Handbuch Teil 1

R O B O T R O N

Mikrorechnerbausatz Z 1 0 1 3

Handbuch Teil 1

Inhaltsverzeichnis

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 Schalt- kreise, 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 Pro- gramme, die die Zusammenarbeit der einzelnen Systemkomponenten organisieren bzw. ueberhaupt ermeeglichen. Worin unter- scheidet sich aber nun ein Mikrorechner von einer herkoemlichen 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 be- reits 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 Verknuep- fungsmoeglichkeiten 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 Anwei- sungen 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 dis- kreten 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 allenangeschlossenen 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 zuruecktransportiert 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 Anzahlnder 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 Aufbewah- rung 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 OutputPIO) 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 ausgeaehlten 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 folgenden 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 Informa- tionen 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 Faecher bereitgestellt.

Wenn Informationen sowohl gelesen als auch abgelegt 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:

```
                n         n-1            1         0
      Z = d * x  + d * x   +...+ d * x  + d * x
```

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.

```
               2        1        0
   123 = 1 x 10 + 2 x 10 + 3 x 10          (dezimal)
             6       5       4       3       2
       = 1 x 2 + 1 x 2 + 1 x 2 + 1 x 2 + 0 x 2 +
             1       0
         1 x 2 + 1 x 2
       = 11111011B                         (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.

```
123 : 2 = 61                Rest: 1--------------------
   -------|                                           |
   |                                                  |
   V                                                  |
 61 : 2 = 30                      1------------------- |
   -------|                                         | |
   |                                               | |
   V                                               | |
 30 : 2 = 15                      1----------------- | |
   -------|                                        | | |
   |                                              | | |
   V                                              | | |
 15 : 2 =  7                      1--------------- | | |
   -------|                                       | | | |
   |                                             | | | |
   V                                             | | | |
  7 : 2 =  3                      1------------- | | | |
   -------|                                     | | | | |
   |                                           | | | | |
   V                                           | | | | |
  3 : 2 =  1                      1----------- | | | | |
   -------|                                   | | | | | |
   |                                         | | | | | |
   V                                         | | | | | |
  1 : 2 =  0                      1--------- | | | | | |
                                           | | | | | | |
                                          V V V V V V V V
                                 Binaer:  1 1 1 1 0 1 1  B
                                          ==============
```

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:

```
                                          Wertigkeit oder
  +---------------------------------------+  Exponent zur
  |   7 | 6  |  5 |  4 |  3 |  2 |  1  |  0 |  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          |    l           | 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. Zahl    |  Ergebnis
-------------------------------------
   0       |     0       |  1
   0       |     1       |  1
   1       |     0       |  1
   1       |     1       |  0
```

    Sinnbild:

  - **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. Zahl    |  Ergebnis
-------------------------------------
   0       |     0       |  0
```

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

Sinnbild:

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

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

Sinnbild:

- **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 Vorzeichenstelle ein-, aber kein Uebertrag aus der Vorzeichenstelle 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 ein 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:

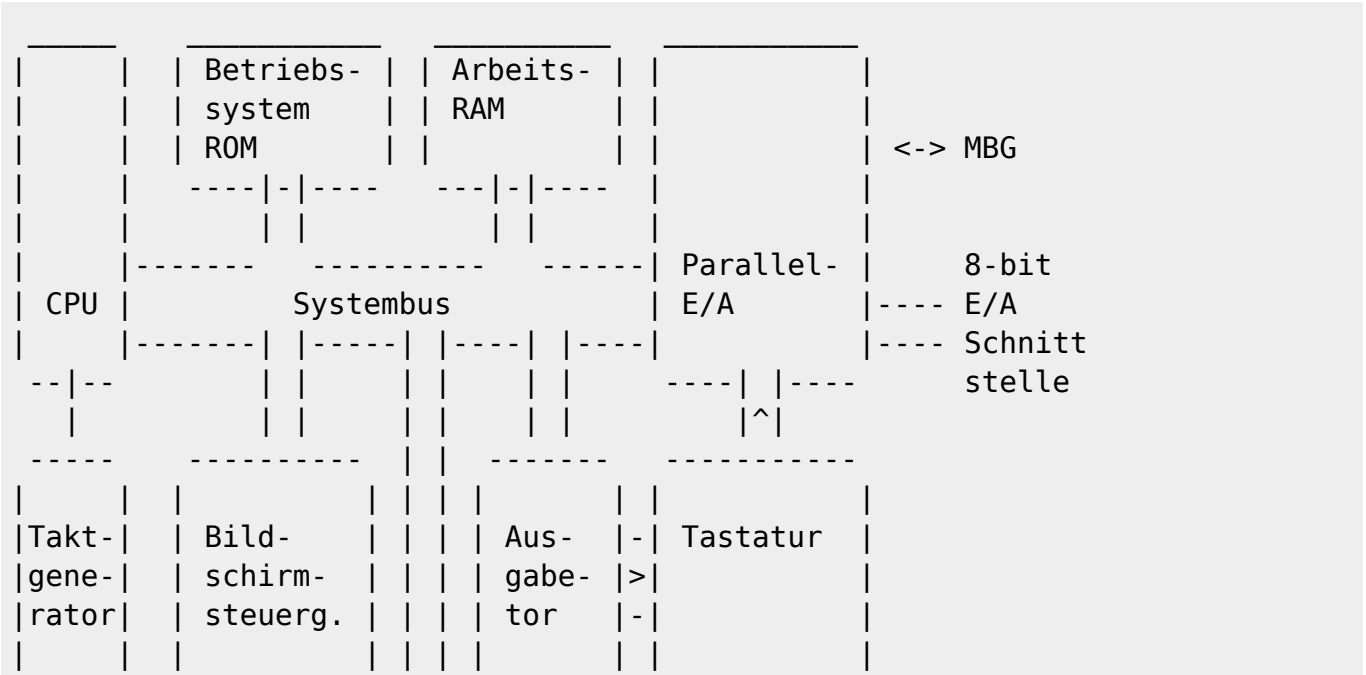
```
 Ausgangswerte:     0 0 0 0 1 0 0 1  *  0 1 0 1
 ---------------------------------------------
                    0 0 0 0 1 0 0 1  |  1
                  0 0 0 0 0 0 0 0    |  0 = 5
                0 0 0 0 1 0 0 1      |  1
              0 0 0 0 0 0 0 0        |  0
 ---------------------------------------------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



```
-----    ----------  | |  -------   -----------
             |       | |
         BAS |       | |
       |-----|       | |
       |  ____|__     | |
       | | HF-   |    | |
       | | Modu- |    | |
       | | lator |    | |
       |  -------     | |
       |     |        | |
     TV-Geraet     Systemsteckverbinder
```

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-Information 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, 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 erforderlich hohes 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 nichtvon 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) dernaechstfolgenden 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 verknuepft 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 Adressbe- reichen 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 entnom- men 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 Steuerwoerter. 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 Ml 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 Funk- tionen 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).

```
          _________________          _________________
         | E/A-Baugruppe |         | E/A-Baugruppe |
         |       1       |         |       2       |
5P -----| IEI       IEO |---------| IEI       IEO |---->
         ----------------           ----------------
```

  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  ==>IOSEL0, 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. **Interruptsteuerwert**

```
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 erfuel-
                    len
           0  , Interrupt freigeben
           1  , Interrupt gesperrt
```

5. **Interruptmaskenwort**
   Die Bitzelle 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 Spatennummer 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 ge- nutzt. 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 zurueckgewonnen.

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 fluereszierenden 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 Bildschirmsteuerung des MRB Z1013 arbeitet nach folgendem Prinzip:

Die gesamte Erzeugung des fernsehgerechten Signals, des sogenannten BAS-Signals, wird durch die Zaehlkaskade ohne Mitarbeit 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 nacheinander die Bildpunktzeilen an den nachfolgenden Parallel/Serien-Wandler A21/22 uebergeben. Hier wird das uebernommene Bitmuster 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 Synchronimpulse 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-Signal.

In einem HF-Modulator wird das BAS-Signal auf eine HF-Traegerfrequenz, die auf den Fernsehkanal 3 abgestimmt ist, aufmoduliert. Der Ausgang dieses Modulators kann nun direkt mit dem Antenneneingang des Fernsehgeraetes verbunden werden.

Wie gelangen aber nun in diese selbstaendig arbeitende Einheit 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 normaler 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 Span- nungen 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 benoetig- ten 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)

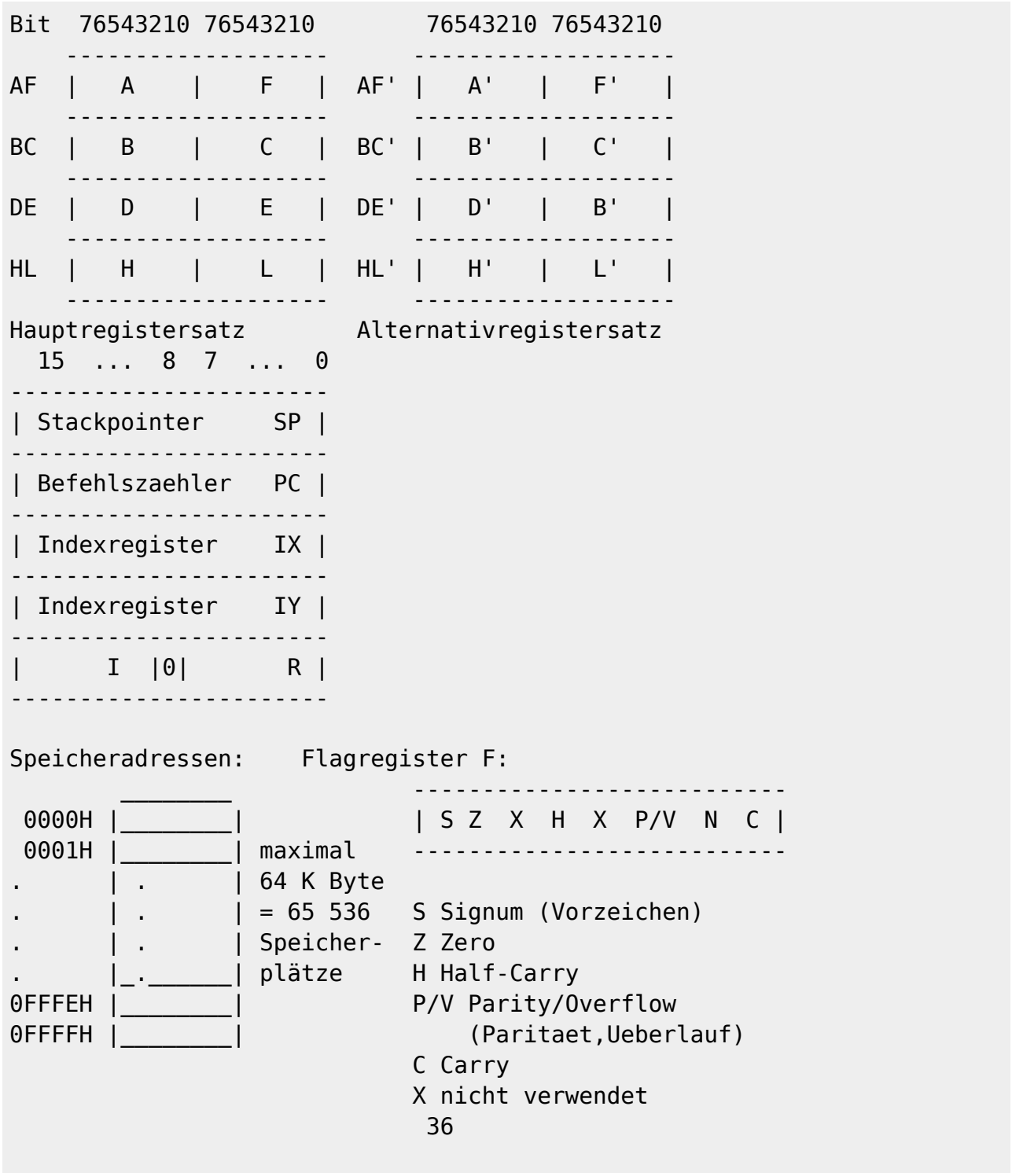
Die genaue Zuordnung der einzelnen Signale zu den jeweiligen Anschluessen ist der Anlage 6 zu entnehmen.

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.

<note tip> 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.</note>

```
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 Signalbytedar.

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
```

4.1.4. 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 Programmie- rung 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. Registerindirektadressierung

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 Doppel- register 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 er- halten. 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
100A     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 Doppelbytezaehlbefehle 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       Ml: 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 Ml,
                                   ;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- Maschinen-   Quellkode    Kommentar
zaehler  kode
----------------------------------------
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- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------
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- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------
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- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------
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:=0, Z:=0,
                      ;P:=1, A:=00H!!!
                              46
```

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 Hilfsoperationen

```
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 (Pseudoergebnisse). 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-Regiter 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

             |---------------------|
       __    |      _________      |
      |CY| <-+-----|7       0|<----|
       --           ---------

RL s       ; Linksverschieben ueber das Carry-Flag

|-----------------------------------|
|     __            _________       |
|<----|CY| <-------|7       0|<----|
       --           ---------

SLA s      ; Linksverschieben durch Carry-Flag


      __            _________
      |CY| <-------|7       0|<----0
       --           ---------

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

      |---------------------|
      |      _________      |     __
      |----->|7       0|-----+-----|CY|
             ---------            --

RR s       ; Rechtsverschiebung ueber das Carry-Flag

      |-----------------------------------|
```

```
      |       _________              __     |
      |----->|7       0|---------->|CY|--->|
             ---------              --

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


             _________              __
       ----->|7       0|---------->|CY|
      |      |--------              --
       -------

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


             _________              __
      0----->|7       0|---------->|CY|
             ---------              --

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-Registers = 1 ist. Wenn das A-Register geloescht war, sollen keine Verschiebungen stattfinden.

```
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------------------
1000     B0           OR A         ;Z = 1, wenn A = 0
                                   ;ist
1001     28 0D        JRZ END-#    ;Abbruch des
                      ;Programms, wenn A = 0
1003     CB 3F    Ml: SRL A        ;Rechtsverschiebung
                                   ;durch CY
1005     30 FC        JRNC M1-#    ;Ruecksprung zu Ml
                      ;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 Arithmetik. Beispielsweise laesst sich der SRA-Befehl als Division durch 2 oder der SLA-Befehl als Multiplikation mit 2 einsetzen.

Ein Sonderfall der Verschiebungsoperationen ist die Verschiebung um vier Dualstellen.
Linksverschiebung um vier Dualstellen

```
RLD   ;    Der Befehl wirkt zwischen A-Register und dem durch
           HL indirekt adressierten Speicheroperanden.

            ____________________
           |                    |
  _________|__         _________v__
A:|7   4|3   0| (HL):|7   4|3   0|
   --------^--         --|-^---|--
           |              | |   |
            ------------    ---
```

Zahlenbeispiel:

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

Rechtsverschiebung um vier Dualstellen

```
RRD     ; der Befehl wirkt ebenfalls nur zwischen A und (HL)
            _____________   ___
           |             | |   |
  _________|__         __v_|___v__
A:|7   4|3   0| (HL):|7   4|3   0|
   --------^--         --------|--
           |                   |
            -------------------
```

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
```

DieBerechnung 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- Maschinen-   Quellkode    Kommentar
zaehler  kode
-----------------------------------------------------
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.

```
   Kellern---+         +-------> Entkellern
             |         |
             V         |
        +--------------------+
nn-6 -> |    4.   :    1.   |
nn-4 -> |    3.   :    2.   |
nn-2 -> |    2.   :    3.   |
nn   -> |    1.   :    4.   |
        +--------------------+
```

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 Unterprograrn
             ;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 Sprungadres- se. 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
```

<note tip>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 wei- terlaeuft. </note> 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 Adressierung 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 Inhalt des von Register C adressierten Kanals wird gelesen und danach entsprechend die Flags gesetzt.

Beispiel: Das Ausgaberegister mit der Adresse 10H soll definiert 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
                              58

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 wiederholende Gruppenoperationen) Diese Befehle basieren auf den einfachen Gruppenoperationen. Es werden lediglich die Testung der Flags und entspechende Ruecksprueng zur Wiederholung des Befehls automatisch ausgefuehrt. Allerdings ist bei den Ein- und Ausgabegruppenoperationen 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 Rettung 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 Zweier-
     komplementdarstellung.
     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 Doppelregister.

Beispiel:

```
Befehls- Maschinen-   Quellkode    Kommentar
zaehler  kode
-------------------------------------------------------
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 Programmuebersetzung.

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 Unterprogramm 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 maskierbare 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 Unter-
     brechungswunsch der Interruptquelle der naechste abzu-
     arbeitende Befehl vom Datenbus geholt und in den Pro-
     grammlauf 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 in-
     terruptanfordernde 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 entsprechen-
      de Programme aktiviert.

IM2:  Diese Betriebsart stellt fuer den Prozessor die lei-
      stungsfaehigste Unterbrechungsbehandlung dar. Man
      nennt sie auch Vektorinterrupt, weil die Organisation
      der Behandlungsroutinen ueber Zeiger auf einen Adress-
      vektor realisiert wird.
      Bei Unterbrechungsanforderung stellt jede Interrupt-
      quelle den fuer die gewuenschte Interruptroutine not-
      wendigen niederwertigen "Zeigerteil" durch Aussenden
      eines 8-Bit-Wertes auf den Datenbus bereit (sogenann-
      tes Vektorlesen).
      Dieser Wert bildet den niederwertigen Teil, der Inhalt
      des I-Registers den hoeherwertigen Teil einer Adresse
      einer bestimmten Speicherzelle. Diese und die nachfol-
```

```
      gende Speicherzelle beinhalten dann die Adresse des In-
      terruptbehandlunnsprogramms. Dies setzt aber natuerlich
      das vorbereitende Laden des I-Registers und der ent-
      sprechenden Register der Peripheriebausteine voraus
      (sogenannte Initialisierung).
```

Beispiel:

```
          3 Peripheriebausteine verlangen jeweils eine ande-
          re 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 entsprechen-
             de 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.
```