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 Befehls-adresse 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
                                  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.
```