als Programmzeile interpretiert und abgespeichert. Diese Zeilennummern sind ganze Zahlen im Bereich zwischen 1 und 32767. Bei der Numerierung der Programmzeilen geht man sinnvollerweise in Zehnerschritten vor. Dadurch ergibt sich die Moeglichkeit, noch genuegend Einfuegungen in be-

reits bestehende Programme vorzunehmen.

Die Abarbeitung des BASIC-Programmes erfolgt in der Reihenfolge der Zeilennummern.

Alle anderen Eingaben ohne Zeilennunmmern werden als Befehl zur sofortigen Ausfuehrung angesehen. Sind sie zulaessig werden sie ausgefuehrt, sonst erfolgt eine Fehlermeldung. Danach kann die naechste Eingabe erfolgen.
Die Arbeit ohne Zeilennummer nennt man auch Tischrechnermodus.

```
>A=66-20; PRINT A
   46
READY
```

Innerhalb einer einzugebenden Zeile kann beliebig oft mit den Kursortasten "Kursor links '<-'" oder "Kursor rechts '->'" korrigiert werden. Der Kursor wird unter das fehlerhafte Zeichen bewegt und durch Eingabe des richtigen Zeichens der Fehler behoben. Danach muss der Kursor wieder an das Zeilenende gebracht werden (auf die erste freie Zeichenstelle) und die Zeile kann mit der Enter-Taste abgeschlossen werden.
Soll eine bereits im Programmspeicher abgespeicherte Programmzeile geaendert werden, wird diese einfach neu eingegeben (mit der gleichen Zeilennummer). Soll eine Zeile geloescht werden, so muss nur ihre Zeilennummer angegeben werden.
Alle Befehle und Kommandos koennen mit einem Punkt nach dem ersten oder nach weiteren Buchstaben abgekuerzt werden. In der im Anhang befindlichen Befehlsliste sind die moeglichen Abkuerzungen zu finden. Es sind auch kuerzere Varianten der Befehle moeglich, aber dadurch kommt es im Interpreter unter Umstaenden zu Verwechslungen mit anderen Befehlen und zu fehlerhaften Abarbeitungen.
Auch Leerzeichen zwischen den einzelnen Elementen der Programmzeilen koennen weggelassen werden. Diese beiden Moeglichkeiten der Programmverkuerzung erlauben es, den Programmspeicher besser auszunutzen, um mehr Programmzeilen unterzubringen. Dadurch geht aber die Uebersichtlichkeit der Programme verloren.
Es duerfen mehrere Anweisungen in einer Programmzeile untergebracht werden. Diese Anweisungen sind durch ein Semikolon voneinander zu trennen.
Der BASIC-Interpreter realisiert eine einfache Ganzzahlarithmetik in den vier Grundrechenarten Addition, Subtraktion, Multiplikation und Division (+ - * /) im Zahlenbereich von -32768 bis +32767. Aus diesem Grund besitzt er nur einen Datentyp. Zu beachten ist, dass auch die Division nur ganzzahlig ausgefuehrt wird, d. h., dass z. B. 9/4=2 ist. Der Teil des Resultates hinter dem Komma wird einfach weggelassen.
Den meisten der Schluesselworte folgt ein weiterer Ausdruck. Das koennen Zahlen, Variable oder arithmetische Konstruktionen mit diesen sein. Erforderlichenfalls sind derartige Konstruktionen mit Klammern aufzubauen, um diese Konstruktionen mathematisch eindeutig zu machen. Der Interpreter arbeitet nach der bekannten Rechenregel: Punktrechnung geht vor Strichrechnung. Mehrere Klammern koennen dabei beliebig geschachtelt werden. Innerhalb eines Befehls koennen weitere Befehle oder Funktionen verwendet werden.
Als Variable sind alle Buchstaben des Alphabets von A bis Z erlaubt. Eine Variable darf aber nur aus einem einzelnen Buchstaben bestehen. Es koennen nur Grossbuchstaben verwendet werden.

Mit dem Symbol '@' ist die Nutzung eines eindimensionalen Feldes (Vektor) moeglich. Die Teilanweisung @(A) stellt dabei das A-te Element des Feldes dar. Anstelle von A kann sowohl ein anderer Buchstabe des Alphabetes als auch eine Zahl oder eine arithmetische Konstruktion stehen, d. h. ein Ausdruck wie oben bereits beschrieben.
Werden bei der Arbeit mit dem BASIC-Interpreter Syntaxfehler gemacht, so werden diese Fehler erkannt und angezeigt. Logische Fehler im Programm kann der Interpreter nicht finden, das bleibt dem Geschick des Programmierers ueberlassen. Der BASIC-Interpreter kennt drei verschiedene Fehlermeldungen:

WHAT? - Das Schluesselwort bzw. der Ausdruck sind nicht erlaubt bzw. fehlerhaft, d. h. der Interpreter versteht die Anweisung nicht.

HOW? - Die Ausfuehrung der Anweisung ist im Rahmen der Moeglichkeiten dieses Interpreters nicht moeglich (z. B. bei einer Zahlenbereichsueberschreitung).

SORRY - Die Ausfuehrung der Anweisung ist zwar moeglich, aber nicht unter den aktuellen Voraussetzungen( z. B. der Programmspeicher ist erschoepft).

Tritt eine Fehlermeldung bei der Abarbeitung eines Programmes auf, so wird zur Fehlermeldung auch die Zeile ausgegeben, in der der Fehler aufgetreten ist. An der fehlerhaften Stelle wird vom BASIC-Interpreter ein Fragezeichen eingefuegt. Das erleichtert die Fehlersuche.

## 5.3.5. Kommandos des BASIC-Interpreters

Im nachfolgenden sind alle Befehle und Kommandos, die der BASIC-Interpreter verstehen und ausfuehren kann, aufgefuehrt und erlaeutert. Zusammen mit dem im vorhergehenden Abschnitt gesagten ergeben sich daraus alle Moeglichkeiten der im MRB Z 1013 realisierten Programmiersprache BASIC. Die im Anhang befindliche Befehlsliste beinhaltet auch die moeglichen Kurzformen.

**LIST**

Dieses Kommando bewirkt das Auflisten des im Speicher stehen den BASIC-Programmes auf dem Bildschirm. Dadurch lassen sich Programme leicht kontrollieren.

```
>LIST     Auflisten des gesamten Programmes in aufsteigender
          Reihenfolge der Zei~ennnunrnern.
>LIST 10  Auflisten des BASIC-Programmes ab der Zeile 10
          Es werden genau 20 Zeilen aufgelistet.
```

**RUN**

Mit dem Kommando 'RUN' wird ein BASIC-Programm gestartet. Bei der Abarbeitung wird mit der niedrigsten Zeilennummer begonnen. Ist die letzte Zeile abgearbeitet oder eine 'STOP'- Anweisung erreicht, kehrt der Interpreter in die Eingabeschleife zurueck und meldet sich mit 'READY' und auf der naechsten Zeile mit den Aufforderungszeichen '>'. Er erwartet jetzt weitere Kommando- oder Befehlseingaben.

**NEW**

Das im Speicher vorhandene BASIC-Programm wird scheinbar geloescht. Tatsaechlich ist es noch im Programmspeicher enthalten, wird aber bei Neueingabe eines Programmes ueberschrieben.

**BYE**
**Mit 'BYE' wird die Arbeit mit dem BASIC-Interpreter beendet und ins Monitorprogramm zurueckgekehrt. Das zuletzt eingegebene Programm befindet sich noch im Speicher. Wird an dessen Inhalt nichts geaendert, kann mit dem Monitorkommande 'J 0103' wieder der BASIC-Interpreter aktiviert werden (Restart). Jetzt erscheint nur ein 'READY' und in der naechsten Zeile das Aufforderungszeichen '>' auf dem Bildschirm. Die Arbeit mit dem BASIC-Interpreter kann fortgesetzt und das vorher eingegebene Programm wieder gestartet werden.**

**END**
**Dieser Befehl wird zum Vergroessern des vom Interpreter genutzten Programmspeichers verwendet. Mit 'END' kann also das Programmspeicherende neu gesetzt werden, z. B. bei Speichererweiterung. Laesst der augenblickliche Ausstattungsgrad des MRB Z1013 diesen Speicherbedarf nicht zu, weil er real nicht vorhanden ist, wird die Fehlermeldung 'SORRY' ausgegeben. Dabei ist zu beachten, dass ein Bereich von 140 Byte hinter den Programmspeicher freibleibt, der vom BASIC-Interpreter intern verwaltet wird.**

**Beispiel:**
```
>END HEX(3FFF)-140
```

**Die oben aufgefuehrten Kommandos duerfen in keinen Programm auftauchen, sie dienen nur zur Arbeit mit dem Interpreter und werden prinzipiell sofort ausgefuehrt.**

**CSAVE**
**Mit dem Kommando CSAVE "name" wird ein BASIC-Programm unter dem angegebenen Namen auf Magnetband abgespeichert. Dabei kann der Name maximal 16 Zeichen umfassen und muss von Anfuehrungezeichen eingeschlossen sein.**

**CLOAD**
**Mit diesem Kommando wird ein durch CSAVE abgespeichertes BASIC-Programm wieder von Magnetband eingegeben. Zur Kontrolle wird der im CSAVE-Kommande verwendete Name auf dem Bildschirm ausgegeben.**

## 5.3.6. Programmierbare Befehle bzw. Anweisungen

**LET**
**Definiert den Anfang einer Ergibtanweisung, d. h. einer Wertzuweisung. 'LET' muss nicht vorangestellt werden, es dient lediglich der besseren Uebersichtlichkeit.**
**Die im folgenden angegebenen Beispiele stellen nicht in jedem Fall Programme dar, sondern koennen auch nur Varianten eines Anweisungstypes verdeutlichen.**
```
 >10  LET A=1
 >20  A=50;B=30
 >30  LET C=A-B+3
 >40  LET X=3+A+(B-3)/C
 >50  LET @(3)=24
```

**IF**

**Mit der 'IF'-Anweisung werden Bedingungen fuer die Ausfuehrung einer der Anweisung folgenden Anweisung festgelegt. Im Zusammenhang mit der GOTO-Anweisung lassen sich damit Programmverzweigungen realisieren.**

```
 >100 IF B=10 GOTO 200
 >108 IF C=0 PRINT 'FERTIG'
 >115 IF A+B<100 A=A+1;GOTO 100
```

**Die 'IF'-Anweisung selbst darf keine Folgeanweisung in einer Programmzeile sein, muss also immer am Zeilenanfang stehen. Die 'IF'-Anweisung ist damit einer Vergleichsoperation gleichzusetzen.**
**Nachfolgend sind alle erlaubten Vergleichsoperatoren aufgefuehrt:**

```
  >= groesser gleich
  #  ungleich
  >  groesser
  =  gleich
  <  kleiner
  <= kleiner gleich
```

**Ist die in der IF-Anweisung angegebene Vergleichsoperation wahr, wird die in der gleichen Programmzeile folgende Anweisung ausgefuehrt, sonst die nachfolgende Programmzeile abgearbeitet.**

**GOTO**

**Unbedingter Sprung zu einer Zeile, deren Zeilennummer direkt angegeben wird, oder sich aus den angegebenen Ausdruck berechnet. 'GOTO zlnr' als Direktanweisung (ohne vorangestellte Programmzeilennummer) startet das Programm ab der angegebenen Zeilennummer. In Verbindung mit 'IF' kann 'GOTO' zur Konstruktion von bedingten Sprunganweisungen verwendet werden (siehe auch Beispiele der 'IF'-Anweisung).**

```
 >100 GOTO 120
 >GOTO 120
 >110 GOTO 120+B
 >120 GOTO A
 >120 IF A>0 GOTO 100
```

**FOR...TO...STEP...NEXT**

**Damit lassen sich leicht Programmschleifen aufbauen, die mit einer freibestimmbaren Anzahl von Schleifendurchlaeufen abgearbeitet werden sollen. Jede mit 'FOR' eroeffnete Schleife muss mit einer NEXT-Anweisung, die die gleiche Zaehlvariable wie die FOR-Anweisung beinhaltet, abgeschlossen werden.**

```
 >100 N=10
 >110 FOR I=0 to N
 >120 LET a=I*10;b=I*I
 >121 PRINT A,B, A*B
 >150 NEXT I
 >160 ...
```

**Der Programmabschnitt von Zeile 110 bis Zeile 150 wird N-mal durchlaufen, wobei '1' mit dem Wert '0' beginnend in jedem Durchlauf um '1' erhoeht wird, bis der Wert N ueberschrittcn wurde. Danach wird die Schleife verlassen-**
**Eine Schrittweite wird mit dem Schluesselwort STEP gekennzeichnet. Wird STEP weggelassen, wird der Standardwert 1 genommen.**

In der 'FOR..NEXT'-Anweisung koennen die Anfangs- und Endwerte sowie die Schrittweite der Schleifendurchlaeufe fuer die Zaehlvariable (im Beispiel das 'I') beliebige arithmetische Konstruktionen bzw. Ausdruecke sein. Bei jeden Schleifendurchlauf wird die Zaehlvariable um den Wert der Schrittweite veraendert, bis der Endwert ueberschritten wird. Bei negativer Schrittweite ist auf die richtige Angabe der Anfangs- und Endwerte zu achten.

```
 >110 FOR X=A TO N+B STEP C
 >120 ...
 ...
 >150 NEXT X
```

Eine 'FOR-NEXT'-Schleife kann zu jedem beliebigen Zeitpunkt durch eine 'IF..GOTO'-Anweisung verlassen werden. Es darf aber nicht in eine 'FOR...NEXT'-Schleife von ausserhalb der Schleife hineingesprungen werden.

GOSUB...RETURN

Mit GOSUB erfolgt der Aufruf eines in BASIC geschriebenen Unterprogramms, welches mit 'RETURN' beendet werden muss. Nach dem Befehl 'RETURN' wird mit dem, dem Unterprogrammruf folgenden Befehl im BASIC-Programm fortgesetzt. Unterprogramme sind dort sinnvoll, wo gleiche Programmteile an mehreren Stellen benoetigt werden. Dadurch wird Speicherplatz eingespart. (aehnliche Problematik der MC-Programmierung, siehe Abschnitt 4.) Innerhalb eines Unterprogrammes sind die Variablen des rufenden Programmes ebenfalls gueltig und werden zur Parameter- und Ergebnisuebermittlung genutzt.

```
 >120 C=25;GOSUB 180;PRINT 'ZEIT',
 >125 C=60;GOSUB 180;PRINT 'SCHLEIFE'
 ...
 >170 STOP
 >180 REM ZEITPROGRAMM
 >190 IF C#0 C=C-1; GOTO 190
 >210 RETURN
```

REM

Dadurch werden in einen BASIC-Programm; Kommentarzeilen gekennzeichnet. Sie dienen der besseren Uebersichtlichkeit der Programme und werden bei der Abarbeitung durch den Interpreter ueberlesen. Die Programmzeile belegt aber entsprechend ihrer Laenge Speicherplatz in RAM-Bereich des Rechners.

```
 >110 REM LET C=1024
 >180 REM ZEITPROGRAMM
```

Diese Zeilen werden nicht mit abgearbeitet.

INPUT

Dadurch wird die Eingabe von numerischen Werten mittels der Tastatur ermoeglicht. Ein eingegebener Wert wird dabei einer Variablen zugeordnet. Alle eingegebenen Zeichen werden wieder auf dem Bildschirm ausgegeben. Korrekturen der Eingabe sind mit der Taste 'Kursor links' bzw. 'Kursor rechts' moeglich. Die gesamte Eingabe ist mit der Enter-Taste abzuschliessen (nach Korrekturen muss der Kursor auf die erste freie Position in der Eingabezeile gestellt werden!).
Nach der 'INPUT'-Anweisung kann ein in Hochkomma eingeschlossener Text angegeben werden, welcher bei der Ausfuehrung der Anweisung auf dem Bildschirm mit ausgegeben wird. Mittels einer 'INPUT'-Anweisung koennen mehrere Eingaben ausgefuehrt werden. Anstelle einer Zahl kann auch ein Ausdruck eingegeben werden.

```
 >10 INPUT X
 >20 INPUT 'SPRUNGWEITE' S
 >30 INPUT 'WERTEPAAR A'A,B
 >RUN
 X: eingabe
 SPRUNGWEITE: eingabe
 WERTEPAAR A: eingabe  B: eingabe
```

Das Wort 'eingabe' erscheint nicht auf dem Bildschirm, es wurde hier nur verwendet, um deutlich zu machen, dass an dieser Stelle eine Eingabe mittels der Tastatur erfolgt.

```
 >20  INPUT ' 'S
 >RUN
 : eingabe
```

**PRINT**

Damit wird die Ausgabe von numerischen Werten und von Textketten ermoeglicht. Texte sind dabei in Hochkomma einzuschliessen. Mehrere Ausgabeparaneter innerhalb einer 'PRINT'-Anweisung sind mit Komma voneinander zu trennen. Zahlen werden bei fehlender Formatangabe sechsstellig mit unterdrueckten Vornullen rechtsbuendig ausgegeben (d. h.: eine einstellige Ziffer beansprucht sechs Zeichenpositionen auf den Bildschirm, wobei die ersten 5 leer bleiben und die auszugebende Ziffer die sechste Position einnimmt).
Durch ein Doppelkreuz, gefolgt von einer Zahl 1...6), kann diese Formatisierung geaendert werden. Die Zahl gibt die maximal auszugebende Stellenzahl an. Die Formatisierung bleibt bis zur naschsten 'PRINT'-Ausgabe bestehen. Wird die 'PRINT' Anweisung mit einem Komma beendet, so beginnt die Ausgabe der naechsten 'PRINT'-Anweisung in der gleichen Zeile nach der vorangegangenen Ausgabe.

```
 >10  X=5;Y=50;Z=500
 >20  PRINT 'ZAHL X=',X
 >30  PRINT 'ZAHL X=',#2,X
 >40  PRINT X,Y,
 >50  PRINT Z
 >RUN
 ZAHL X=5
 ZAHL X=5
      5  50  500
```

**STOP**

Diese Anweisung beendet die Abarbeitung des BASIC-Programmes. Der Interpreter kehrt in die Eingabeschleife zurueck. Eine 'STOP'-Anweisung muss nicht unbedingt als letzte Anweisung eines BASIC-Programmes stehen. Gegebenenfalls kann diese Anweisung bei der Abarbeitung durch bedingte Spruenge ('IF...GOTO') oder Unterprogrammrufe uebersprungen werden, ehe sie dann im weiteren Programmlauf erreicht wird.

**CALL**

Mit 'CALL' und einer nachfolgenden, als Hexadezimalausdruck gekennzeichneten MC-Adresse wird ein in Maschinensprache geschriebenes Unterprogramm vom BASIC-Interpreter aus gestartet. Soll nach Abarbeitung des MC-Unterprogrammes die Arbeit des Interpreters mit der folgenden BASIC-Anweisung fortgesetzt werden, so ist das MC-Unterprogramm mit einen 'RETURN'-Befehl (bedingt oder unbedingt, z. b. 0C9H, siehe auch Abschnitt 4, Unterprogrammtechnik) abzuschliessen. Zur Uebermittlung der Parameter werden die PEEK- und POKE-Befehle verwendet.

**Beispiel:**

```
 >200 CALL (HEX (3000))
 ...
 MC-Unterprogramne:
 3000 3A FF 31  LD A, (31FFH)
 3003 3C        INC  A
 3004 27        DAA
 3005 32 FF 31  LD (31FFH), A
 3008 C9
```

Das durch die CALL-Anweisung in Zeile 200 aufgerufene MC-Unterprogramm (von Adresse 3000H bis 3009H) zaehlt den Inhalt des Speicherplatzes 31FFH dezimal um '1' weiter. Auf diesem Speicherplatz koennte dann mit einer PEEK-Anweisung zugegriffen werden.
Mit diesem Befehl koennen also auch die Routinen des Monitors aufgerufen werden. Das wird z. B. im Anwendungsprogramm 'Telefonverzeichnis' beim Retten von Dateien praktiziert.

**PEEK**

Die PEEK-Anweisung ermoeglicht den direkten Speicherzugriff zum RAM bzw. ROM des Rechners. Die Anweisung enthaelt eine Adresse als Parameter. Vom Speicherplatz, der durch die Adresse ausgewaehlt wurde, wird ein Byte als Wert einer Variablen dezimal zugewiesen.

```
 >10  X=PoeK (1023)
 Speicherplatz 1023 wird adressiert (dezimale Adresse)
 >20  X=PoeK (HEX (3FF))
 Speicherplatz 1023 wird adressiert (hexadezimale Adresse)
```

**POKE**

Mit Hilfe von POKE kann ein Speicherplatz beschrieben werden (wobei der Speicherplatz im RAM-Bereich liegen muss). Der erste Parameter bestimmt die Adresse des Speicherplatzes, der zweite gibt den Datenwert an, der abgespeichert werden soll.

```
 >10  POKE HEX (ED08),65
```

Der Wert 65 (dezimal) wird in den Speicherplatz mit der Adrtsse ED08H (BWS) als 41H abgespeichert. 41H entspricht dem ASCII-Kode fuer den Grossbuchstaben A.

```
 >10  FOR I=0 TO HEX (3FF)
 >20  POKE HEX (3000) + I, PEEK (HEX (F000) + I) ; NEXT I
 ODER
 >10  A=HEX (3000); B=HEX (F000)
 >20  FOR I=0 TO 1023; POKE A + I, PEEK (B + I); NEXT I
```

Es wird der Monitor ab der Adresse F000 in der Laenge von 1K Byte nach Adresse 3000H umgespeichert.
Dieses Programm entspricht dem Monitorkommande: T F000 3000 3FF. Man beachte die unterschiedlichen Ausfuehrungszeiten.

**BYTE**

Damit wird der Wert des nachfolgenden Ausdrucks als Hexadezimalausdruck auf dem Bildschirm ausgegeben. Es erfolgt nur die Ausgabe von dezimalen Werten bis 255 als zweistellige Hexadezimal zahl.

```
 >BYTE (16)
     10
```

**WORD**
**Diese Anweisung wirkt aehnlich wie Byte. Es werden aber hier 4 Stellen hexadezimal ausgegeben.**

```
 >10 N=1023
 >20 WORD (N)
 >RUN
   03FF
```

**HEX**
**Mittels der HEX-Anweisung wird eine angegebene Hexadezimalzahl in eine Dezimalzahl umgewandelt.**

```
 >10  X=HEX (1000)
 >20  PRINT X
 >RUN
   4096
```

**' ' (QUOTE)**
**Der durch ' '(Hochkomna) dargestellte Befehl QUOTE realisiert die Einzelzeichenunwandlung eines ASCII-Zeichens in einen Dezimalwert. Das ASCII-Zeichen ist zwischen dem Hochkomma anzugeben.**

```
 >10  X='B'
 >20  PRINT X
 >RUN
     66
```

**OUTCHAR**
**Der der OUTCHAR-Anweisung folgende dezimale Ausdruck wird als entsprechendes ASCII-Zeichen auf dem Bildschirm ausgegeben. Bestimmte Sonderzeichen werden sofort ausgefuehrt (siehe OUTCH-Funktion des Monitors).**

```
 >10 OUTCHAR 65
 >20 X=66
 >30 OUTCHAR X
 >40 X='C'
 >50 OUTCHAR X
 >60 PRINT X
 >RUN
 ABC
     67
```

**OUTCHAR 12 loescht z. B. den gesamten Bildschirm.**

**INCHAR**
**Die INCHAR-Anweisung ermoeglicht die Eingabe eines einzelnen Zeichens mittels der Tastatur. Bei der Eingabe dieses Zeichens wird es nicht auf dem Bildschirm ausgegeben. Die Enter-Taste muss nach dar Zeicheneingabe nicht betaetigt werden. Der Wert des ASCII-Zeichens wird der in der Anweisung mit angegebenen Variablen zugewiesen.**

```
 >10 PRINT INCHAR; GOTO 10
 oder
 >10 X=INCHAR; PRINT X; GOTO 10
```

**Mit dieser Programmzeile kann die gesamte Tastatur getestet werden. Es werden alle Tasten mit allen Shift-Tasten kombiniert betaetigt und dadurch der dezimale Zahlenwert auf dem Bildschirm angezeigt. Ein Verlassen dieser Programmschleife ist nur mit S4/K (STOP) moeglich.**

**OUT**
Der in der OUT-Anweisung angegebene Wert des Ausdruckes wird an die in der Anweisung zugewiesene E/A-Adresse des MRB Z1013 ausgegeben. Der Wert darf 255 nicht ueberschreiten (255 ist bekanntlich die groesste Dezimalzahl, die mit 8 Bit darstellbar ist.).

```
 >10 OUT (0)=10
```

Das Bitmuster fuer 10 (00001010B) wird an die E/A-Adresse 0 ausgegeben. In der Grundvariante des MRB Z1013 ist 0H die moegliche E/A-Adresse des PIO-Port's A, die die freie Verwendung durch den Anwender ermoeglicht.
Bei Verwendung der PIO als Port darf die entsprechende Initialisierung nicht vergessen werden, z. B.:

```
 >10 OUT (1)=HEX (CF)
 >20 OUT (1)=0   (Ausgabe)
 bzw.
 >20 OUT (1)=255 (Eingabe)
```

**IN**
Diese Anweisung ermoeglicht die Eingabe von Werten von einer E/A-Adresse des MRB. (Adresse der Grundvariante ist 0H.) Der Wert, der an der E/A-Adresse anliegt, wird einer Variablen zugeordnet.

```
 >10 X=IN (0)
```

**I$**

```
 >10 I$ (TOP)
```

Eingabe einer Zeichenkette ueber Tastatur auf den ersten freien Speicherplatz nach einem BASIC-Programm.
**O$**

```
 >20 O$ (TOP)
```

Ausgabe einer Zeichenkette, die ab dem ersten freien Speicherplatz nach dem BASIC-Programm gespeichert ist, auf dem Bildschirm.

**LEN**
Stellt die Laenge der zuletzt mit einer I$-Anweisung eingegebenen Zeichenkette zur Verfuegung.

```
 >10  I$ (TOP)
 >20  FOR I=0 TO LEN
 >30  IF PEEK (TOP + I) = 'A' PRINT (TOP + I)
 >40  NEXT I
```

Folgende Funktionen werden ausserdem durch den BASIC-Interpreter realisiert:

**RND**
Die RND-Funktion weist einer Variablen einen zufaelligen Wert zwischen 1 und dem in der Anweisung festgelegten Endwert zu.

```
 >10  X=RND (2000)
 >20  PRINT X
 >RUN
   1576
```

**ABS**
Es wird der Absolutbetrag des folgenden Ausdrucks gebildet und einer Variablen zugewiesen.

```
 >10  A=-120
 >20  A=ABS (A)
 >30  PRINT A
 >RUN
   120
```

**TAB**
**Die TAB-Funktion stellt eine sogenannte Tabulatoranweisung dar. Die sich aus dem nachfolgenden Ausdruck ergebende Anzahl von Leerzeichen wird auf dem Bildschirm ausgegeben. Die nachfolgende Ausgabe von Zeichen beginnt nach diesen Leerzeichen.**

```
>10  PRINT 'ANWEISUNG:',
>20  X=5
>30  TAB (X)
>40  PRINT 'TAB',
>50  TAB (6)
>60  PRINT '(',#1,'X,')'
```

**ANWEISUNG TAB (5)**
**Durch die TAB-Punktion wird die Darstellung von Kurven moeglich.**

```
>10  FOR I=-5 TO 5; TAB (I*I); PRINT '*'; NEXT I
```

**TOP**
**Die TOP-Funktion ermittelt den zum Zeitpunkt der TOP-Funktion aktuellen ersten freien Speicherplatz hinter dem soeben eingegebenen BASIC-Programm (dezimal).**

```
>PRINT TOP
  3561
```

**SIZE**
**Damit wird der aktuelle freie RAM-Speicherbereich ermittelt, der fuer ein BASIC-Programm noch zur Verfuegung steht.**

```
>10 PRINT SIZE, 'FREIE BYTE'
>RUN
  260 FREIE BYTE
```

**Die ermittelte Anzahl freier Speicherplaetze wird SIZE zugewiesen und kann im BASIC-Programm weiterverwendet werden.**

## 5.4. Hinweise fuer die Erarbeitung Anwenderprogrammen

### 5.4.1. Allgemeine Hinweise

**In diesem Abschnitt soll dem Ungeuebten die Moeglichkeit gegeben werden, sich anhand von einfachen Beispielen mit grundsaetzlichen Problemen der Programmierung zu befassen. Bevor wir beginnen unserem Mikrorechner eine Leistung abzuverlangen, muessen wir uns darueber im klaren sein, dass er unsere Probleme erst dann loesen hilft, wenn wir ihm eine eindeutige Rechenvorschrift in Form eines Maschinenkode- oder BASIC-Programm geliefert haben.**
**Natuerlich koennen wir uns diese Denkarbeit ersparen und den Rechner mit fertigen Programmen "fuettern", die sich ein anderer fuer uns erdacht hat. Das ist bei der Loesung von haefig wiederkehrenden Problemen zweckmaessig. Hier sei als Beispiel nur das mitgelieferte BASIC-Interpreter-Programm genannt, in dem eine Fuelle geistiger Arbeit steckt und das einem breiten Anwenderkreis eine vereinfachte Programmierung auf der Basis von einheitlichen BASIC-Befehlen ermoeglicht. Wenn es aber gilt, eigene spezifische Probleme rechentechnisch zu loesen, kommen wir um die Aufbereitung nicht herum. Diese notwendige Vorarbeit kann man im wesentlichen in drei Schritte einteilen:**

1. **Problemanalyse**
2. **Erarbeitung der Rechenanweisung (Algorithmus)**
3. **Programmierung**

## 5.4.2. Problemanalyse

Um eine gestellte Aufgabe mit Hilfe unseres Mikrorechners loesen zu koennen, gibt es ider Regel eine Fuelle von Moeglichkeiten. Bei der Praezisierung unseres Problems und der Anpassung an die rechentechnischen Gegebenheiten muessen folgende Gesichtspunkte in Einklang gebracht werden:

- Speicherplatzbedarf
- Verarbeitungszeit
- Uebersichtlichkeit
- Art der Datenein- und -ausgabe
- Bedienungskomfort.

Im folgenden soll das anhand eines Beispiels verdeutlicht werden:
Wir wollen den Inhalt unseres 2K-Zeichengenerators auf dem Bildschirm uebersichtlich darstellen. Die grafische Darstellung soll so erfolgen, dass der Kode aller Zeichen ersichtlich ist (siehe auch Anlage 7). Da fuer die Darstellung eines Zeichens 8 Byte notwendig sind (8x8 Bildpunkte), koennen prinzipiell 2048/8 - 256 verschiedene Zeichen dargestellt werden.
Das Bildschirmformat unseres Rechners bietet uns die Moeglichkeit, 32 Zeilen mit jeweils 32 Zeichen darzustellen, d. h. ein Bild kann aus max. 1024 Zeichen zusammengesetzt werden (siehe Anlage 8). Wir koennten nun mit einem einfachen, kurzen Programm den Zeichengeneratorinhalt mit ansteigender Kode-Zahl "hintereinanderweg" abbilden, das ergibt 256/32 - 8 Zeilen. Darunter wuerde aber die Uebersichtlichkeit leiden, da in dieser gedraengten Darstellung insbesondere die Grafikzeichen, die das 8x8-Bildpunkt-Format ausnutzen, Zum Teil ohne Uebergang ineinanderfliessen. Da wir mit dieser Methode auch nur ein Viertel des Bildschirmes ausnutzen, bietet es sich an, die Zeichendarstellung durch das Einfuegen von Leerzeichen und Leerzeilen aufzuspreizen. Wir erhalten damit 16 beschriebene Zeilen mit jeweils 16 Zeichen. Diese Darstellungsart deckt sich auch gut mit der hexadezimalen Kodierung: In der ersten Zeile stehen die Zeichen 0H.. FH, in der zweiten (beschriebenen) Zeile die Zeichen 10h...1FH usw., so dass wir auf eine Beschriftung der Zeichen, fuer die ohnehin der Raum nicht ausreichen wuerde, verzichten koennen.
Auf Probleme der Verarbeitungezeit verden wir bei der Programmierung noch zusprechen kommen. Prinzipiell waere es moeglich, den Bedienungekomfort zu steigern, z. B. eine Anwahl bestimmter Zeichen oder Zeichengruppen ueber die Tastatur zu ermoeglichen und parallel dazu den Zeichenkode sowohl hexadezimal als auch dezimal darzustellen. Das wuerde aber unseren Forderungen nach Einfachheit und Uebersichtlichkeit (Erfassung aller Zeichen auf einen Blick) widersprechen.

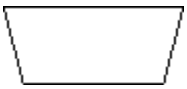
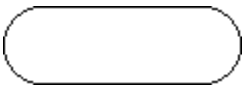
## 5.4.3. Erarbeitung der Rechnenanweisung (Algorithmus)

Nachdem wir unsere jeweilige Aufgabenstellung entsprechend den Empfehlungen des vorhergehenden Abschnittes analysiert und praezisiert haben, messen wir fuer unseren Rechner eine exakte Rechenvorschrift aufstellen.
Fuer unser gewaehltes Beispiel koennte man den Algorithmus folgendermassen beschreiben:

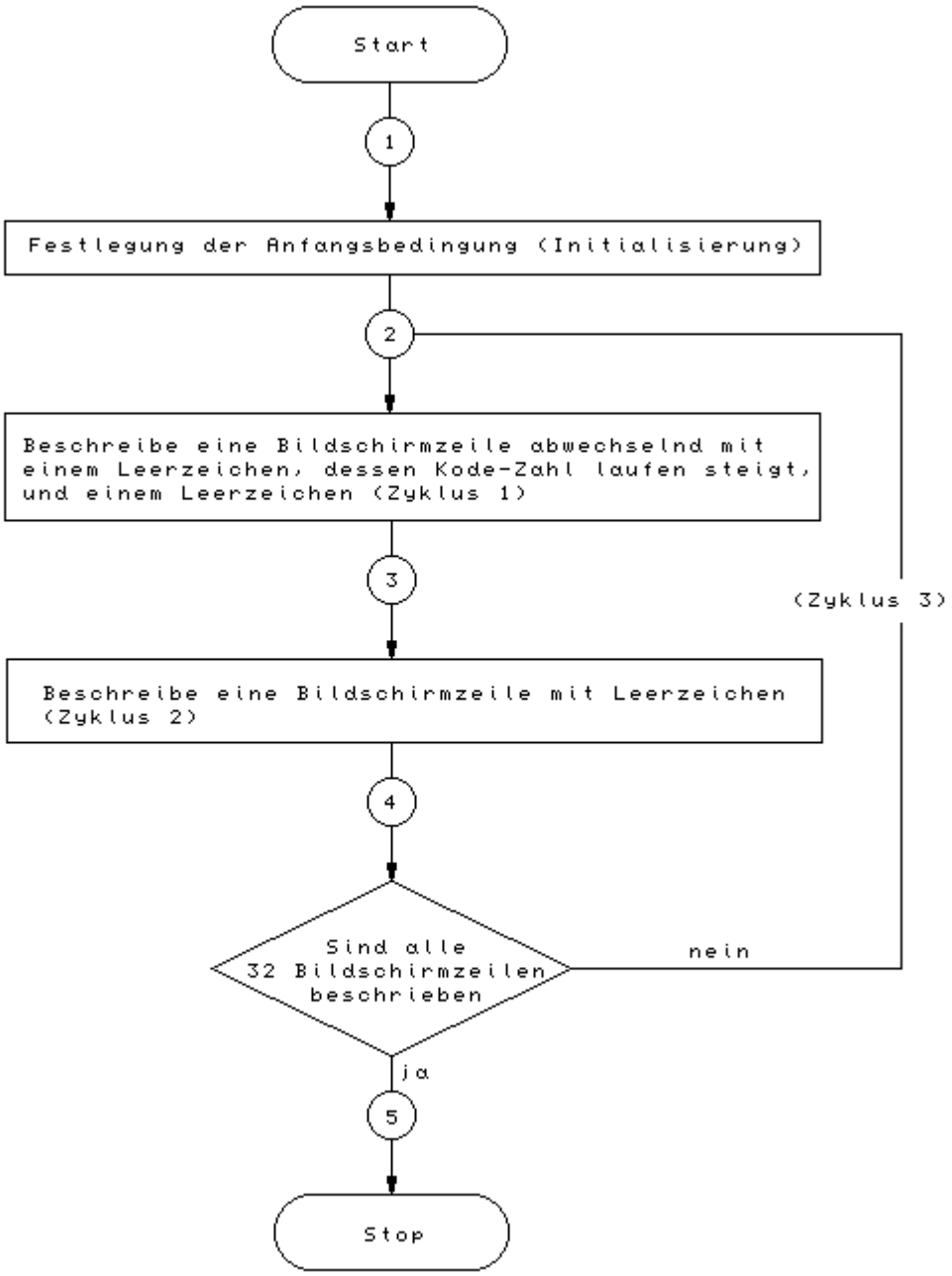
1. Setze auf die Anfangsadresse des Bildschirms das Zeichen mit der Kode-Zahl 0!
2. Setze auf die nachfolgende Adresse ein Leerzeichen!
3. Wiederhole die beiden vorhergehenden Operationen mit wachsender Adresse und ansteigender Kode-Zahl, bis eine Bildschirmzeile voll ist, d. h. 16 mal!

4. Schreibe eine Leerzeile, d. h. setze auf die naechsten 32
   Adressen jeweils 1 Leerzeichen!
7. Beschreibe auf die gleiche Art die folgenden 30 Zeilen!

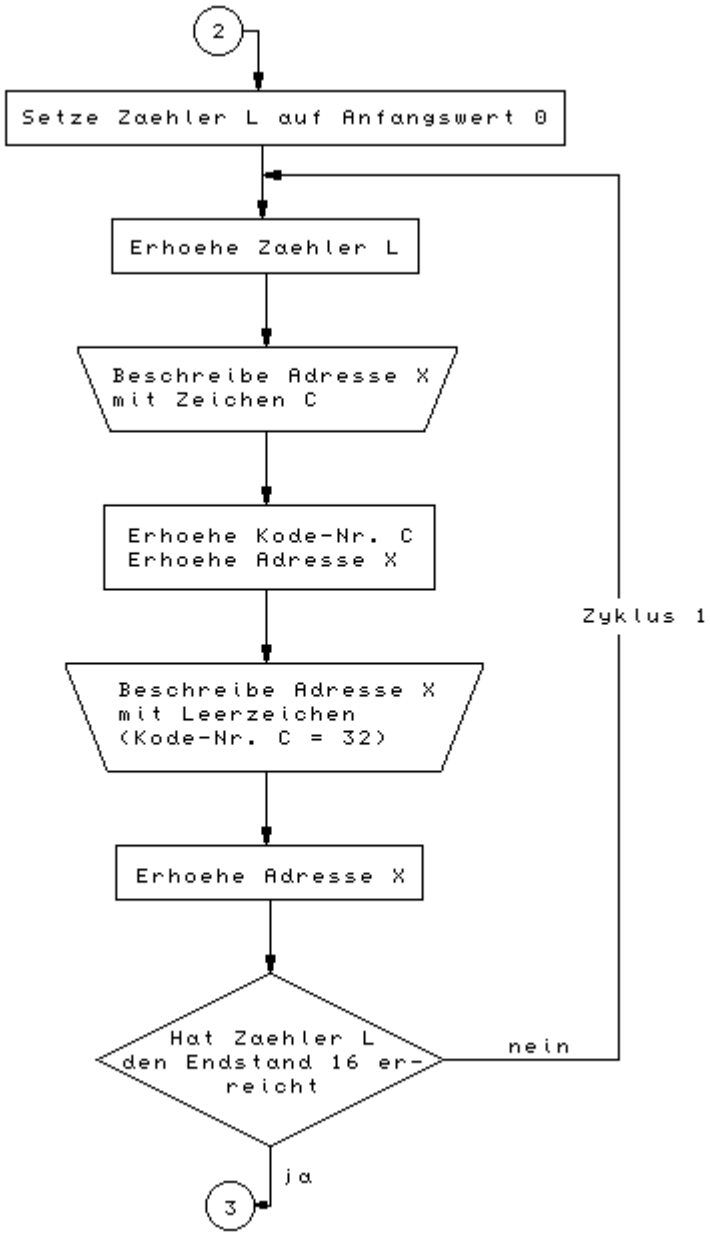
Dieser mit Worten beschriebene Algorithmus stellt eine Liste dar, die von oben nach unten abgearbeitet wird. Dadurch wird es schwierig, sich wiederholende Programmabschnitte (Zyklen) und Programmverzweigumgen (Spruenge) exakt zu beschreiben.
Hierfuer erweist sich der Programmablaufplan (PAP) als ein guenstiges Hilfsmittel bei der Erarbeitung eines Programms.
Durch die Moeglichkeit der zweidimensionalen Darstellung wer den zu loesende Probleme anschaulicher und ueberschaubarer.
Fuer die Gestaltung eines PAP werden folgende Elemente benoetigt:

| Bild | Bedeutung | |
|---|---|---|
|  | Anweisung | einzelne oder mehrere Operationen (Folge) |
|  | Verzweigung | Variation des Programmablaufes in Abhaengigkeit einer oder mehrerer Bedingungen, dadurch Aufbau von Zyklen moeglich |
|  | Eingabe/Ausgabe | beliebige E/A-Operationen |
|  | Grenzstelle | Anfang (Start), Unterbrechung (Zwischenstop) oder Ende (Stop) eines Programmes |
|  | Verbindungsstelle | Verbindung von Programmteilen |
|  | Unterprogramm | Programmteil, das an dieser Stelle aufgerufen wird |

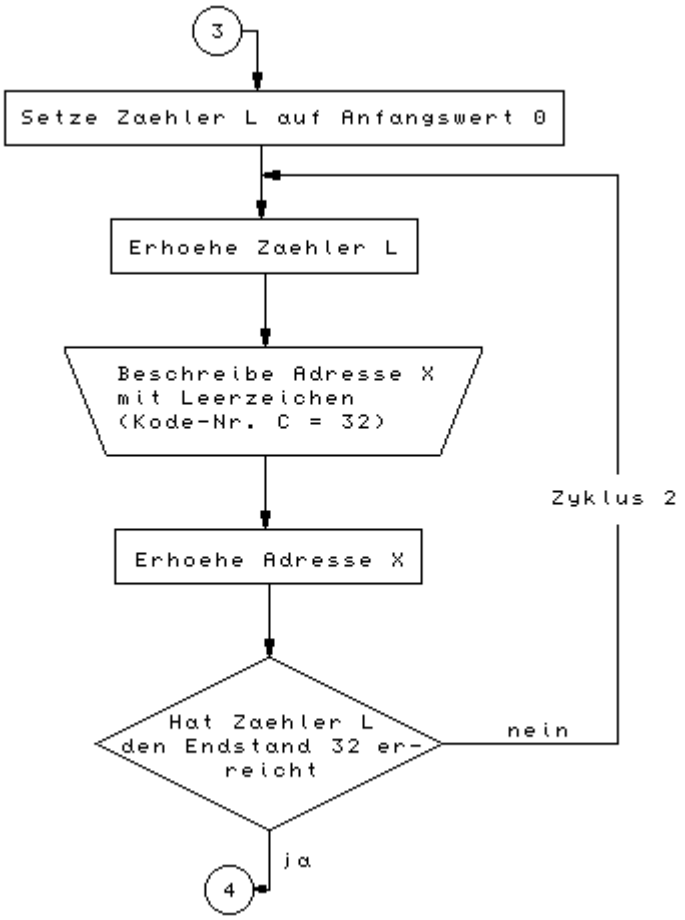
Um bei einem Programmentwurf durch die Vielfalt von Programmteilen und Gestaltungselementen nicht die Uebersicht zu verlieren, empfiehlt es sich, schrittweise den Abstraktionsgrad zu verfeinern.
Diese Methode soll an unserem gewaehlten Beispiel demonstriert werden. Den vorhin angefuehrten verbalen Algorithmus koennte man in einer groben Abstraktion folgendermassen als PAP darstellen:




**Dieser PAP enthaelt viele Vereinfachungen, die noch nicht geeignet sind fuer eine Uebersetzung in die BASIC-Sprache. Insbesondere die Zyklen 1 und 2 sind soweit abstrahiert, dass ihr zyklischer Inhalt gar nicht erkennbar ist.**
**Wir wollen deshalb zunaechst Zyklus 1 praezisieren. Um eine Bildschirmzeile entsprechend unseren Wuenschen zu fuellen, muss die Anweisungsfolge (Zeichen schreiben/Leerzeichen schreiben) mit wachsender Kode-Zahl C jeweils 16 mal wiederholt werden. Fuer die fortlaufende Adressierung des Bildschirms wird die Variable X benutzt, fuer die Anzahl der durchlaufenen Zyklen die Variable L.**
**Der Zyklus 1 stellt eine in sich abgeschlossene funktionelle Einheit dar, die auch als Modul bezeichnet wird. Hier die Darstellung von Zyklus 1 als Teil-PAP:**

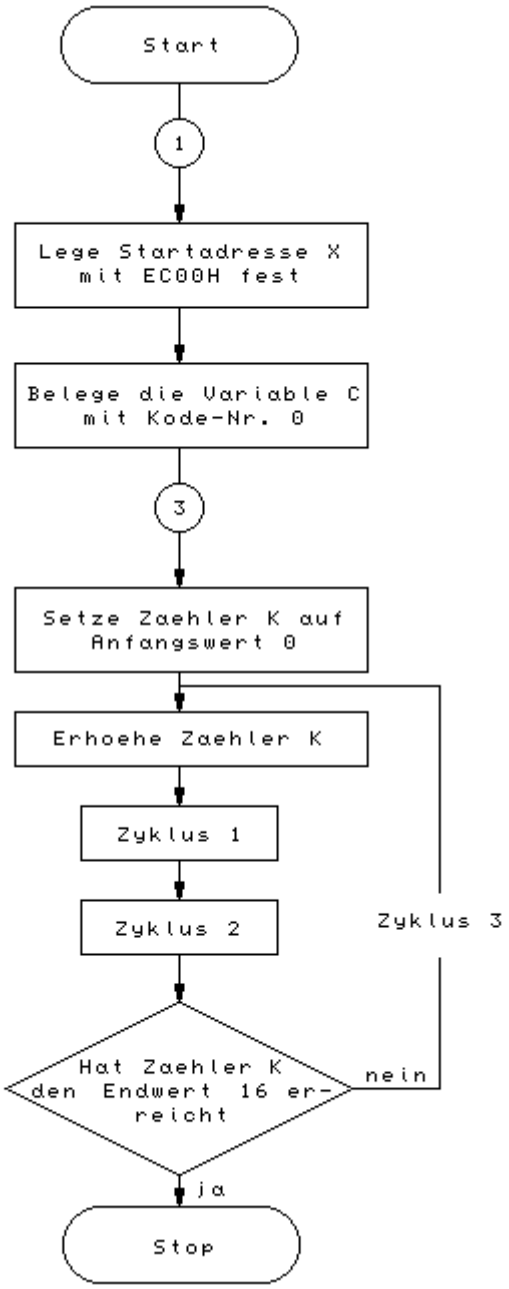



**Zyklus 2 besteht aus der 32fachen zyklischen Wiederholung des Schreibens eines Leerzeichens bei ebenfalls fortlaufender Erhoehung der Adresse X und der Kode-Zahl C. Da der Zyklus 2 an den abgeschlossenen Zyklus 1 anschliesst, kann die Variable L, nachdem ihr ein neuer Anfangswert zugeordnet wurde, erneut fuer das Zaehlen der Zyklen-Anzahl benutzt werden:**




**Mit der einmaligen Realisierung der Folge (Zyklus 1/Zyklus 2) haben wir erst das Programm fuer 2 Bildschirmzeilen abgearbeitet. Um das Gesamtprogramm zu realisieren, muessen wir die genannte Folge 16mal abarbeiten (Zyklus 3). Als Zaehler verwenden wir jetzt die Variable K. Vorher sind noch die Anfangsbedingungen festzulegen. Als erste Bildschirmadresse wird entsprechend Anlage 8 die Adresse EC00H festgelegt, als erste Kode-Zahl wird C = 0 gewaehlt.**

**Der komplette PAP soll im folgenden aus Platzgruenden unter Verwendung der bereits aufgefuehrten Programm-Module Zyklus 1 und Zyklus 2 dargestellt werden:**




**Damit ist unser PAP soweit aufbereitet, dass die Umsetzung in die Programmsprache erfolgen kann. Daraus resultiert auch die wichtige Schlussfolgerung, dass ein Programmablaufplan auf die verwendete Programmsprache und ihre Moeglichkeiten "maasgeschneidert" werden muss, d. h. die Erarbeitung des PAP setzt die gruendliche Kenntnis der Elemente der beabsichtigten Programmiersprache voraus.**

## 5.4.4. Programmierung

Bei der eigentlichen Programmierung koennen wir die Vorteile nutzen, die uns das BASIC durch seine Dialogfaehigkeit bietet. Das bedeutet fuer uns, dass wir unser Programm schrittweise aufbauen und testen koennen.
Wir wollen das wieder an unserem Programmbeispiel ueben. Sicherheitshalber werden bei diesem Beispiel noch einmal die wichtigsten Bedienungshinweise mit angegeben.
Nachdem wir den Anfangszustand hergestellt haben (in Abschn. 1.2.3. ausfuehrlich beschrieben), erwartet der Rechner, der sich jetzt im Betriebsprogramm befindet, eine Bedienereingabe. Es signalisiert das durch Ausgabe des Zeichens "#" als Promptsymbol, gefolgt von einem Leerzeichen und dem Kursor "_".
Gehen wir davon aus, dass wir unseren BASIC-Interpreter entsprechend den Hinweisen von Abschn. 5.3.3. auf Magnetband gespeichert vorliegen haben. Durch Eintippen der Anweisung

```
# L 100 BFF
```

geben wir dem Rechner zu verstehen, dass er in den Speicheradressraum von 100H bis BFFH Daten einlagern soll. Nach dem Positionieren des Magnetbandes (optimal ist hier ein Bandlaengenzaehlwerk) und Einschalten der Wiedergabe erwarten wir den Kennton. Sobald er ertoent, aktivieren wir durch Druecken der ENTER-Taste unsere vorbereitete Bedienereingabe. Der Rechner liest den BASIC-Interpreter ein. Waehrend des Einlesens befindet sich der Kursor am Zeilenanfang. Nach fehlerfreiem Einlesen des BASIC-Interpreters erscheint unter unserer Bedienereingabe wieder das Promptsymbol "#".
Sollte das nicht auf Anhieb klappen, z. B. durch zu spaetes Druecken der ENTER-Taste, erkennen wir das an der Bildschirmausschrift.
Es ist z. B. moeglich, dass durch Staubkoernchen zwischen Band und Tonkopf oder durch Kontaktfehler der Uebertragungsleitung einzelne "Bits" verlorengehen. Das erkennt unser Rechner durch Kontrolle der Uebereinstimmung der Summe aller Daten in einer Zeile mit den ebenfalls im Datensatz uebermittelten Checksummen.
Durch Ausschrift z. B. von

```
CS < 0300
```

zeigt der Rechner uns an, dass im uebertragenen Datenblock < Adresse 0300H ein Checksummenfehler aufgetreten ist. Es kann auch vorkommen, dass der Rechner "steckenbleibt", d. h. der Kursor bleibt am Bildanfang stehen, es erscheint kein Promptsymbol. Da hilft uns nichts weiter, als nach Abstellen der Fehlerursachen und erneutem RESET von vorn anzufangen.
Prinzipiell ist es auch moeglich, beim Auftreten von einzelnen Checksummenfehlern mit dem D-Kommando den angegebenen Datenblock durch Vergleich mit der BASIC-Interpreter-Liste nach rehlern abzusuchen und diese mit dem N-Kommando zu korrigieren.
Im allgemeinen ist es aber weniger umstaendlich, das Einlesen neu zu starten, vorausgesetzt, das auf Band befindliche Naschinenkode-Programm ist fehlerfrei abgespeichert.
Haben wir diesen Schritt erfolgreich abgeschlossen, starten wir den BASIC-Interpreter ab der Anfangsadresse 0100H, indem wir eingeben:

```
J 100.
```

Nach Ausfuehren von ENTER meldet sich der BASIC-Interpreter, wie in 5.3.4. beschrieben und fordert mit dem neuen Promptsymbol ">" eine Bedieneranweisung, jetzt in Form von gueltigen BASIC-Kommandos oder -Programmzeilen.

Beginnen wir jetzt, unseren PAP in die BASIC-Sprache umzusetzen. Als erstes legen wir die Startadresse auf unserem Bildschirm fest. Sie ist uns als EC00H bekannt. Der Interpreter erwartet aber von uns eine dezimale Adressenangabe. Wir koennen die umstaendliche Umrechnung dem Rechner ueberlassen, indem wir den HEX-Befehl benutzen. Unsere erste Programmzeile, der wir entsprechend der Vorschrift aus Abschn. 5.3.4. die Zeilennummer 10 zu ordnen, lautet:

```
 10 X=HEX (EC00).
```

Erst durch Betaetigen von ENTER wird die Zeile als Programmzeile abgespeichert. Vorher ist es noch moeglich, eventuell notwendige Korrekturen durch Verschieben des Kursors mit "<--" und "-->", durch Ueberschreiben mit anderen Zeichen oder durch Loeschen mit der Leerzeichentaste " "anzubringen. Wichtig ist es, dass nur die vor dem Kursor befindlichen Zeichen bei Abschluss mit ENTER in die Programmzeile uebernommen werden.
Also:
Vor dem ENTER noch einmal auf den Bildschirm schauen!
Die naechste Anweisung aus dem PAP ergibt eine neue Programmzeile:

```
 20 C=0
```

Damit wird der Variablen C die Zahl 0 zugeordnet. Das ist unsere Anfangsbedingung, die sichert, dass als erstes Zeichen das mit der Kode-Zahl 0 aus dem Zeichengenerator geholt wird.

Um anfaenglichen Unsicherheiten bei fehlerhafter Bedienerfuehrung von vornherein entgegenzutreten, ist an dieser Stelle noch einmal der Hinweis angebracht, die Abschnitte 5.3.4. und 5.3.5. besonders gruendlich zu studieren und anzuwenden. Sollte z. B. durch mehrfache Fehlerausschriften, Aenderungen usw. die Programdarstellung auf dem Bildschirm an Uebersicht verlieren, kann man jederzeit durch Eingabe des LIST-Kommados den aktuellen Stand aller geordneten Programmzeilen erfahren.
Wenden wir uns nun zunaechst der Programmierung von Zyklus 1 zu (vgl. PAP zu Zyklus 1).
Der Variablen L wird der Anfangswert 0 zugeordnet:

```
 50 L=0
```

Da die Variable L als Zaehler dient, muss sie in jeder Programmschleife um den Wert 1 erhoeht werden:

```
 60 L=L+1
```

Zum Beschreiben der Bildschirmdresse X mit dem Zeichen, Kode-Zahl C, wird der POKE-Befehl angewendet:

```
 70 POKE X,C
```

Die Kode-Nr. und die Bildschirmadresse werden anschliessend fuer die nacehste Prograamschleife um 1 erhoeht:

```
 80 C=C+1
 90 X=X+1
```

Auf diese neue Adresse schreiben wir ein Leerzeichen (Kode-Nr. 32) und erhoehen anschliessend wieder die Adresse:

```
 100 POKE X,32
 110 X=X+1
```

Damit diese Befehlsfolge nach eimaligem Durchlauf beendet wird, verwenden wir den bedingten Sprung IF..GOTO Als Abbruchbedingung wird der Stand unseres Zaehlers L kontrolliert:

```
 120 IF L<16 GOTO 60
```

Sind wir hier erfolgreich angelangt (zweckmaessig ist hier noch einmal die Kontrolle des bisherigen Programms mit LIST), koennen wir das Teilprogramm bereits durch Eingabe des Kommandos RUN starten und damit das Beschreiben der ersten Bildschirmzeile ausloesen.

Sollten wir aber vorher bereits durch unser Programmieren die unterste Bildschirmzeile erreicht haben, wird durch das infolge der Fertigmeldung mit READY und Erscheinen des Promptsymbols um 2 Zeilen weiterrollende Bild unser "Programmiererfolg" wieder verschwinden. Um das von vornherein zu verhindern, wenden wir einen Trick an, indem wir den Rechner "endlos" beschaeftigen. Dazu koennen wir z. B. eine Programmschleife konstruieren, die den Rechner zwingt, solange einen bedingten Sprung auszufuehren, bis durch unseren Eingriff ein Abbruch erfolgt:

```
 200 GOTO 200
```

Befindet sich der Rechner in einer solchen Schleife, reagiert er nicht mehr auf normale Eingabebefehle. Einen Abbruch erreichen wir Jetzt nur durch gleichzeitiges Betaetigen von 54 und K. Dann erscheint auch wieder das Promptsymbol als Zeichen der Bereitschaft.
Wuenschen wir ueber Bedienereingabe einen leeren Bildschirm, betaetigen wir gleichzeitig S4 und T mit anschliessendem ENTER.
Kommenn wir nun zum Zyklus 2. Mit unseren Erfahrungen vom Zyklus 1 koennen wir ihn jetzt sofort umsetzen:

```
 130 L=0
 140 L=L+1
 150 POKE X,HEX(20)
 160 X=X+1
 170 IF L<32 GOTO 140
```

Wer sich nicht damit abfinden kann, dass die zweite Zeile mit ihren 32 Leerzeichen "unsichtbar" bleibt, kann z. B. anstelle der Leerzeichen das Zeichen "*" (Kode-Nr. 2AH) verwenden:

```
 150 POKE X,HEX(2A)
```

Dieser Befehl laesst sich spaeter auf Wunsch wieder zurueckwandeln. Durch LIST koennen wir uns ueberzeugen, dass der neue Programmteil sich entsprechend der Zeilennumerierung in das Gesamtprogramm eingefuegt hat.
Bei erneutem Starten des bisherigen Programms durch RUN werden jetzt die ersten beiden Bildschirmzeilen ueberschrieben. (Nicht vergessen: vor dem Weiterarbeiten S4/K druecken!)
Um unser Gesamtprogramm zu verwirklichen, muessen wir noch den Zyklus 3 umsetzen:

```
 30 K=0
 40 K=K+1
 180 IF K<16 GOTO 40
```

Wenn wir alles erfolgreich bewaeltigt haben, erscheint jetzt nach RUN der komplette Zeichengenerator in angemessenen Tempo auf dem Bildschirm. Damit ist unser kleines Programm fertig. Es soll aber gezeigt werden, dass es auch andere Moeglichkeiten gibt, zum gleichen Ziel zu gelangen. Wir koennen z. B. vor dem Zyklus 1 den gesamten Bildschirm loeschen. Damit entfaellt der Zyklus 2.

Benutzen wir fuer das Bildschirmloeschen die entsprechende Monitorroutine, indem wir ueber den OUTCHAR-Befehl das Steuerzeichen 0CH = 12D abrufen (s.a. Abschnitt 5.1.1.), so haben wir den Vorteil, dass das Loeschen in wesentlich kuerzerer Zeit realisiert wird. Zum Ueben ist es zu empfehlen, hierfuer den PAP abzuaendern. Wenn wir zur Realisierung der Programmschleifen gleich noch die elegantere FOR...TO...NEXT-Anweisung benutzen, koennen wir durch Loeschen und Ueberschreiben einiger Programzeilen folgende Variante erhalten:

```
 10 X=HEX (EC00)
 20 C=0
 30 OUTCHAR 12
 40 FOR K=1 TO 16
 60 FOR L=1 TO 16
 70 POKE X,C
 80 C=C+1
 90 X=X+2
120 NEXT L
140 X=X+32
180 NEXP K
200 GOTO 200
```

Zeile 30 bewirkt das schlagartige Loeschen des gesamten Bildschirmes. Zeile 60..120 realisiert Zyklus 1, Zeile 40..180 stellt den Zyklus 3 dar, Zeile 90 rueckt die Bildschirmadresse um jeweils 2 Plaetze weiter, Zeile 140 bewirkt das Ueberspringen einer Zeile.
Einen interessanten Aspekt bietet die Gegenueberstellung zu einem Maschinenprogramm, das nach demselben Algorithmus arbeitet wie unser Ausgangsprogramm. Es soll im folgenden komplett wiedergegeben werden (man beachte die Ausfuehrungezeit im Vergleich zum BASIC-Programm):

```
3000  3E 00    LD A , 00
3002  21 00 30 LD HL, EC00
3005  06 10    LD B , 10
3007  C5       PUSH BC
3008  06 10    LD B , 10
300A  77       LD M , A
300B  23       INC HL
3000  36 20    LD M , 20
3003  23       INC HL
30Q?  30       INC A
3010  10 F8    DJNZ 300A - #
3012  06 20    LD B , 20
3014  36 20    LD B , 20
3016  23       INC HL
3017  10 FB    DJNZ 3014 - #
3019  Cl       POP BC
301A  10 EB    DJNZ 3007 - #
3010  76       HALT
```

## 6. Erweiterungen des MRB Z1013

### 6.1. Allgemeine Hinweise

Die Grundausbaustufe des MRB Z1013 stellt das Kernstueck eines Mikrorechnersystems dar, welches durch zusaetzliche Baugruppen immer weiter komplettiert werden kann.
Diese Erweiterungsbaugruppen sollen ebenfalls industriell gefertigt und im Handel angeboten werden. Dazu gehoert ein Baugruppentraeger mit Rueckverdrahtung, die ausser den Bauelementen zur Verstaerkung der Systemsignale eine Anzahl von Steckverbinderplaetzen enthaelt.
Versierte Bastler, die neben der Beschaeftigung mit dem MRB Z1013 auch Freude am Selbstbau elektronischer Baugruppen haben, werden sich ihre Erweiterungen vielleicht selbst bauen. Dieser Selbstbau ist aber nur erfahrenen Elektronikern anzuraten, da tiefgruendiges Wissen und grosse Erfahrung vorausgesetzt werden. Weiterhin ist zu beachten, dass aus Kostengruenden die Grundausbaustufe nur die unbedingt notwendigen Bauelemente enthaelt, die Systemsignale an den Steckverbinderanschluessen meist unverstaerkt und direkt von der CPU kommen und damit Fehler in Erweiterungsschaltungen die Grundausbaustufe zerstoeren koennen.
Deshalb sollte bei umfangreichen Erweiterungen im Selbstbau die Verstaerkung und Entkopplung der Systemsignale sowie ein leistungsfaehigeres Netzteil an erster Stelle stehen.
Aus den angefuehrten Gruenden werden zu Hardwareerweiterungen nur einige grundsaetzliche Hinweise gegeben, die fuer erfahrene Elektroniker ausreichen duerften. Auf die Wiedergabe von Schaltungen wird bewusst verzichtet.
ACHTUNG! Beachten Sie unbedingt den folgenden Hinweis. Es entfallen bei der Durchfuehrung von Loetarbeiten auf der Z1013-Leiterplatte, ausgenommen das Anloeten der Tastaturkabel, im Garantiezeitraum alle Garantieansprueche.

### 6.2. Speichererweiterungen

Die Grundausbaustufe des MRB Z1013 besitzt standardmaessig einen 2K Byte ROM mit dem Monitorprogramm im Adressbereich F000H bis F7FFH und einen 1K Byte umfassenden statischen RAM im Bereich EC00H bis EFFFH als BWS. Zusaetzlich ist er abhaengig von der Bestueckungsvariante mit 16K Byte dynamischen RAM (Z1013.01) oder mit 1K Byte statischen RAM (Z1013.12) ab Adresse 0000H ausgeruestet.
Die letztgenannte Variante erfordert fuer die Nutzung des BASIC-Interpreters einen groesseren Arbeitsspeicher.
Achtung! Bei allen Erweiterungen ist zu beachten, dass die Belastbarkeit der Signalleitungen und der Stromversorgung nur fuer die Grundausbaustufe ausgelegt ist.

## 6.3. Anschluss von Steuereinheiten

**Fuer kleine Steuer und Regelungsaufgaben steht als E/A-Port eine 8 Bit-Schnittstelle des PIO's zur Verfuegung (PIO-Port A). Dieser Peripherjeanschluss hat die ElAGrundadresse 0. Die erforderlichen Anschluesse stehen am Steckverbinder X4 zur Verfuegung (siehe Anlage 6). Je nach Umfang der zu steuernden Aufgabe kann das bereits ausreichend sein. Das Fort wird im Modus bitgesteuerte E/A betrieben und kann sowohl Eingabesignale entgegennehmen als auch den Prozess boeinflussende Signale abgeben.**
**Sollte bei gewachsenen Anforderungen dieser 8 Bit breite Fort nicht ausreichen, kann zusaetzlich mit den Spaltenauswahlleitungen der Tastatur eine Auswahl der Signalquellen sowie der zu steuernden Einheiten vorgenommen werden. Es koennen damit 10 Jeweils 8 Bit breite E/A-Ports ausgewaehlt werden, ohne dass zusaetzlicher Dekodieraufwand notwendig ist. Beispiel: Es wird eine Information auf dem PIO-Fort A ausgegeben. Die auszugebenden Daten werden durch den PIO am Steckverbinder X4 zur Verfuegung gestellt. Durch die Ausgabe einer Spaltennummer zwischen 0 bis 9 auf die E/A-Adresse 8 wird die entsprechende Spaltenleitung aktiviert (Low-aktiv) und gewaehrleistet die Datenuebernahme in das ausgewaehlte Port. Da durch den Spaltendekoder 10 Spaltenleitungen aktiviert werden koennen, ist der Anschluss von 10 verschiedenen E/A-Ports am Steckverbinder X4 moeglich. Damit stehen einem Anwender ohne grossen zusaetzhohen Aufwand 80 Kommandoleitungen zur Verfuegung, die nach eigenem Ermessen in Ein- und Ausgabeleitungen eingeteilt werden koennen.**

R O B O T R O N

Mikrorechnerbausatz Z 1 0 1 3

## Anlagenteil

VEB Robotron - Elektronik Riesa

---

Mikrorechnerbausatz Z 1 0 1 3

Anlagen:
---


1 Befehlssatz U 880
2 Speicherbereichsaufteilung
3 E/A Adressen
4 Arbeitszellen des Monitors
5 BASIC-Befehlsliste
6 Steckverbinderbelegung
7 Zeichensatz/ASCII-Code
8 Bildschirmadressen
9 Verwendete Schaltkreise
10 Zeitverhalten einiger CPU-Funktionen
11 Liste des Reassemblers fuer U 880-Befehle
12 Liste des BASIC-Interpreters
13 MC-Beispielprogramme
14 BASIC Beispielprogramme
15 Belegungsplan
16 Stromlaufplan


## Anlage 1: Befehlssatz der CPU 880

8-Bit-Ladebefehle

| | A | B | C | D | E | H | L | (HL) | (BC) | (DE) | (nn) | n |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LD A,. | 7F | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 0A | 1A | 3AXXXX | 3EXX |
| LD B,. | 47 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | | | | 06XX |
| LD C,. | 4F | 48 | 49 | 4A | 4B | 4C | 4D | 4E | | | | 0EXX |
| LD D,. | 57 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | | | | 16XX |
| LD E,. | 5F | 58 | 59 | 5A | 5B | 5C | 5D | 5E | | | | 1EXX |
| LD H,. | 67 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | | | | 26XX |
| LD L,. | 6F | 68 | 69 | 6A | 6B | 6C | 6D | 6E | | | | 2EXX |
| LD (HL),. | 77 | 70 | 71 | 72 | 73 | 74 | 75 | | | | | 36XX |
| LD (BC),. | 02 | | | | | | | | | | | |
| LD (DE),. | 12 | | | | | | | | | | | |
| LD (nn),. | 32XXXX | | | | | | | | | | | |

| A | B | C | D | E | H | L |
|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| LD ., (IX+d) | DD7Exx | DD46XX | DD4EXX | DD56XX | DD5EXX | DD66XX | DD6EXX |
| LD ., (IY+d) | FD7Exx | FD46XX | FD4EXX | FD56XX | FD5EXX | FD66XX | FD6EXX |
| LD (IX+d),. | DD77XX | DD70XX | DD71XX | DD72XX | DD73XX | DD74XX | DD75XX |
| LD (IX+d),. | FD77XX | FD70XX | FD71XX | FD72XX | FD73XX | FD74XX | FD75XX |

| | |
|---|---|
| LD (IX+d),n | DD36ddnn |
| LD (IX+d),n | FD36ddnn |

| | | S | Z | H | P/V | N | C |
|---|---|---|---|---|---|---|---|
| LD A,I | ED57 | * | * | 0 | * | 0 | - |
| LD A,R | ED5F | * | * | 0 | * | 0 | - |
| LD I,A | ED47 | - | - | - | - | - | - |
| LD R,A | ED4F | - | - | - | - | - | - |

16-Bit-Ladebefehle

| | BC | DE | HL | SP | IX | IY |
|---|---|---|---|---|---|---|
| LD .., nn | 01XXXX | 11XXXX | 21XXXX | 31XXXX | DD21XXXX | FD21XXXX |
| LD ..,(nn) | ED4BXXXX | ED5BXXX | 2AXXXX | ED7BXXXX | DD2AXXXX | FD2AXXXX |
| LD (nn),.. | ED43XXXX | ED53XXXX | 22XXXX | ED73XXXX | DD22XXXX | FD22XXXX |
| LD SP,.. | | | F9 | | DDF9 | FDF9 |

| | BC | DE | HL | AF | IX | IY |
|---|---|---|---|---|---|---|
| PUSH .. | C5 | D5 | E5 | F5 | DDE5 | FDE5 |
| POP .. | C1 | D1 | E1 | F1 | DDE1 | FDE1 |

| | | | |
|---|---|---|---|
| EX (SP),HL | E3 | EX DE,HL | EB |
| EX (SP),IX | DDE3 | EXAF | 08 |
| EX (SP),IY | FDE3 | EXX | D9 (BC-BC' DE-DE' HL-HL') |

**Blocktansfer- und Suchbefehle**

| | | S | Z | H | P/V | N | C | |
|---|---|---|---|---|---|---|---|---|
| LDI | EDA0 | - | - | 0 | * | 0 | - | LD (DE), (HL); INC HL; INC DE; DEC BC |
| LDIR | EDB0 | - | - | 0 | 0 | 0 | - | wie LDI, wiederholen bis BC=0 |
| LDD | EDA8 | - | - | 0 | * | 0 | - | LD (DE), (HL); DEC HL; DEC DE; DEC BC |
| LDDR | EDB8 | - | - | 0 | 0 | 0 | - | wie LDD, wiederholen bis BC=0 |
| CPI | EDA1 | * | * | * | * | 1 | - | LD A,(HL); INC HL, DEC BC |
| CPIR | EDB1 | * | * | * | * | 1 | - | wie CPI, wiederholen bis BC=0 oder Zeichen gefunden |
| CPD | EDA9 | * | * | * | * | 1 | - | LD A,(HL); DEC HL, DEC BC |
| CPDR | EDB9 | * | * | * | * | 1 | - | wie CPD, wiederholen bis BC=0 oder Zeichen gefunden |

**Sprungbefehle**

| | Z | NZ | C | NC | PE | PC | M | P |
|---|---|---|---|---|---|---|---|---|
| JP.. | CAXXXX | C2XXXX | DAXXXX | D2XXXX | EAXXXX | E2XXXX | FAXXXX | F2XXXX |
| CA.. | CCXXXX | C4XXXX | DCXXXX | D4XXXX | ECXXXX | E4XXXX | FCXXXX | F4XXXX |
| RE.. | C8 | C0 | D8 | D0 | E8 | E0 | F8 | F0 |
| JR.. | 28XX | 20XX | 38XX | 30XX | | | | |

| | unbedingt | (HL) | (IX) | (IY) |
|---|---|---|---|---|
| JMP | C3XXX | E9 | DDE9 | FDE9 |
| CALL | CDXXX | | | |
| RET | C9 | | | |
| JR | 18XX | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| RST | 00 | 08 | 10 | 18 | 20 | 28 | 30 | 38 |
| | C7 | CF | D7 | DF | E7 | EF | F7 | FF |

DJNZ 10XX DEC B; JRNZ e

RETI ED4D zurueck vom Interrupt

RETN ED45 zurueck vom nicht maskierbaren Interrupt

**CPU-Steuerbefehle**

| | | S | Z | H | P/V | N | C | |
|---|---|---|---|---|---|---|---|---|
| NOP | 00 | - | - | - | - | - | - | Leerbefehl |
| HALT | 76 | - | - | - | - | - | - | |
| CCF | 3F | - | - | * | - | 0 | * | Komplementiere Carry-Flag |
| SCF | 37 | - | - | 0 | - | 0 | 1 | Setze Carry Flag |
| EI | FB | - | - | - | - | - | - | Interrupts freigeben |
| DI | F3 | - | - | - | - | - | - | Interrupts sperren |
| IM 0 | ED46 | - | - | - | - | - | - | Interrupt-Modus 0 |
| IM 1 | ED56 | - | - | - | - | - | - | Interrupt-Modus 1 |
| IM 2 | ED5E | - | - | - | - | - | - | Interrupt-Modus 2 |

**Ein-/Ausgabebefehle**

| | A | B | C | D | E | H | L | S | Z | H | P/V | N | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IN | ED78 | ED40 | ED48 | ED50 | ED58 | ED60 | ED68 | * | * | 0 | * | 0 | - |
| OUT | ED79 | ED41 | ED49 | ED51 | ED59 | ED61 | ED69 | - | - | - | - | - | - |

(Kanaladresse in C)

INF ED70 Setzen des Flag Registers * * ? * 0 -

(Kanaladresse in C)

| | | S | Z | H | P/V | N | C | |
|---|---|---|---|---|---|---|---|---|
| IN n | DBXX | - | - | - | - | - | - | Kanaladresse 'n' |
| OUT n | D3XX | - | - | - | - | - | - | Kanaladresse 'n' |
| INI | EDA2 | ? | * | ? | ? | 1 | - | IN (HL),(C); INC HL; DEC B |
| INIR | EDB2 | ? | 1 | ? | ? | 1 | - | wie INI, wiederholen solange B<>0 |
| IND | EDAA | ? | * | ? | ? | 1 | - | IN (HL),(C); DEC HL; DEC B |
| INDR | EDBA | ? | 1 | ? | ? | 1 | - | wie IND, wiederholen solange B<>0 |
| OUTI | EDA3 | ? | * | ? | ? | 1 | - | OUT (C),(HL); INC HL; DEC B |
| OTIR | EDB3 | ? | 1 | ? | ? | 1 | - | wie OUTI, wiederholen solange B<>0 |
| OUTD | EDAB | ? | * | ? | ? | 1 | - | OUT (C),(HL); DEC HL; DEC B |
| OTDR | EDBB | ? | 1 | ? | ? | 1 | - | wie OUTD, wiederholen solange B<>0 |

## 8-Bit Arithmetische und Logische Befehle

| | B | C | D | E | H | L | (HL) | A | n | (IX+d) | (IY+d) | S | Z | H | P/V | N | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | C6XX | DD86XX | FD86XX | * | * | * | * | 0 | * |
| ADC | 88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F | CEXX | DD8EXX | FD8EXX | * | * | * | * | 0 | * |
| SUB | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | D6XX | DD96XX | FD96XX | * | * | * | * | 1 | * |
| SBC | 98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F | DEXX | DD9EXX | FD9EXX | * | * | * | * | 1 | * |
| AND | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | E6CC | DDA6XX | FDA6XX | * | * | 1 | * | 0 | 0 |
| XOR | A8 | A9 | AA | AB | AC | AD | AE | AF | EEXX | DDAEXX | FDAEXX | * | * | 1 | * | 0 | 0 |
| OR | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | F6XX | DDB6XX | FDB6XX | * | * | 1 | * | 0 | 0 |
| CMP | B8 | B9 | BA | BB | BC | BD | BE | BF | FEXX | DDBEXX | FDBEXX | * | * | * | * | 1 | * |
| INC | 04 | 0C | 14 | 1C | 24 | 2C | 34 | 3C | | DD34XX | FD34XX | * | * | * | * | 0 | - |
| DEC | 05 | 0D | 15 | 1D | 25 | 2D | 35 | 3D | | DD35XX | FD35XX | * | * | * | * | 1 | - |

| | | S | Z | H | P/V | N | C | |
|---|---|---|---|---|---|---|---|---|
| DAA | 27 | * | * | * | * | - | * | BCD-Korrektur im A-Register |
| CPL | 2F | - | - | 1 | - | 1 | - | Komplementiere A-Register (1er-Komplement) |
| NEG | ED44 | * | * | * | * | 1 | * | Komplementiere A-Register (2er-Komplement) |

## Rotations- und Schiebebefehle

| | B | C | D | E | H | L | (HL) | A | (IX+d) | (IY+d) |
|---|---|---|---|---|---|---|---|---|---|---|
| RR | CB18 | CB19 | CB1A | CB1B | CB1C | CB1D | CB1E | CB1F | DDCBXX1B | FDCBXX1E |
| RL | CB10 | CB11 | CB12 | CB13 | CB14 | CB15 | CB16 | CB17 | DDCBXX16 | FDCBXX16 |
| RRC | CB08 | CB09 | CB0A | CB0B | CB0C | CB0D | CB0E | CB0F | DDCBXX0E | FDCBXX0E |
| RLC | CB00 | CB01 | CB02 | CB03 | CB04 | CB05 | CB06 | CB07 | DDCBXX06 | FDCBXX06 |
| SRA | CB28 | CB29 | CB2A | CB2B | CB2C | CB2D | CB2E | CB2F | DDCBXX2E | FDCBXX2E |
| SLA | CB20 | CB21 | CB22 | CB23 | CB24 | CB25 | CB26 | CB27 | DDCBXX26 | FDCBXX36 |
| SRL | CB38 | CB39 | CB3A | CB3B | CB3C | CB3D | CB3E | CB3F | DDCBXX3E | FDCBXX3E |

| | S | Z | H | P/V | N | C | |
|---|---|---|---|---|---|---|---|
| RR/RL | * | * | 0 | * | 0 | * | Rotiere Reg rechts/links durch Carry |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| RRC/RLC | * | * | 0 | * | 0 | * | Rotiere Reg rechts/links |
| SRA/SLA | * | * | 0 | * | 0 | * | Shift Reg rechts/links arithmetisch |
| SRL | * | * | 0 | * | 0 | * | Shift Reg rechts/links logisch |

| | | S | Z | H | P/V | N | C | |
|---|---|---|---|---|---|---|---|---|
| RRCA | 0F | - | - | 0 | - | 0 | * | Rotiere A-Register rechts |
| RLCA | 07 | - | - | 0 | - | 0 | * | Rotiere A-Register links |
| RRA | 1F | - | - | 0 | - | 0 | * | Rotiere A-Register rechts durch Carry |
| RLA | 17 | - | - | 0 | - | 0 | * | Rotiere A-Register links durch Carry |
| RLD (HL) | ED6F | * | * | 0 | * | 0 | - | Rotiere Ziffer links zwischen A-Register und (HL) |
| RRD (HL) | ED67 | * | * | 0 | * | 0 | - | Rotiere Ziffer rechts zwischen A-Register und (HL) |

**Einzelbitbefehle**

| | B | C | D | E | H | L | (HL) | A | (IX+d) | (IY+d) |
|---|---|---|---|---|---|---|---|---|---|---|
| BIT 0,. | CB40 | CB41 | CB42 | CB43 | CB44 | CB45 | CB46 | CB47 | DDCBXX46 | FDCBXX46 |
| BIT 1,. | CB48 | CB49 | CB4A | CB4B | CB4C | CB4D | CB4E | CB4F | DDCBXX4E | FDCBXX4E |
| BIT 2,. | CB50 | CB51 | CB52 | CB53 | CB54 | CB55 | CB56 | CB57 | DDCBXX56 | FDCBXX56 |
| BIT 3,. | CB58 | CB59 | CB5A | CB5B | CB5C | CB5D | CB5E | CB5F | DDCBXX5E | FDCBXX5E |
| BIT 4,. | CB60 | CB61 | CB62 | CB63 | CB64 | CB65 | CB66 | CB67 | DDCBXX66 | FDCBXX66 |
| BIT 5,. | CB68 | CB69 | CB6A | CB6B | CB6C | CB6D | CB6E | CB6F | DDCBXX6E | FDCBXX6E |
| BIT 6,. | CB70 | CB71 | CB72 | CB73 | CB74 | CB75 | CB76 | CB77 | DDCBXX76 | FDCBXX76 |
| BIT 7,. | CB78 | CB79 | CB7A | CB7B | CB7C | CB7D | CB7E | CB7F | DDCBXX7E | FDCBXX7E |
| RES 0,. | CB80 | CB81 | CB82 | CB83 | CB84 | CB85 | CB86 | CB87 | DDCBXX86 | FDCBXX86 |
| RES 1,. | CB88 | CB89 | CB8A | CB8B | CB8C | CB8D | CB8E | CB8F | DDCBXX8E | FDCBXX8E |
| RES 2,. | CB90 | CB91 | CB92 | CB93 | CB94 | CB95 | CB96 | CB97 | DDCBXX96 | FDCBXX96 |
| RES 3,. | CB98 | CB99 | CB9A | CB9B | CB9C | CB9D | CB9E | CB9F | DDCBXX9E | FDCBXX9E |
| RES 4,. | CBA0 | CBA1 | CBA2 | CBA3 | CBA4 | CBA5 | CBA6 | CBA7 | DDCBXXA6 | FDCBXXA6 |
| RES 5,. | CBA8 | CBA9 | CBAA | CBAB | CBAC | CBAD | CBAE | CBAF | DDCBXXAE | FDCBXXAE |
| RES 6,. | CBB0 | CBB1 | CBB2 | CBB3 | CBB4 | CBB5 | CBB6 | CBB7 | DDCBXXB6 | FDCBXXB6 |
| RES 7,. | CBB8 | CBB9 | CBBA | CBBB | CBBC | CBBD | CBBE | CBBF | DDCBXXBE | FDCBXXBE |
| SET 0,. | CBC0 | CBC1 | CBC2 | CBC3 | CBC4 | CBC5 | CBC6 | CBC7 | DDCBXXC6 | FDCBXXC6 |
| SET 1,. | CBC8 | CBC9 | CBCA | CBCB | CBCC | CBCD | CBCE | CBCF | DDCBXXCE | FDCBXXCE |

SET 2,. CBD0 CBD1 CBD2 CBD3 CBD4 CBD5 CBD6 CBD7 DDCBXXD6 FDCBXXD6

SET 3,. CBD8 CBD9 CBDA CBDB CBDC CBDD CBDE CBDF DDCBXXDE FDCBXXDE

SET 4,. CBE0 CBE1 CBE2 CBE3 CBE4 CBE5 CBE6 CBE7 DDCBXXE6 FDCBXXE6

SET 5,. CBE8 CBE9 CBEA CBEB CBEC CBED CBEE CBEF DDCBXXEE FDCBXXEE

SET 6,. CBF0 CBF1 CBF2 CBF3 CBF4 CBF5 CBF6 CBF7 DDCBXXF6 FDCBXXF6

SET 7,. CBF8 CBF9 CBFA CBFB CBFC CBFD CBFE CBFF DDCBXXFE FDCBXXFE

Flagbeeinflussung der Einzelbitbefehle:

| | S | Z | H | P/V | N | C |
|---|---|---|---|---|---|---|
| BIT | ? | * | 1 | ? | 0 | - |
| SET | - | - | - | - | - | - |
| RES | - | - | - | - | - | - |

Flag-Register

| BIT | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | S | Z | X | H | X | P/V | N | C |

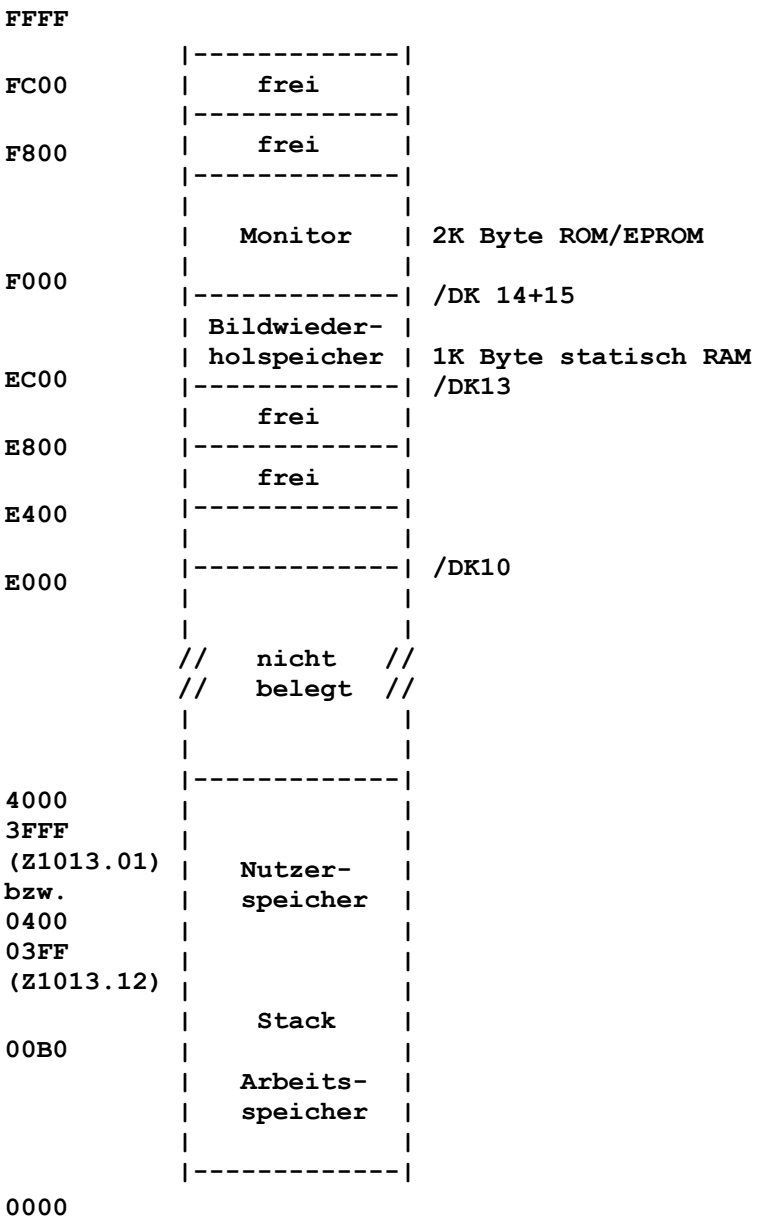
| | Frage, ob | gesetzt | nicht gesetzt | wird gesetzt bei |
|---|---|---|---|---|
| C | Carry-Flag | C | NC | Uebertrag von Bit 7 |
| N | Add-/Subtract-Flag | | | Subtraktion |
| P/V | Parity-/Overflow-Flag | PE | PO | geraderParitaet |
| H | Half-Carry-Flag | | | Uebertrag von Bit3 |
| Z | Zero-Flag | Z | NZ | Ergebnis 0 |
| S | Sign-Flag | M | P | neg. Ergebnis |
| X | nicht verwendet | | | |

Beeinflussung der Flags:

1 gesetzt
0 zurueckgesetzt
* abhaengig vom Ergebnis einer Operation
- nicht beeinflusst
? unbestimmt

## Anlage 2: Speicherbereichsaufteilung

**Speicher:**




## Anlage 3: E/A Adressen

**E/A-Adressen:**

| | | | | |
|---|---|---|---|---|
| 0000 | /IOSEL0 | PIO | PORT A | Daten |
| 01 | | | | Steuerwort |
| 02 | | | PORT B | Daten |
| 03 | | | | Steuerwort |
| 0004 | /IOSEL1 | | | |
| 05 | | | | |
| 06 | | | | |
| 07 | | | | |
| 0008 | /IOSEL2 | Tastaturspalten-Treiber | | |
| 09 | | | | |

## Anlage 4: Arbeitszellen des Monitors

| Adresse | Laenge | Bedeutung |
|---|---|---|
| 0000 | 3 | RST 0H, frei fuer den Anwender |
| 0003 | 1 | Zwischenspeicher f. Spezifikationsbyte RST 20H |
| 0004 | 1 | Merkzelle fuer letztes Zeichen von Tastatur |
| 0005 | 3 | freu fuer Anwendung eines zentralen CALL 5 |
| 0008 | 3 | RST 8H, frei fuer den Anwender |
| 000B | 2 | Zwischenspeicher fuer BREAK-Adresse |
| 000D | 3 | Zwischenspeicher fuer Operandenfolge bei BREAK |
| 0010 | 3 | RST 10H, frei fuer den Anwender |
| 0013 | 3 | Arbeitszellen fuer INHEX-Routine |
| 0016 | 2 | (SOIL) Anfangsadresse der Eingabezeile |
| 0018 | 3 | RST 18H, frei fuer den Anwender |
| 001B | 2 | (ARG1) Parameter 1 |
| 001D | 2 | (ARG2) Parameter 2 |
| 001F | 1 | Code-Zwischenspeicher fuer OUTCH |
| 0020 | 3 | RST 20H, zentr. Ansprung f. Monitorroutinen |
| 0023 | 2 | (ARG3) Parameter 3 |
| 0025 | 2 | 2. Adresse der Eingabezeile |
| 0027 | 1 | Merkzelle ASCII(=0)/Grafik(=80H) |
| 0028 | 3 | RST 28H, frei fuer den Anwender |
| 002B | 2 | (CURSR) aktuelle Cursorposition |
| 002D | 1 | Cursor-Zwischenspeicher |
| 002F | 1 | Merkzelle fuer Phasenlage bei CLOAD |
| 0030 | 3 | RST 30H, frei fuer den Anwender |
| 0033 | 2 | Laenge der Synchronisationsluecke bei CSAVE |
| 0035 | 2 | Beginn Tastencodetabelle |
| 0038 | 3 | RST 38H, wird als zentraler Fehleransprung verwendet, bei eintritt in den Monitor erscheint ?# |
| 003B | 12 | Fortsetzung Tastencodetabelle |
| 0047 | 2 | Rolldistanz bei OUTCH |
| 0049 | 2 | Anfangsadresse des BS Rollbereiches |
| 004B | 2 | Endadresse+1 des Rollbereiches |
| 004D | 24 | Registerrettbereich |
| 0066 | 3 | NMI, frei fuer den Anwender |
| . | | |
| . | | |
| 0090 | >32 | Anwender-Stackbereich (Stack laeuft nach unten!) |
| . | | |
| . | | |
| 00B0 | >=32 | System-Stackbereich (Stack laeuft nach unten!) |
| 00B0 | <=48 | fei füer Kommandoerweiterungstabelle des Monitors, welche ueber @... erreichbar ist |
| 00E0 | 32 | Kassettenueberspielbereich |

## Anlage 5: BASIC-Befehlsliste

**Kommandos:**

| | | |
|---|---|---|
| BYE | B. | Verlassen BASIC |
| CLOAD | CL. | Laden von Kassette |
| CSAVE | CS. | Laden auf Kassette |
| LIST | L. | Auflisten BASIC-Programm |
| NEW | N. | Loeschen BASIC-Programm |
| RUN | R. | Start |

**Befehle:**

| | | |
|---|---|---|
| ABS | A. | absoluter Betrag |
| BYTE | BYT. | Ausgabe hexadezimal 8-Bit-Wert |
| CALL | C. | Aufruf Maschinenunterprogramm |
| FOR | F. | Schleifenbeginn |
| GOSUB | GOS. | Aufruf BASIC-Unterprogramm |
| GOTO | G. | Sprungbefehl |
| HEX | H. | Umwandlung hexadezimal |
| I$ | | Eingabe Zeichenkette |
| IF | | Bedingungsabfrage |
| IN | | Eingabe von Maschinenport |
| INCHAR | INC. | Eingabe Zeichen von Tastatur |
| INPUT | INP. | Eingabe Zahl |
| LEN | LE. | Pseudovariable, enthaelt Laenge der zuletzt eingegeben Zeichenkette |
| NEXT | N. | Schleifenende |
| O$ | | Ausgabe Zeichenkette |
| OUT | | Ausgabe auf Maschinenport |
| OUTCHAR | OUTC. | Ausgabe Zeichen |
| PEEK | PE. | direkter Speicherzugriff |
| POKE | PO. | direkter Speicherzugriff |
| PRINT | PR. | Ausgabe |
| REM | | Kommentarkennzeichen |
| RETURN | RE. | Rueckkehr vom BASIC-Unterprogramm |
| RND | RN. | Zufallsgenerator |
| SIZE | S. | Pseudovariable, enthaelt Ausgabe ueber verfuegbaren Speicher |
| STEP | STE. | Schrittweite |
| STOP | STO. | Programmende |
| TAB | T. | Ausgabe Zwischenraum |
| TO | | Festlegung Schleifenendwert |
| TOP | | Pseudovariable, erster freier Speicher-platz |
| WORD | W. | Ausgabe hexadezimal 16-Bit-Wert |

**Arithmetische Operationen:**

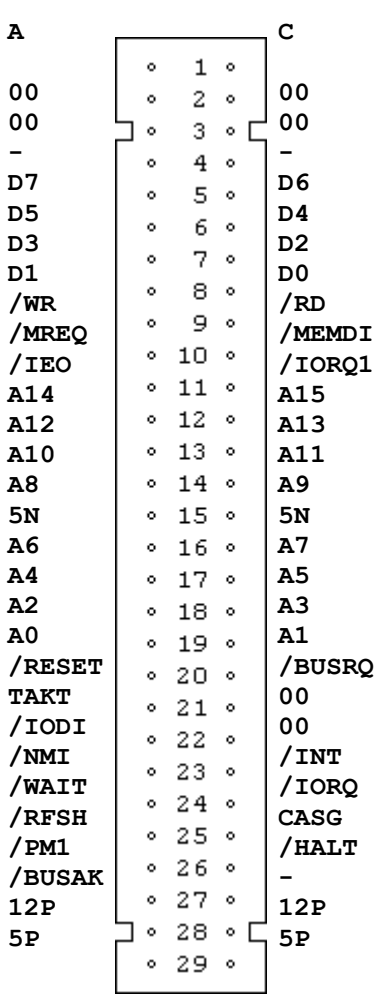
| | |
|---|---|
| + | Addition |
| - | Subtraktion |
| / | Division (ganzzahlig) |
| * | Multiplikation |

**Vergleichsparameter:**

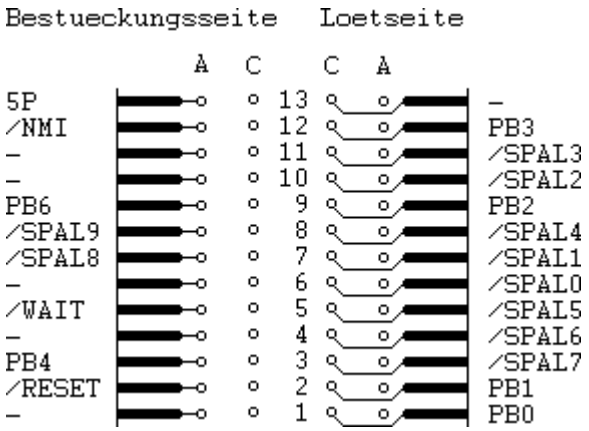
| | |
|---|---|
| >= | groesser gleich |
| # | ungleich |
| > | grosser |
| = | gleich |
| <= | kleiner gleich |
| < | kleiner |

## Anlage 6: Steckverbinderbelegung

**X1 Systemsteckverbinder**

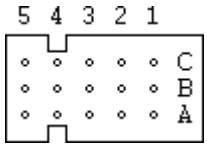



**X2 Loetkamm fuer Folienflachtastatur**



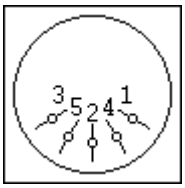

**X4 Anwendertor**

| | A | B | C |
|---|---|---|---|
| 1 | 5P | /ASTB | BRDY |
| 2 | PA6 | PA7 | ARDY |
| 3 | PA3 | PA4 | PA5 |
| 4 | PA0 | PA1 | PA2 |
| 5 | 00 | 00 | /BSTB |




**X5 Magnetbandanschluß**

1 Ausgabe
2 Masse
3 Eingabe
4 Ausgabe
5 Eingabe

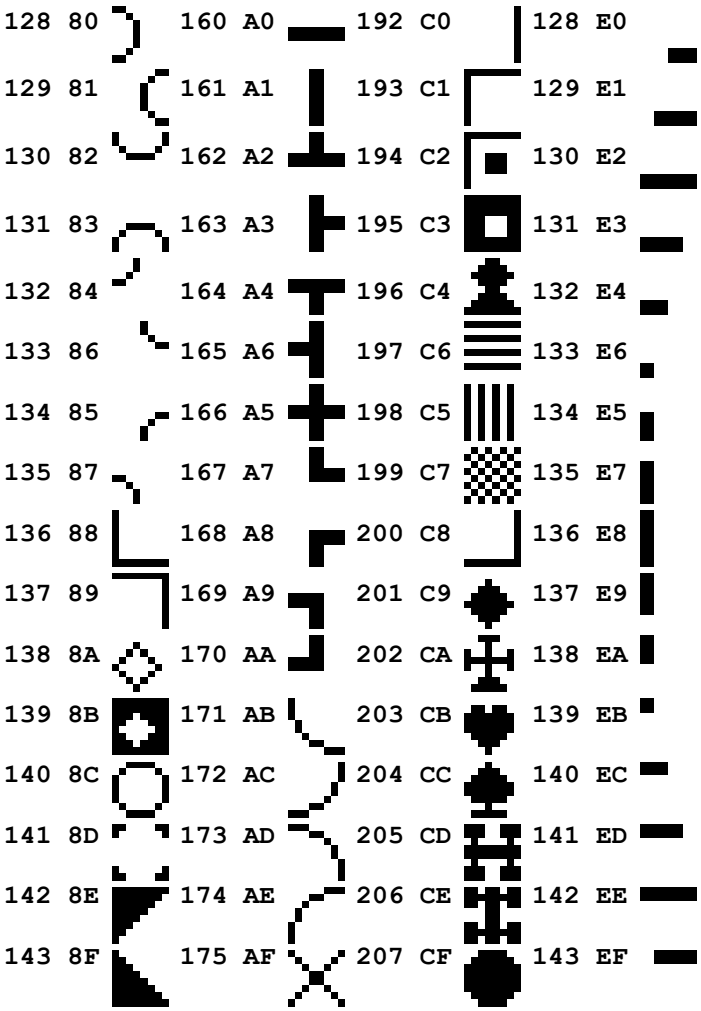


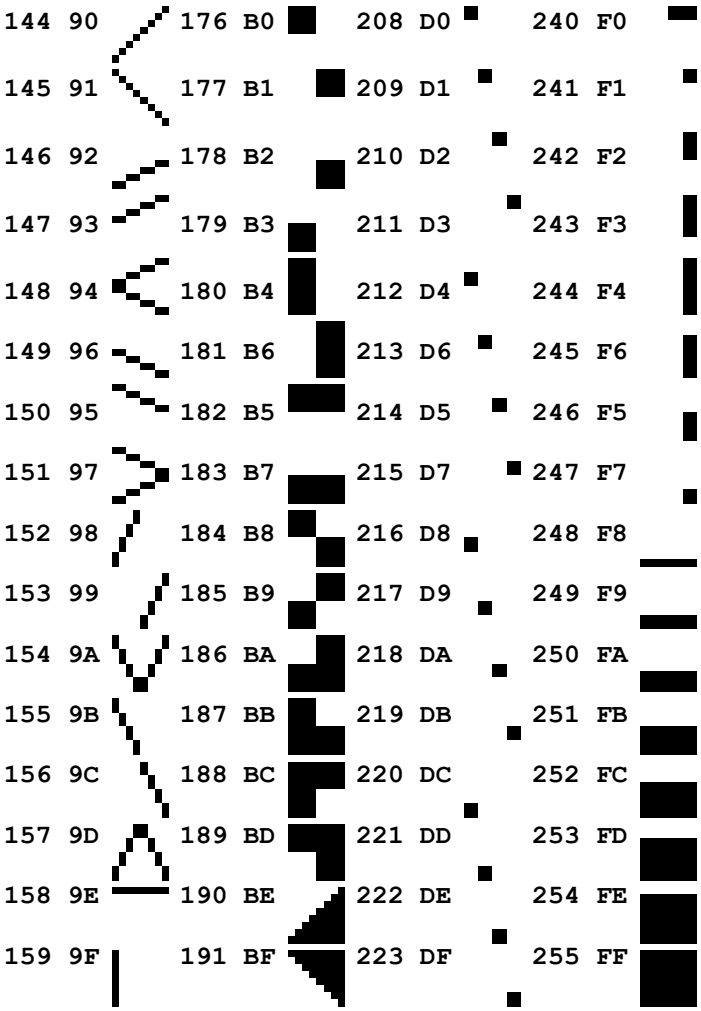

## Anlage 7: Zeichensatz/ASCII-Code

| Dez. | Hex. | Zeichen | Dez. | Hex. | Zeichen |
|---|---|---|---|---|---|
| 32 | 20 | Space | 82 | 50 | P |
| 33 | 21 | ! | 83 | 51 | Q |
| 34 | 22 | " | 84 | 52 | R |
| 35 | 23 | # | 85 | 53 | S |
| 36 | 24 | $ | 86 | 54 | T |
| 37 | 25 | % | 87 | 55 | U |
| 38 | 26 | & | 88 | 56 | V |
| 39 | 27 | ' | 89 | 57 | W |
| 40 | 28 | ( | 90 | 58 | X |
| 41 | 29 | ) | 91 | 59 | Y |
| 42 | 2A | * | 92 | 5A | Z |
| 43 | 2B | + | 93 | 5B | [ |
| 44 | 2C | , (comma) | 94 | 5C | \ |

| | | | | | |
|---|---|---|---|---|---|
| 45 | 2D | - (minus) | 95 | 5D | ] |
| 46 | 2E | . (period) | 96 | 5E | ^ |
| 47 | 2F | / | 97 | 5F | _ (underline) |
| 48 | 30 | 0 | 96 | 60 | ` |
| 49 | 31 | 1 | 97 | 61 | a |
| 50 | 32 | 2 | 98 | 62 | b |
| 51 | 33 | 3 | 99 | 63 | c |
| 52 | 34 | 4 | 100 | 64 | d |
| 53 | 35 | 5 | 101 | 65 | e |
| 54 | 36 | 6 | 102 | 66 | f |
| 55 | 37 | 7 | 103 | 67 | g |
| 56 | 38 | 8 | 104 | 68 | h |
| 57 | 39 | 9 | 105 | 69 | i |
| 58 | 3A | : | 106 | 6A | j |
| 59 | 3B | ; | 107 | 6B | k |
| 60 | 3C | < | 108 | 6C | l |
| 61 | 3D | = | 109 | 6D | m |
| 62 | 3E | > | 110 | 6E | n |
| 63 | 3F | ? | 111 | 6F | o |
| 64 | 40 | @ | 112 | 70 | p |
| 65 | 41 | A | 113 | 71 | q |
| 66 | 42 | B | 114 | 72 | r |
| 67 | 43 | C | 115 | 73 | s |
| 68 | 44 | D | 116 | 74 | t |
| 69 | 45 | E | 117 | 75 | u |
| 70 | 46 | F | 118 | 76 | v |
| 71 | 47 | G | 119 | 77 | w |
| 72 | 48 | H | 120 | 78 | x |
| 73 | 49 | I | 121 | 79 | y |
| 74 | 4A | J | 122 | 7A | z |
| 75 | 4B | K | 123 | 7B | { |
| 76 | 4C | L | 124 | 7C | \| |
| 77 | 4D | M | 125 | 7D | } |
| 78 | 4E | N | 126 | 7E | ~ |
| 79 | 4F | O | 127 | 7F | |

**ASCII-Code= American Standard Code of Information Interchange**
**Codierung der Grafiksymbole:**


128 80
129 81
130 82
131 83
132 84
133 86
134 85
135 87
136 88
137 89
138 8A
139 8B
140 8C
141 8D
142 8E
143 8F
160 A0
161 A1
162 A2
163 A3
164 A4
165 A6
166 A5
167 A7
168 A8
169 A9
170 AA
171 AB
172 AC
173 AD
174 AE
175 AF
192 C0
193 C1
194 C2
195 C3
196 C4
197 C6
198 C5
199 C7
200 C8
201 C9
202 CA
203 CB
204 CC
205 CD
206 CE
207 CF
128 E0
129 E1
130 E2
131 E3
132 E4
133 E6
134 E5
135 E7
136 E8
137 E9
138 EA
139 EB
140 EC
141 ED
142 EE
143 EF




**Schachfiguren:**

| | weiß D | H | schwarz D | H |
|---|---|---|---|---|
| Bauer | 23 | 17 | 14 | 0E |
| | 26 | 1A | 17 | 11 |
| Turm | 24 | 18 | 15 | 0F |
| | 26 | 1A | 17 | 11 |
| Springer | 25 | 19 | 16 | 10 |
| | 26 | 1A | 17 | 11 |
| Läufer | 27 | 1B | 18 | 12 |
| | 28 | 1C | 19 | 13 |
| Dame | 29 | 1C | 20 | 14 |
| | 31 | 1F | 22 | 16 |
| König | 30 | 1E | 21 | 15 |
| | 31 | 1F | 22 | 16 |

## Anlage 8: Bildschirmadressen

| Zeile | Spalte 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EC00 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EC20 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EC40 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EC60 | 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EC80 | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ECA0 | 5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ECC0 | 6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ECE0 | 7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ED00 | 8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ED20 | 9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ED40 | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ED60 | 11 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ED80 | 12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EDA0 | 13 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EDC0 | 14 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EDE0 | 15 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EE00 | 16 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EE20 | 17 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EE40 | 18 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EE60 | 19 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EE80 | 20 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EEA0 | 21 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EEC0 | 22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EEE0 | 23 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EF00 | 24 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EF20 | 25 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EF40 | 26 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EF60 | 27 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EF80 | 28 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EFA0 | 29 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EFC0 | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EEE0 | 31 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

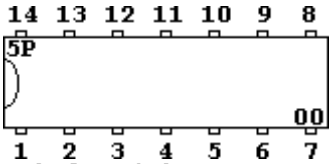
**Bildschirmbelegung: EC00 .... EFFF**

**Beispiel:** **Zeile 15, Spalte 28**

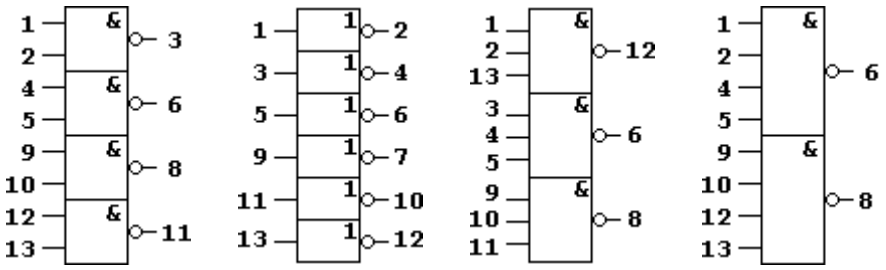
**Adresse: EDE0**
**+ 1C**
**EDFC**

## Anlage 9: Verwendete Schaltkreise

**Grundgatter**
**Gehäuse**



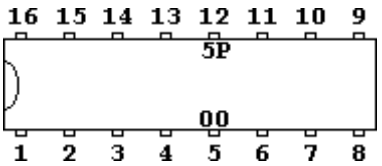

**Schaltzeichen**

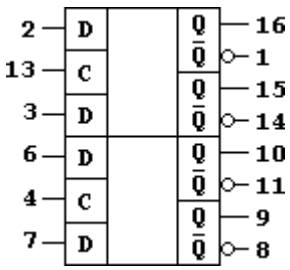



**D-Flip-Flop**
**Statisch gesteuerter D-FF D175**

**Schaltzeichen**

**Gehäuse**



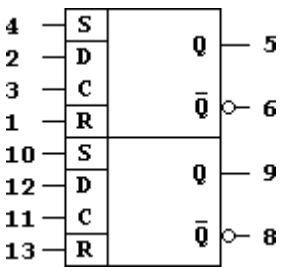

**Erläuterungen: C - Takteingang**
**D - Dateneingang**
**Q - Datenausgang**
**/Q - negierter Datenausgang**

**H-Pegel an C bewirkt Übernahme von D an Q.**
**Führt C L-Pegel dann führen Änderungen an D zu keinen Änderungen an Q.**
**Flankengesteuertes D-FF DL 074**

**Gehäuse: wie Grundgatter**

**Erläuterungen:**

**Schaltzeichen**




**S - Eingang Setzen: S=0 = Q=1**

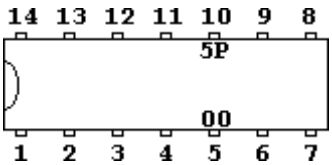
**R - Eingang Rücksetzen: R=0 = Q=0**

**C - Takt: - Flanke führt zur Übernahme der Information von D an Q (S=R=1)**

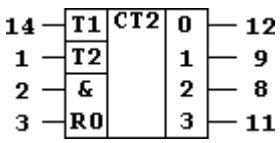
**D - Dateneingang**

**Zähler**
**Asynchroner 4 Bit Binärzähler DL 093**

**Gehäuse**



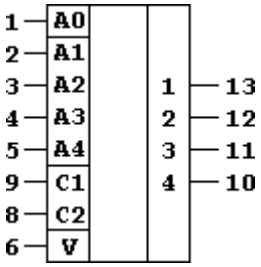

**Schaltzeichen**



Erläuterung:
T1 - H-L-Flanke schaltet Ausgang 0
T2 - Zähleingang für Ausgänge 1,2,3; schaltet mit
H-L-Flanke
H-Pegel an den Rückstelleingängen R0 setzt Ausgänge auf L

Schieberegister

4 Bit Rechts/Links- Schieberegister D 195
Gehäuse: wie Grundgatter
Schaltzeichen



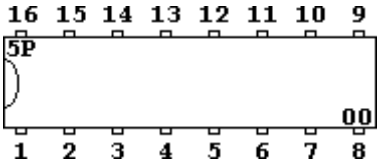

Erläuterung:
A0 - serieller Dateneingang
A1...A4 - parallele Dateneingänge (A4 - niederwertigstes Bit)
1...4 - paralle Datenausgänge ( analog A1...A4)
C1, C2 - Schiebetakteingänge
V - Steuereingang

| C1 | C2 | V | A0 | A1...A4 | |
|---|---|---|---|---|---|
| x | | 1 | x | Daten | Parallel Laden und Schieben |
| | x | 0 | Daten | x | Serielles Schieben |

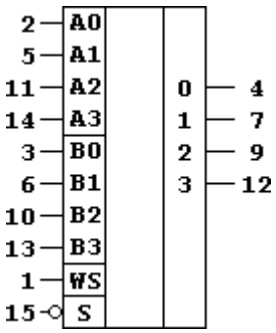
Multiplexer

4facher 2 zu 1 Multiplexer DL257 (K555 KP 11)

Gehäuse

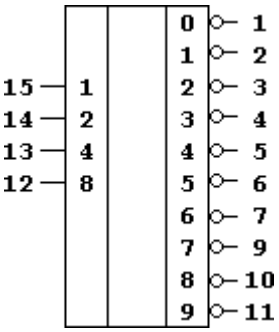



Schaltzeichen




Erläuterung:
A0...A3 Eingänge Wert 1
B0...B3 Eingänge Wert 2
WS Wortauswahl: 0 Wort 1 an Ausgängen 0...3
1 Wort 2 an Ausgängen 0...3
S Steuereingang: 0 Freigabe des Multiplexers
1 Ausgänge hochohmig

**Dekoder**
**BCD zu Dezimal Dekoder MH7442**
**Gehäuse: wie DL 257**
**Schaltzeichen**



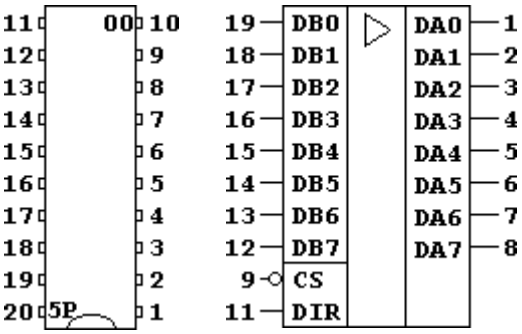

**Erläuterung:**

| Eingänge | | | | Ausgänge | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 8 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| L | L | L | L | L | H | H | H | H | H | H | H | H | H |
| L | L | L | H | H | L | H | H | H | H | H | H | H | H |
| L | L | H | L | H | H | L | H | H | H | H | H | H | H |
| L | L | H | H | H | H | H | L | H | H | H | H | H | H |
| L | H | L | L | H | H | H | H | L | H | H | H | H | H |
| L | H | L | H | H | H | H | H | H | L | H | H | H | H |
| L | H | H | L | H | H | H | H | H | H | L | H | H | H |
| L | H | H | H | H | H | H | H | H | H | H | L | H | H |
| H | L | L | L | H | H | H | H | H | H | H | H | L | H |
| H | L | L | H | H | H | H | H | H | H | H | H | H | L |
| H | L | H | L | H | H | H | H | H | H | H | H | H | H |

...

**Bustreiber**
**8 Bit bidirektionaler Bustreiber DS 8286**

**Gehäuse** **Schaltzeichen**

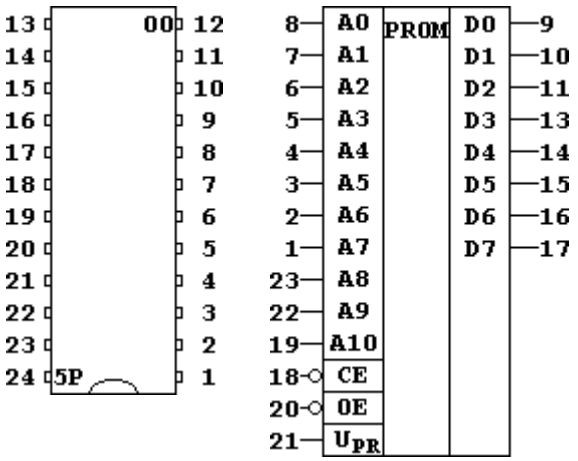



**Erläuterung:**

| /CS | DIR | |
|---|---|---|
| 0 | 0 | Daten von DB nach DA |
| 0 | 1 | Daten von DA nach DB |
| 1 | x | DA und DB hochohmig |

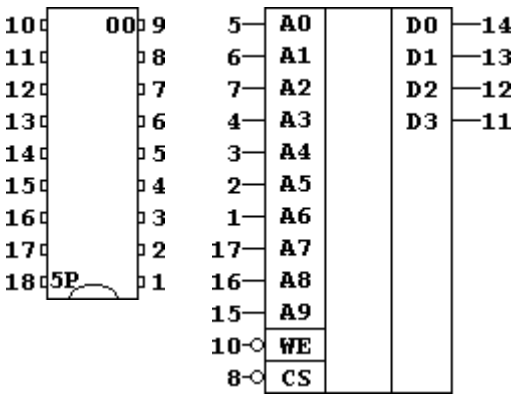
**Schpeicherschaltkreise**
**PROM U2616**

**Gehäuse** **Schaltzeichen**




**Erläuterung:**
**A0...A10 : Adresseingänge**
**D0...D7 : Datenausgänge**
**/CE : L-Pegel aktiviert PROM**
**/OE : L-Pegel gibt Ausgänge frei**
**$U_{PR}$ : Programmiereingang**
**Statischer RAM U2114**

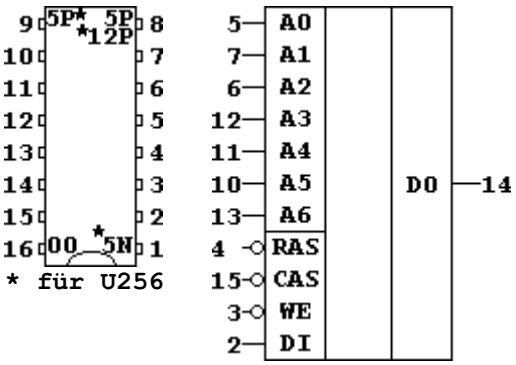
**Gehäuse** **Schaltzeichen**




**Erläuterung:**
**A0...A9 : Adresseingänge**
**D0...D3 : Datenausgänge**
**/WE : 0 - Daten schreiben**
**: 1 - Daten lesen**
**/CS : Busfreigabe**

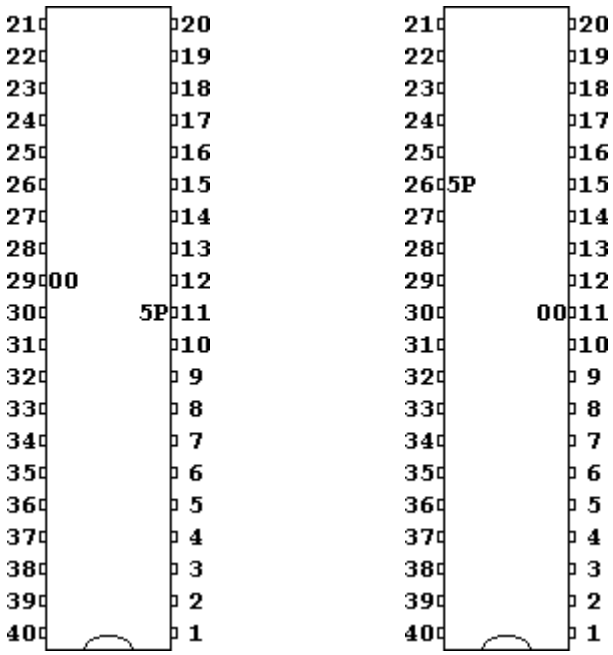
**Statischer RAM U256 (K565 RU3) und K565 RU6**

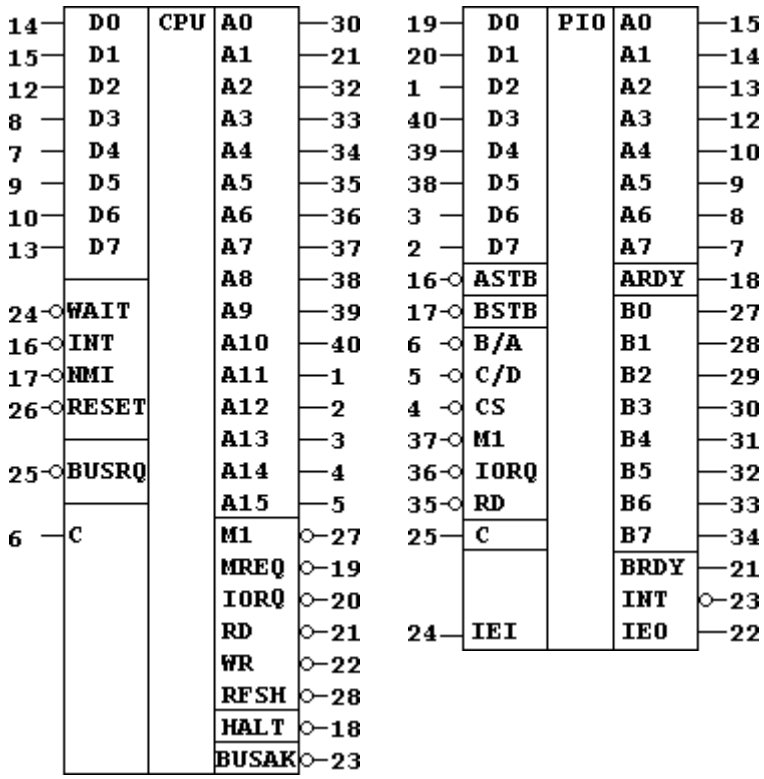
**Gehäuse** **Schaltzeichen**




**Erläuterung:**
**A0...A6 : Adresseingänge**
**DI : Dateneingang**
**DO : Datenausgang**
**RAS/CAS : s. Abschnitt 3.3**
**WE : s. U2114**

**Mikroprozessor** **Parallel E/A**




CPU
D0
D1
D2
D3
D4
D5
D6
D7
WAIT
INT
NMI
RESET
BUSRQ
C
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
M1
MREQ
IORQ
RD
WR
RFSH
HALT
BUSAK
14
15
12
8
7
9
10
13
24
16
17
26
25
6
30
21
32
33
34
35
36
37
38
39
40
1
2
3
4
27
19
20
PIO
ASTB
BSTB
B/A
C/D
CS
M1
IORQ
RD
C
IEI
A0
A1
A2
A3
A4
A5
A6
A7
ARDY
B0
B1
B2
B3
B4
B5
B6
B7
BRDY
INT
IEO
1
40
39
38
3
2
16
17
6
5
4
37
36
35
25
24
15
14
13
12
10
9
8
7
18
27
28
29
30
31
32
33
34
21
23
22