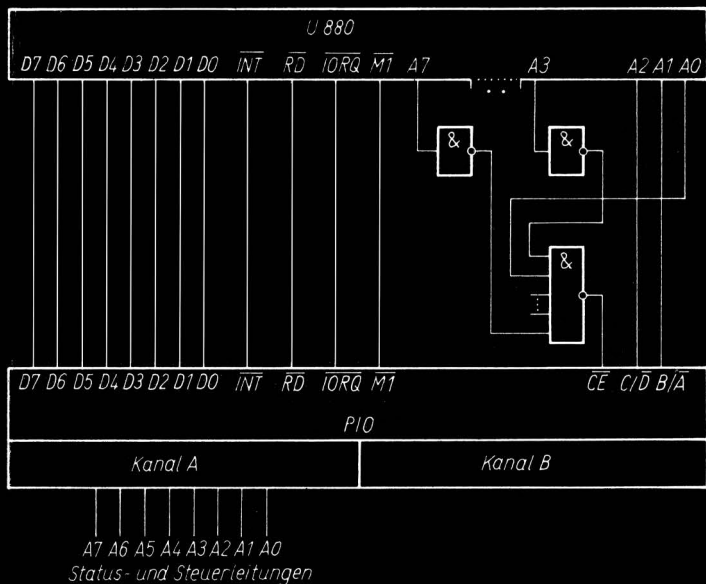


# amateurreihe electronica



**Barthold/Bäurich**

**Mikroprozessoren –  
Mikroelektronische  
Schaltkreise und ihre  
Anwendung (Teil 2)**

**224/  
225**

electronica · Band 224/225

HANS BARTHOLD  
DR. HEINZ BÄURICH

# **Mikroprozessoren – Mikroelektronische Schaltkreise und ihre Anwendung**

Teil 2:  
Periphere Schaltkreise / Programmbeispiele



MILITÄRVERLAG  
DER DEUTSCHEN DEMOKRATISCHEN  
REPUBLIK

Barthold, H.; Dr. Bäurich, H.;  
Mikroprozessoren – Mikroelektronische  
Schaltkreise und ihre Anwendung.  
Teil 2: Periphere Schaltkreise/Programmbeispiele. –  
3., überarbeitete Aufl., – Berlin:  
Militärverlag der DDR (VEB), 1985. –  
146 S.: 80 Bilder – (electronica: 224/225)

3. Auflage, 1985  
© Militärverlag der  
Deutschen Demokratischen Republik (VEB) – Berlin, 1980  
Lizenz-Nr. 5  
Printed in the German Democratic Republic  
Gesamtherstellung: Druckerei Märkische Volksstimme Potsdam  
Lektor: Steffen Würtenberger  
Zeichnungen: Johanna Goernemann, Angelika Ulsamer  
Typografie: Martina Schwarz  
Redaktionsschluß: 20. November 1984  
LSV 3539  
Bestellnummer: 7466932  
00380

# Inhaltsverzeichnis

|           |                                                                                                               |          |
|-----------|---------------------------------------------------------------------------------------------------------------|----------|
| <b>1.</b> | <b>Übersicht über verwendete Abkürzungen . . . . .</b>                                                        | <b>6</b> |
| <b>2.</b> | <b>Ein-/Ausgabe-Bausteine zum Prozessor <i>U 880 D</i> . .</b>                                                | <b>7</b> |
| 2.1.      | Grundfunktion der E/A-Bausteine . . . . .                                                                     | 7        |
| 2.2.      | Paralleler Ein-/Ausgabe-Baustein <i>U 855 D</i> . . . . .                                                     | 9        |
| 2.2.1.    | Struktur des <i>U 855 D</i> . . . . .                                                                         | 9        |
| 2.2.2.    | Beschreibung der Anschlüsse des <i>U 855 D</i> . . . . .                                                      | 11       |
| 2.2.3.    | Arbeitsweise des <i>U 855 D</i> . . . . .                                                                     | 13       |
| 2.2.4.    | Beispiel zum <i>U 855 D</i> . . . . .                                                                         | 17       |
| 2.3.      | Zeitgeberbaustein <i>U 857 D</i> . . . . .                                                                    | 19       |
| 2.3.1.    | Struktur des CTC-Bausteins . . . . .                                                                          | 19       |
| 2.3.2.    | Beschreibung der Anschlüsse des CTC-Bausteins . .                                                             | 21       |
| 2.3.3.    | Arbeitsweise des CTC-Bausteins . . . . .                                                                      | 22       |
| 2.4.      | Serieller Ein-/Ausgabe-Baustein <i>U 856 D</i> . . . . .                                                      | 24       |
| 2.4.1.    | Struktur des <i>U 856 D</i> . . . . .                                                                         | 24       |
| 2.4.2.    | Beschreibung der Anschlüsse des <i>U 856 D</i> . . . . .                                                      | 25       |
| 2.4.3.    | Aufbau eines Kanals des <i>U 856 D</i> . . . . .                                                              | 27       |
| 2.4.4.    | Arbeitsweise des SIO-Bausteins . . . . .                                                                      | 29       |
| 2.4.5.    | Interruptbildung und Statusbildung . . . . .                                                                  | 32       |
| 2.4.6.    | Beschreibung der Bit-Stellen der Steuerregister $WR_0$<br>bis $WR_7$ und der Statusregister $RR_0$ bis $RR_2$ | 37       |
| 2.5       | DMA-Steuerbaustein <i>U 880 – DMA</i> . . . . .                                                               | 43       |
| 2.5.1.    | Funktion und Struktur des DMA-Bausteins . . . . .                                                             | 43       |
| 2.5.2.    | Beschreibung der Anschlüsse des DMA-Bausteins .                                                               | 45       |
| 2.5.3.    | Arbeitsweise des DMA-Bausteins . . . . .                                                                      | 46       |
| 2.5.4.    | Beispiel zum DMA-Baustein . . . . .                                                                           | 51       |
| 2.6.      | Serieller Ein-/Ausgabeschaltkreis <i>8251</i> . . . . .                                                       | 54       |
| 2.6.1.    | Funktion des <i>8251</i> . . . . .                                                                            | 54       |
| 2.6.2.    | Anschlußbelegung . . . . .                                                                                    | 55       |
| 2.6.3.    | Logische Struktur des Bausteins <i>8251</i> . . . . .                                                         | 57       |
| 2.6.4.    | Arbeitsweise . . . . .                                                                                        | 58       |
| 2.6.5.    | Programmierung . . . . .                                                                                      | 58       |

|            |                                                                                              |     |
|------------|----------------------------------------------------------------------------------------------|-----|
| <b>3.</b>  | <b>Assemblersprache MAPS K 1520</b>                                                          | 60  |
| 3.1.       | Format einer Assembleranweisung                                                              | 60  |
| 3.2.       | Pseudoanweisungen der Sprache MAPS K 1520                                                    | 61  |
| 3.3.       | Makroorganisation                                                                            | 63  |
| 3.4.       | Gegenüberstellung der Schreibweisen von Befehlen<br>und Pseudobefehlen UDOS/MAPS K 1520      | 64  |
| <b>4.</b>  | <b>Programmaufbereitungssysteme</b>                                                          | 66  |
| 4.1.       | Schritte der Programmaufbereitung                                                            | 66  |
| 4.2.       | Systemprogramme zur Programmaufbereitung                                                     | 67  |
| <b>5.</b>  | <b>Basiskonfiguration eines Mikrorechners</b>                                                | 69  |
| 5.1.       | Übersicht                                                                                    | 69  |
| 5.2.       | Steuerung des Prozessors                                                                     | 70  |
| 5.3.       | Bedienelemente                                                                               | 72  |
| 5.4.       | Anzeige mit LED-Elementen                                                                    | 74  |
| 5.5.       | Programm zur LED-Anzeige ohne Konvertierung<br>und Zwischenspeicher                          | 77  |
| <b>6.</b>  | <b>Programmbeispiele</b>                                                                     | 82  |
| 6.1.       | Numerische Programme und Programme zur Zahlen-<br>umwandlung                                 | 82  |
| 6.2.       | Rechenprogramme                                                                              | 87  |
| 6.3.       | Programme mit peripheren Bausteinen                                                          | 108 |
| <b>7.</b>  | <b>Mikrorechnersoftware</b>                                                                  | 118 |
| <b>8.</b>  | <b>Schlußbetrachtung</b>                                                                     | 124 |
| 8.1.       | Einkartenrechner                                                                             | 125 |
| 8.2.       | Mikrorechnersystem                                                                           | 126 |
| 8.3.       | Mehrprozessorsysteme                                                                         | 128 |
| 8.4.       | Hierarchische Rechnersysteme (Rechnernetze)                                                  | 132 |
| <b>9.</b>  | <b>Literatur</b>                                                                             | 134 |
| <b>10.</b> | <b>Tabellenanhang</b>                                                                        | 136 |
| 10.1.      | Darstellung der Zahlen 0 bis 32 in dezimaler, dualer,<br>oktaler, hexadezimaler und BCD-Form | 136 |
| 10.2.      | Addition hexadezimaler Zahlen                                                                | 137 |
| 10.3.      | Multiplikation hexadezimaler Zahlen                                                          | 138 |

|       |                                                      |     |
|-------|------------------------------------------------------|-----|
| 10.4. | Umrechnung hexadezimal → dezimal . . . . .           | 138 |
| 10.5. | Umrechnung dezimal → hexadezimal . . . . .           | 139 |
| 10.6. | Umrechnung hexadezimal → oktal . . . . .             | 140 |
| 10.7. | Potenzen der Basis 16 . . . . .                      | 141 |
| 10.8. | Potenzen der Basis 10 in hexadezimaler Darstellung . | 142 |
| 10.9. | ASCII-Code/SIF-1000-Code . . . . .                   | 143 |

# 1. Übersicht über verwendete Abkürzungen

Nachstehend sind die verwendeten Formelzeichen und Abkürzungen aufgeführt. Spezielle Zeichen, die nur wenig benutzt werden, sind im Text erläutert.

|         |                                                                                       |
|---------|---------------------------------------------------------------------------------------|
| CE      | Chip Enable (Chip-Freigabe)                                                           |
| CLK     | Clock (Takt)                                                                          |
| CRC     | Cyclic Redundancy Check<br>(Zyklische Blockprüfung)                                   |
| CS      | Chip Select (Chip-Auswahl)                                                            |
| CTC     | Counter Timer Circuit (Zähler-Zeitgeber-Baustein)                                     |
| DI      | Disable Interrupt (Sperren Interrupt)                                                 |
| DMA     | Direct Memory Access (Baustein für direkten Speicherzugriff – Direkter Speicherkanal) |
| EI      | Enable Interrupt (Freigabe Interrupt)                                                 |
| HWT     | Höherwertiger Teil von 2 Byte                                                         |
| IEI     | Interrupt Enable Input (Interruptfreigabeeingang)                                     |
| IEO     | Interrupt Enable Output (Interruptfreigabe-Weitergabesignal)                          |
| INT     | Interrupt (Programmunterbrechung)                                                     |
| NWT     | Niederwertiger Teil von 2 Byte                                                        |
| PIO     | Parallel Input/Output (Paralleler Ein-/Ausgabe-Baustein)                              |
| RDY     | Ready (Bereitschaft)                                                                  |
| RR      | Read Register (Statusregister im SIO)                                                 |
| Sgn (p) | Sign (p) = Vorzeichen von p (1, wenn p positiv; -1, wenn p negativ)                   |
| SIO     | Serial Input/Output (Serieller Ein-/Ausgabe-Baustein)                                 |
| STB     | Strobe (Tönungsimpuls)                                                                |
| WR      | Write Register (Steuerregister im SIO)                                                |

## 2. Ein-/Ausgabe-Bausteine zum Prozessor U 880

### 2.1. Grundfunktion der E/A-Bausteine

Ein- und Ausgabe-Bausteine bilden das Interface zum Anschluß von peripheren Geräten an den Rechnerbus.

Bild 2.1 zeigt das Prinzip des Anschlusses von einem Bedienpult, eines Lochbandlesers, eines Magnetbandgerätes und eines Lochbandstanzers an den Prozessor U 880. Auf die gleiche Weise lassen sich auch AD-Wandler, DA-Wandler, Drucker sowie Bildschirm-einheiten steuern.

Der Ein-/Ausgabe-Baustein erzeugt aus den Signalen des Rechnerbusses die Signale, die zur Steuerung peripherer Geräte erforderlich sind. Das periphere Gerät sendet Fertigmeldung und Fehlermeldungen dem E/A-Baustein, in dem diese Meldungen gesammelt und an den Rechnerbus weitergegeben werden. Die CPU kann solche Meldungen am peripheren Baustein abfragen (Polling). Eine zweite Möglichkeit besteht in der Bildung einer Interruptanforderung an die CPU. Zur Interruptbildung können die E/A-Bausteine der U 880-Familie Interruptketten bilden.

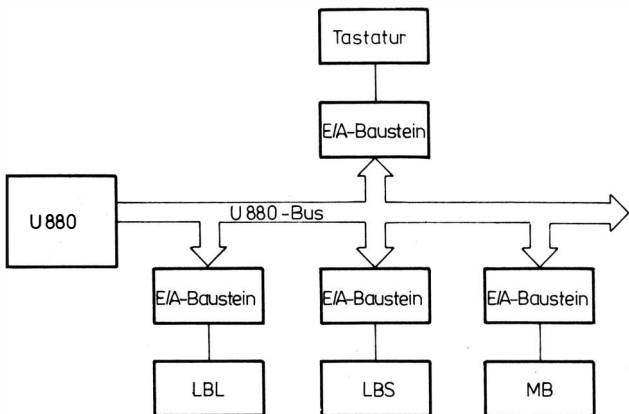


Bild 2.1 Anschluß von Bedienpult, Lochbandleser, Lochbandstanzer und Magnetbandgerät an den Prozessor U 880

## Struktur einer Interruptkette

Jeder E/A-Baustein hat die Anschlüsse IEI, IEO,  $\overline{\text{INT}}$ . Wenn IEI (interrupt enable input) H-Pegel führt, so kann der Baustein ein Interruptsignal bilden. IEO (interrupt enable output) ist «High», wenn IEI auf H-Pegel liegt und keine Interruptanforderung vom Baustein selbst kommt. Mit den beiden Signalen IEI und IEO lassen sich die Bausteine in einer Kette nach Bild 2.2 zusammenstellen. Der dem Prozessor am nächsten liegende Baustein hat die höchste Priorität.

Der folgende Interruptablauf (bezogen auf Bild 2.2.) zeigt die Arbeitsweise der einzelnen Bausteine, damit die Interruptkette alle Prioritätsfunktionen erfüllt.

- Baustein B3 bildet eine Interruptanforderung (die CPU ist in der Betriebsart IM2).
- INT wird aktiv (INT-Leitung = «Low»).
- IEO von Baustein B0 wird «Low». Damit werden Interruptanforderungen von Baustein B4 und den weiter rechts liegenden Bausteinen gesperrt.
- Ist der Interrupteingang der CPU offen, erfolgt Interruptannahme. Die CPU führt einen Interruptzyklus (mit IORQ und M1) aus.
- Während des Interruptzyklus kann Baustein B3 (wenn nicht vorher B1 oder B2 eine Interruptanforderung gebildet haben) seinen Interruptvektor zur CPU geben. (Sonst gibt der Baustein seinen Interruptvektor, der eine Interruptanforderung gestellt hat und der CPU am nächsten liegt.)
- Mit der Aufnahme des Interruptvektors durch die CPU erfolgt der Sprung in die zugehörige Behandlungsroutine und das Sperren des Interrupteingangs der CPU.

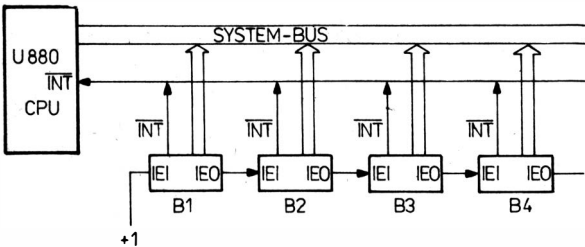


Bild 2.2 Zusammenschaltung von peripheren Bausteinen B1 bis B4 zu einer Interruptprioritätenkette

- Am Ende der Behandlungsroutine kommen im allgemeinen die Befehle EI und RETI. Durch EI wird der CPU-Interrupteingang wieder frei. RETI wird vom Baustein B3 erkannt. B3 setzt nun IEO wieder auf «High» und gibt damit den Rest der Kette frei.
- Ein besonderer Fall tritt dann ein, wenn während der Behandlungsroutine z. B. B2 eine Interruptanforderung stellt; IEO von B2 wird «Low» und B3 wäre gesperrt. Damit B3 den Abschluß RETI seiner Routine erkennt, setzt B2 mit dem 1. Byte von RETI (RETI hat die Codierung ED 4D) sein IEO auf «High» und B3 kann beide Bytes von RETI entschlüsseln.

## 2.2. Paralleler Ein-/Ausgabe-Baustein U 855 D

### 2.2.1. Struktur des U 855 D

Bild 2.3 zeigt die Grundstruktur des PIO-Bausteins. Der PIO-Baustein hat die Aufgabe, das Rechnerinterface (Rechnerbus) in ein Interface zur Steuerung peripherer Geräte mit paralleler Da-

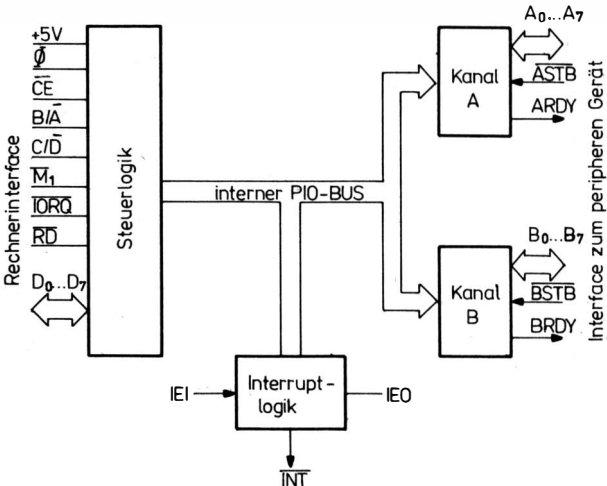


Bild 2.3 Grundstruktur des Bausteins U 855 D

teineingabe bzw. -ausgabe umzusetzen. Der Baustein verfügt über eine interne Steuerlogik, die mit Hilfe des Grundtaktes  $\Phi$  arbeitet, sowie eine Interruptsteuerung mit der Möglichkeit zur Bildung von Prioritätsketten. Die Steuerlogik erzeugt aus den Signalen des Rechnerbusses die zur Peripheriesteuerung notwendigen Signale. Zur Peripheriesteuerung gibt es 2 Ein-/Ausgabe-Kanäle A und B, die im *Hand-shake*-Verfahren den Transfer zu einem peripheren Gerät realisieren. Jeder Kanal hat

- 1 Eingaberegister (8 Bit),
- 1 Ausgaberegister (8 Bit),
- 1 Selektregister (Auswahlregister; 8 Bit),
- 1 Maskenregister (8 Bit),
- 1 Maskensteuerungsregister (2 Bit),
- 1 Betriebsartenregister (2 Bit).

Bild 2.4 zeigt die logische Struktur eines Kanals. Ein Kanal kann in folgenden Betriebsarten arbeiten:

- byteweise Ausgabe,
- byteweise Eingabe,
- byteweise bidirectional (über dieselbe Leitung können Informationen ein- bzw. ausgegeben werden),
- Einzelbitsteuerung.

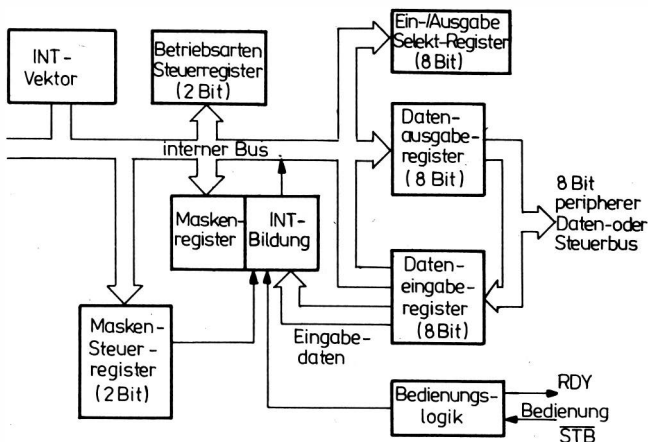


Bild 2.4 Logische Struktur eines Kanals des Bausteins U855 D

## 2.2.2. Beschreibung der Anschlüsse des U 855 D

|                           |                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $D_0 \dots D_7$           | Datenbus zum Prozessor U 880 (bidirectional, tri-state).                                                                                                                                                                                                                                                                                         |
| $B/\overline{A}$          | Kanalauswahl: legt fest, über welchen Kanal der Datentransfer zwischen U 880 und PIO stattfindet (Eingabesignal): «Low» = Tor A, «High» = Tor B.                                                                                                                                                                                                 |
| $C/\overline{D}$ -Auswahl | Steuerwort/Datenwort: legt fest, ob das von U 880 kommende Wort ein Steuerwort oder ein Datenwort ist (Eingabesignal) «Low» = Datenwort, «High» = Steuerwort.                                                                                                                                                                                    |
| $\overline{CE}$           | Chipauswahlsignal (Eingabesignal, aktiv «Low»).                                                                                                                                                                                                                                                                                                  |
| $\overline{M1}$           | Signal für M1-Zyklus: entspricht dem Signal M1 des Prozessors U 880. Es dient zur Synchronisierung der PIO-Logik (Eingabesignal, aktiv «Low»).                                                                                                                                                                                                   |
| $\overline{IORQ}$         | Ein-/Ausgabe-Anforderungssignal: entspricht dem $\overline{IORQ}$ des Prozessors U 880. Es dient der Ansteuerung des Bausteins bei einem Ein- und Ausgabebefehl. Bei gleichzeitigem Erscheinen von $\overline{IORQ}$ und M1 gibt das Interrupt erzeugende Tor automatisch einen Interruptvektor an den Datenbus ab (Eingabesignal, aktiv «Low»). |
| $\overline{RD}$           | Lesesignal: entspricht dem READ-Signal des Prozessors U 880 (Eingabesignal, aktiv «Low»).<br>Der Weg der Daten oder Befehle bei der Ein- und Ausgabe vom U 880 zum PIO wird durch die Signale $\overline{IORQ}$ , $\overline{RD}$ , $B/\overline{A}$ , $C/\overline{D}$ , $\overline{CE}$ gesteuert.                                             |
| IEI                       | Interruptfreigabesignal: dient zur Bildung von Interruptprioritätsschaltungen (Eingabesignal, aktiv «High»).                                                                                                                                                                                                                                     |
| IEO                       | Signal für die Weitergabe der Interruptfreigabe, es ist das zu IEI gehörende Ausgangssignal.<br>Es ist «High», wenn IEI gleich «High» ist und keine Interruptanforderung vom Baustein selbst kommt. Im anderen Fall ist es «Low» (Ausgabesignal).                                                                                                |
| $\overline{INT}$          | Interruptanforderungssignal; wenn das Signal $\overline{INT}$ aktiv ist, fordert das PIO einen Interrupt im Prozessor (Ausgabesignal, aktiv «Low»).                                                                                                                                                                                              |

|                   |                                                                                                                                                                                                                                                                                        |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $A_0 \dots A_7$   | Datenbuskanal A. $A_0 \dots A_7$ dient zur Ein- und Ausgabe der Information zwischen Kanal A und einem peripheren Gerät, $A_0$ ist die niederwertigste Stelle.                                                                                                                         |
| $\overline{ASTB}$ | Strobeimpuls (Torungsimpuls) für Kanal A (Eingabesignal, aktiv «Low»).                                                                                                                                                                                                                 |
| ARDY              | Bereitschaftssignal für Tor A: Bei Ausgabe sagt dieses Signal aus, daß das Ausgaberegister Daten enthält. Bei Eingabe bedeutet dieses Signal, daß der Inhalt des Eingaberegisters abgeholt worden ist und der Kanal bereit ist, neue Daten zu empfangen (Ausgabesignal, aktiv «High»). |
| $B_0 \dots B_7$   | Datenbuskanal B. $B_0 \dots B_7$ dient zur Ein- und Ausgabe der Information zwischen Kanal B und einem peripheren Gerät. $B_0$ ist die niederwertigste Stelle.                                                                                                                         |
| $\overline{BSTB}$ | Strobeimpuls (Torungsimpuls) für Kanal B (Eingabesignal, aktiv «Low»).                                                                                                                                                                                                                 |
| BRDY              | Bereitschaftssignal für Tor B: Bei Ausgabe sagt dieses Signal aus, daß das Ausgaberegister Daten enthält. Bei Eingabe bedeutet dieses Signal, daß der Inhalt des Eingaberegisters abgeholt worden ist und der Kanal bereit ist, neue Daten zu empfangen (Ausgabesignal, aktiv «High»). |

Ein spezielles Löschesignal (RESET) gibt es nicht. Das Löschen der Register und der Flip-Flop des Bausteins geschieht entweder beim Anlegen der Betriebsspannung oder durch  $\overline{M1}$ , wenn nicht gleichzeitig eines der Signale  $\overline{RD}$  bzw.  $\overline{IORQ}$  aktiv ist. Dieses Löschen erzeugt folgenden Zustand:

- die Maskenregister sind gelöscht;
- beide Ausgaberegister sind gelöscht;
- das Register zur Bereitstellung des Interruptvektors wird nicht gelöscht;
- Tor A und B werden hochohmig, die *Hand-shake*-Signale  $\overline{ASTB}$ , ARDY,  $\overline{BSTB}$ , BRDY werden inaktiv.

Der RESET-Zustand bleibt so lange bestehen, bis das PIO ein Steuerwort vom Prozessor erhält.

### 2.2.3. Arbeitsweise des U855 D

Bevor eine Ein- bzw. Ausgabe zum peripheren Gerät vorgenommen werden kann, müssen der Arbeitsmodus und die Interruptlogik eingestellt sein. Dazu werden eine Reihe Steuerwörter in das PIO gespeichert. (Das PIO wird initialisiert.) Zuerst muß vom Prozessor ein Steuerwort zur Auswahl des Arbeitsmodus gegeben werden. Das Steuerwort hat folgenden Aufbau:

| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| M              | M              | x              | x              | 1              | 1              | 1              | 1              |

Die Stellen D<sub>4</sub> und D<sub>5</sub> sind ohne Bedeutung. Durch die Stellen D<sub>6</sub> und D<sub>7</sub> wird der Arbeitsmodus festgelegt.

|         | D <sub>7</sub> | D <sub>6</sub> | Arbeitsmodus       |
|---------|----------------|----------------|--------------------|
| Modus 0 | 0              | 0              | Ausgabe            |
| Modus 1 | 0              | 1              | Eingabe            |
| Modus 2 | 1              | 0              | bidirectional      |
| Modus 3 | 1              | 1              | Einzelbitsteuerung |

Modus 0 Ausgabe:

Nach dem Übergeben des Steuerwortes können bei der Arbeit mit Interrupt ein Interruptvektor und ein Interruptsteuerwort folgen. Der *Interruptvektor* hat folgenden Aufbau:

| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| V <sub>7</sub> | V <sub>6</sub> | V <sub>5</sub> | V <sub>4</sub> | V <sub>3</sub> | V <sub>2</sub> | V <sub>1</sub> | 0              |

Das *Interruptsteuerwort* hat folgenden Aufbau:

| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| M              | x              | x              | x              | 0              | 0              | 1              | 1              |

Ist D<sub>7</sub> gleich 1, so wird die Interruptbildung erlaubt (Bildung des Signals INT). Ist D<sub>7</sub> gleich 0, dann wird die Interruptbildung gesperrt. Nach Speicherung von Interruptsteuerwort und Interruptvektor ist der PIO-Baustein programmiert bzw. initialisiert. Es können die eigentlichen Ausgabebefehle zur Datenausgabe folgen.

Durch den Ausgabebefehl vom Prozessor wird das Ausgaberegister gefüllt und das Signal RDY (entweder ARDY oder BRDY) aktiv. Dieses Signal bleibt so lange aktiv, bis von einem peripheren Gerät das Signal  $\overline{\text{STB}}$  ( $\overline{\text{ASTB}}$ ,  $\overline{\text{BSTB}}$ ) kommt. Mit dem Signal  $\overline{\text{STB}}$  werden gewöhnlich die Daten vom Ausgaberegister in das periphere Gerät übernommen. Ist die Bildung einer Interruptanforderung erlaubt, dann wird durch  $\overline{\text{STB}}$  das Signal  $\overline{\text{INT}}$  erzeugt. Das Signal  $\overline{\text{INT}}$  führt zu einer Programmunterbrechung im Prozessor. Bei Annahme des Interrupt werden gleichzeitig die Signale  $\overline{\text{M1}}$  und  $\overline{\text{IORQ}}$  aktiv. Als Folge der Signale  $\overline{\text{M1}}$  und  $\overline{\text{IORQ}}$  gibt das PIO den Interruptvektor auf den Datenbus. Mit Hilfe des Interruptvektors bestimmt der Prozessor die Startadresse des Interruptbedienprogramms.

### Modus 1 Eingabe:

Nach dem Übergeben des Steuerwortes können bei der Arbeit mit Interrupt ein Interruptvektor und ein Interruptsteuerwort folgen. Der *Interruptvektor* hat folgenden Aufbau:

| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| V <sub>7</sub> | V <sub>6</sub> | V <sub>5</sub> | V <sub>4</sub> | V <sub>3</sub> | V <sub>2</sub> | V <sub>1</sub> | 0              |

Das *Interruptsteuerwort* hat folgenden Aufbau:

| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| M              | x              | x              | x              | 0              | 0              | 1              | 1              |

Ist D<sub>7</sub> gleich 1, so wird die Interruptbildung erlaubt (Bildung des Signals  $\overline{\text{INT}}$ ). Ist D<sub>7</sub> gleich 0, dann wird die Interruptbildung gesperrt. Nach Speicherung von Interruptvektor und Interruptsteuerwort können Eingabebefehle zur Dateneingabe folgen.

Nach einem Eingabebefehl vom Prozessor ist das Eingaberegister leer und das Signal RDY (ARDY, BRDY) aktiv. Dieses Signal bleibt so lange aktiv, bis von einem peripheren Gerät das Signal  $\overline{\text{STB}}$  ( $\overline{\text{ASTB}}$  oder  $\overline{\text{BSTB}}$ ) kommt. Mit der Vorderflanke des Signals  $\overline{\text{STB}}$  können die Daten vom peripheren Gerät in das Eingaberegister übernommen werden.

Ist die Bildung einer Interruptanforderung erlaubt, so wird durch  $\overline{\text{STB}}$  das Signal  $\overline{\text{INT}}$  erzeugt. Das Signal  $\overline{\text{INT}}$  führt zu einer Programmunterbrechung im Prozessor. Bei Annahme des Inter-

rupt werden gleichzeitig die Signale  $\overline{M1}$  und  $\overline{IORQ}$  aktiv. Als Folge der Signale  $\overline{M1}$  und  $\overline{IORQ}$  gibt das PIO den Interruptvektor auf den Datenbus. Mit Hilfe des Interruptvektors bestimmt der Prozessor die Startadresse des Interruptbedienprogramms.

## Modus 2 bidirectional:

Eine bidirectionale Ein- und Ausgabe ist nur über Kanal A möglich. Dabei werden die *Hand-shake*-Signale von Kanal A ( $\overline{ARDY}$ ,  $\overline{ASTB}$ ) zur Ausgabe und die Signale von Kanal B ( $\overline{BRDY}$ ,  $\overline{BSTB}$ ) zur Eingabe benutzt. Vor der Benutzung von Kanal A als bidirektionaler Datenbus muß Kanal B in den Modus Einzelbitsteuerung gebracht werden.

Nach dem Übergeben des Steuerwortes können bei der Arbeit mit Interrupt ein Interruptvektor und ein Interruptsteuerwort folgen. Der *Interruptvektor* hat folgenden Aufbau:

| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| V <sub>7</sub> | V <sub>6</sub> | V <sub>5</sub> | V <sub>4</sub> | V <sub>3</sub> | V <sub>2</sub> | V <sub>1</sub> | 0              |

Das *Interruptsteuerwort* hat wieder folgenden Aufbau:

| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| M              | x              | x              | x              | 0              | 0              | 1              | 1              |

Ist D<sub>7</sub> gleich 1, so wird die Interruptbildung erlaubt (Bildung des Signals  $\overline{INT}$ ). Ist D<sub>7</sub> gleich 0, dann wird die Interruptbildung gesperrt.

Nach der Übergabe von Interruptsteuerwort und Interruptvektor können die Ein- und Ausgabebefehle zum Datentransport gegeben werden. Nach einem Ausgabebefehl vom Prozessor ist das Ausgaberegister gefüllt und das Signal  $\overline{ARDY}$  zur Kennzeichnung der Bereitschaft aktiv.

Eine Abnahme der Daten vom Ausgaberegister kann während der Zeit geschehen, in der  $\overline{ASTB}$  und  $\overline{ARDY}$  aktiv sind. Mit der Rückflanke von  $\overline{ASTB}$  wird bei erlaubtem Interrupt das  $\overline{INT}$ -Signal gebildet. Nach einem Eingabebefehl ist das Signal  $\overline{BRDY}$  aktiv.

Mit dem Signal  $\overline{BSTB}$  können neue Daten von einem peripheren Gerät in das Eingaberegister übernommen werden. Anschließend wird das Signal  $\overline{BRDY}$  inaktiv. Auch in diesem Fall bildet man mit

der Rückflanke von  $\overline{\text{BSTB}}$  das Interruptsignal bei erlaubtem Interrupt.

### Modus 3 Einzelbitsteuerung:

Bei der Einzelbitsteuerung werden die Datenleitungen der Kanäle A und B als einzelne Steuerleitungen verwendet. Die *Hand-shake*-Signale haben in diesem Arbeitsmodus keine Bedeutung. Der Modus dient zur Aufnahme von Alarm- und Statussignalen sowie zur Ausgabe von Steuersignalen. Nach der Ausgabe des *Steuerwortes* mit dem Aufbau

| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 1              | 1              | x              | x              | 1              | 1              | 1              | 1              |

muß ein weiteres Byte folgen, das festlegt, welche Datenleitungen des Kanals Ein- bzw. Ausgangsleitung sein sollen (Auswahlbyte). Ist eine Bit-Stelle dieses Bytes 0, so ist die entsprechende Datenleitung des Kanals Ausgangsleitung. Bei einer 1 in der Bit-Stelle ist die dazugehörige Datenleitung des Kanals eine Eingangsleitung. Wenn z. B. dieses Auswahlbyte den Aufbau

| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 0              | 0              | 1              | 0              | 1              | 0              | 0              | 1              |

hat, so sind die Anschlüsse 0, 3 und 5 Eingangsleitungen und die Anschlüsse 1, 2, 4, 6 und 7 Ausgangsleitungen.

An das Ein-/Ausgabe-Auswahlbyte kann sich bei der Arbeit mit Interrupt ein *Interruptvektor* mit dem Aufbau

| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| V <sub>7</sub> | V <sub>6</sub> | V <sub>5</sub> | V <sub>4</sub> | V <sub>3</sub> | V <sub>2</sub> | V <sub>1</sub> | 0              |

anschließen. Nach dem Interruptvektor folgt das *Interruptsteuerwort*. Es hat den Aufbau:

| D <sub>7</sub>    | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|-------------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| Interrupt erlaubt | ODER UND       | Tief Hoch      | Maske folgt    | 0              | 1              | 1              | 1              |

- Ist  $D_7 = 1$ , so ist die Interruptbildung erlaubt.
- Bei  $D_7 = 0$  ist die Interruptbildung gesperrt.
- Ist  $D_6 = 1$ , so entsteht im PIO das Signal  $\overline{INT}$ , wenn alle als Eingang festgelegten Leitungen das durch  $D_5$  definierte Potential haben (UND-Funktion).
- Bei  $D_6 = 0$  wird im PIO das Signal  $\overline{INT}$  gebildet, wenn eine der als Eingang gekennzeichneten Leitungen das durch  $D_5$  definierte Potential hat (ODER-Funktion).
- Ist  $D_5 = 1$ , so führt ein «Hochpegel» an den Eingangsleitungen zum Interrupt.
- Bei  $D_5 = 0$  führt ein «Tiefpegel» an den Eingangsleitungen zum Interrupt.
- Ist  $D_4 = 1$ , so muß dem Steuerwort eine Interruptmaske folgen.

Interruptmaske

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| $M_7$ | $M_6$ | $M_5$ | $M_4$ | $M_3$ | $M_2$ | $M_1$ | $M_0$ |
|-------|-------|-------|-------|-------|-------|-------|-------|

Zur Interruptbildung werden in diesem Fall nur die Eingänge zugelassen, deren Bit-Stellen in der Interruptmaske 0 sind.

- Bei  $D_4 = 0$  braucht keine Interruptmaske zu folgen.

#### 2.2.4. Beispiel zum U 855 D

An einem PIO Kanal A sollen 3 Steuerleitungen (Ausgabe) und 5 Sensorleitungen (Eingabe) angeschlossen werden.

Dazu werden die Leitungen  $A_0, A_1, A_2$  als Ausgangsleitungen, die Leitungen  $A_3$  bis  $A_7$  als Eingangsleitungen festgelegt. Das Ein-/Ausgabe-Auswahlbyte muß folgenden Aufbau haben:

Ein-/Ausgabe-Auswahlbyte: 1 1 1 1 1 0 0 0

Im Prozessor soll ein Interrupt entstehen, wenn die Sensorsignale  $A_4, A_5$  und  $A_7$  gleichzeitig «Hochpegel» aufweisen. Für diesen Fall sieht das Interruptsteuerwort folgendermaßen aus:

Interruptsteuerwort:

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Interruptmaske:

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Das PIO sei nach Bild 2.5 mit dem Prozessor U 880 verbunden.

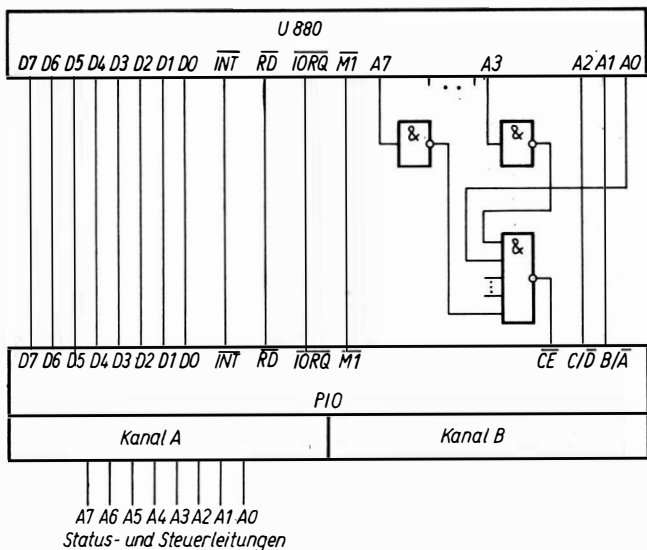


Bild 2.5 Zusammenschaltung eines PIO-Schaltkreises mit dem U880

- Wenn das Adreß-Bit  $A_0 = 1$  und die Adreß-Bits  $A_3$  bis  $A_8$  gleich 0 sind, wird dieses PIO angesteuert.
- Bei  $A_1 = 0$  wird von diesem PIO Kanal A ausgewählt.
- Wenn  $A_2 = 1$  ist, wird über den Datenbus ein Steuerwort, wenn  $A_2 = 0$  ist, ein Datenwort transportiert.

Für die Steuer- und Datenworte nach Kanal A und B ergeben sich aus dieser Festlegung folgende Adressen:

|                                |           |
|--------------------------------|-----------|
| Steuerwort nach Kanal A        | Adresse 5 |
| Datenwort von und nach Kanal A | Adresse 1 |
| Steuerwort nach Kanal B        | Adresse 7 |
| Datenwort von und nach Kanal B | Adresse 3 |

Für die Initialisierung des geforderten Zustands sind folgende Befehle notwendig:

|                                                      | Befehl | Wort auf Datenleitung |
|------------------------------------------------------|--------|-----------------------|
| Ausgabe des Steuerwortes zur Festlegung von Bit-Mode | OUT 5  | 1 1 0 0 1 1 1 1       |

|                                  |       |   |   |   |   |   |   |   |   |
|----------------------------------|-------|---|---|---|---|---|---|---|---|
| Eingabe/Ausgabe                  |       |   |   |   |   |   |   |   |   |
| Auswahlbyte                      | OUT 5 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| Ausgabe Interruptvektor          | OUT 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Ausgabe Interrupt-<br>steuerwort | OUT 5 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| Ausgabe Interruptmaske           | OUT 5 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

Nach dieser Befehlsfolge wird im Prozessor ein Interrupt gefordert, wenn die Leitungen  $A_4$ ,  $A_5$ ,  $A_7$  gleichzeitig «Hochpegel» aufweisen. Durch den Eingabebefehl IN 1 kann festgestellt werden, welche Sensorleitungen Hoch- bzw. Tiefpegel haben. Das durch den Befehl IN 1 in den Prozessor *U 880* eingelesene Datenbyte setzt sich zusammen aus den Bit-Stellen 3 bis 7 des Eingaberegisters und den Bit-Stellen 0 bis 2 des Ausgaberegisters von Kanal A. Weitere Bausteine für die parallele Eingabe sind die Schaltkreise 8255 und 8212. Der Baustein 8255 hat einen dem PIO ähnlichen Aufbau und läßt sich in gleicher Weise einsetzen. Der Schaltkreis 8212 ist in Teil 1 beschrieben.

## 2.3. Zeitgeberbaustein *U 857 D*

### 2.3.1. Struktur des CTC-Bausteins

Bild 2.6 zeigt die logische Struktur des CTC (Counter-Timer-Circuit – Zähler-Zeitgeber-Schaltkreis). Der Baustein CTC ist ein peripherer Baustein zum Prozessor *U 880*. Er dient sowohl als Zähler- wie auch als Zeitgeberbaustein. Mit ihm lassen sich auf 4 getrennten Kanälen Zähler- sowie Zeitgeberfunktionen realisieren. Rechnerseitig hat der Baustein das zum *U 880* passende Interface mit dem 8-Bit-Datenbus und den Steuersignalen  $\overline{IORQ}$ ,  $\overline{RD}$ ,  $\overline{M1}$ , Systemtakt  $\Phi$ ,  $\overline{RESET}$ ,  $\overline{CE}$ ,  $CS_1$ ,  $CS_0$ . Zur Realisierung von Interruptketten verfügt der Baustein über die Signale IEI, IEO,  $\overline{INT}$ .

Bild 2.7 zeigt den Aufbau eines Kanals. Hauptbestandteil des Kanals ist ein 8-Bit-Rückwärtszähler, der über das Zeitkonstantenregister eingestellt werden kann. Die Zählimpulse können über den Takteingang CLK oder über einen Vorteiler kommen. Der Ausgang ZC/TO zeigt den Nulldurchgang des Zählers nach außen. Zur Einstellung der Betriebsarten hat jeder Kanal ein Kanalsteuerregister.

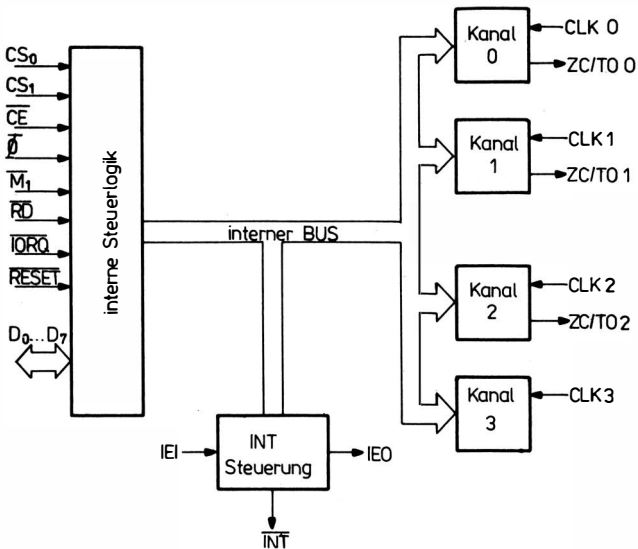


Bild 2.6 Grundstruktur des Bausteins U857D

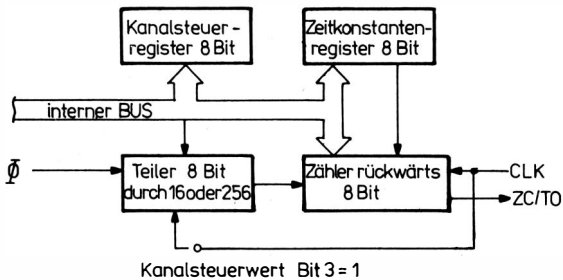


Bild 2.7 Logische Struktur eines Kanals des Bausteins U857D

### 2.3.2. Beschreibung der Anschlüsse des CTC-Bausteins

Im einzelnen haben die Anschlüsse folgende Bedeutung:

$D_0 \dots D_7$  Datenbus zum Prozessor (bidirectional).

$\overline{M1}$ ,  $\overline{IORQ}$ ,  $\overline{RD}$  Steuersignale vom Prozessor (Eingang, aktiv «Low»).

$\overline{CE}$  Chipauswahlsignal (Eingang, aktiv «Low»).

$CS_0$ ,  $CS_1$  Enthält die Nummer des auszuwählenden Kanals (Eingang, aktiv «High»).

| $CS_1$ | $CS_0$ | Kanal-Nr. |
|--------|--------|-----------|
| 0      | 0      | 0         |
| 0      | 1      | 1         |
| 1      | 0      | 2         |
| 1      | 1      | 3         |

$\overline{INT}$  Interruptanforderung vom CTC an den Prozessor (Ausgang, aktiv «Low»).

$IEI$  Interruptfreigabesignal (Eingabe, aktiv «High»). Ist  $IEI$  aktiv («High»), dann kann der Baustein eine Interruptanforderung bilden.

$IEO$  Interruptfreigabesignal für den nächsten in der Kette liegenden Ein-/Ausgabe-Baustein (Ausgabe, aktiv «High»). Der Ausgang  $IEO$  hat nur dann Hochpotential, wenn im Baustein keine Interruptanforderung gebildet wird und  $IEI$  ebenfalls Hochpotential hat. Das heißt, existiert im Baustein und links vom Baustein keine Interruptanforderung, dann werden die Bausteine auf der rechten Seite der Kette freigegeben.

$CLK 0-3$  Externer Zähleingang für die Kanäle 0 bis 3 (Eingang, aktiv «High»).

$ZC/TO 0-2$  Zeigt den Nulldurchgang der Zähler der Kanäle 0 bis 2 an (Ausgang, aktiv «High»).

$\overline{RESET}$  Rücksetzsignal. Dieses Signal stellt die Zähler auf 0, setzt die Interruptfreigabe-Bits zurück (Eingang, aktiv «Low»).

### 2.3.3. Arbeitsweise des CTC-Bausteins

Jeder Kanal kann in der Betriebsart Zähler oder Zeitgeber arbeiten. In der Betriebsart Zähler wird mit der steigenden Flanke des Eingangs CLK der Zähler um 1 zurückgezählt. Der Zähler läuft synchron mit dem Systemtakt  $\Phi$ , d. h., das Rückwärtszählen nach der steigenden Flanke von CLK erfolgt mit dem nächsten Takt. Bei Nulldurchgang des Zählers wird eine Interruptanforderung gebildet und die Zeitkonstante in den Zähler geladen. In der Betriebsart Zeitgeber wird der Rückwärtszähler über einen Vorteiler mit dem Systemtakt  $\Phi$  getaktet. Der Vorteiler läßt sich vom externen Eingang CLK triggern. Der Triggerzeitpunkt wird durch Bit  $D_3$  im Kanalsteuerwort bestimmt. Es gilt:

| $D_3$ | $D_2$ | (Kanalsteuerwort)                                                                                                                                      |
|-------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0     | 0     | Zeitmessung beginnt am Anfang des nächsten Maschinenzyklus.                                                                                            |
| 0     | 1     | Zeitmessung beginnt am Anfang des nächsten Maschinenzyklus nach dem Laden der Zeitkonstante.                                                           |
| 1     | 0     | Zeitmessung beginnt am Anfang des nächsten Maschinenzyklus, nachdem der Eingang CLK die vorgesehene Flanke durchläuft.                                 |
| 1     | 1     | Zeitmessung beginnt am Anfang des nächsten Maschinenzyklus, nachdem die Zeitkonstante geladen ist und die vorgesehene Flanke CLK am Eingang erscheint. |

Beim Nulldurchgang des Zählers wird ein Interrupt gebildet, der zur Realisierung eines Uhrenprogramms mit Hilfe des Mikroprozessors verwendet werden kann. Die Interruptperiode  $t_i$  errechnet sich nach

$$t_i = t_c \cdot P \cdot TK ; \quad (1)$$

mit  $t_c$  – Systemtaktperiode,  $P$  – Vorteiler (16 oder 256),  $TK$  – Zeitkonstante.

#### Programmierung des CTC (Initialisierung)

##### a) Einschreiben des Interruptvektors

Der Interruptvektor hat folgendes Format:

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|-------|-------|-------|-------|-------|-------|-------|-------|
| $V_7$ | $V_6$ | $V_5$ | $V_4$ | $V_3$ | x     | x     | 0     |

Die Bit-Stellen  $D_1$  und  $D_2$  werden im Baustein bei Interruptanforderung mit der betreffenden Kanalnummer belegt.

**b) Einschreiben des Kanalsteuerwortes**

Das Kanalsteuerwort hat folgenden Aufbau:

|       |                                                                                                                                                                                                                                                                                                |                                         |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|
| $D_7$ | INT Freigabe, 0 gesperrt, 1 frei                                                                                                                                                                                                                                                               |                                         |
| $D_6$ | Betriebsart, 0 Zeitgeber, 1 Zähler                                                                                                                                                                                                                                                             |                                         |
| $D_5$ | 0 Teiler 16, 1 Teiler 256                                                                                                                                                                                                                                                                      | } nur in der Betriebsart<br>«Zeitgeber» |
| $D_4$ | 0 neg. Flanke, 1 pos. Flanke                                                                                                                                                                                                                                                                   |                                         |
| $D_3$ | Bestimmt Triggerzeitpunkt                                                                                                                                                                                                                                                                      |                                         |
| $D_2$ | Zeitkonstante laden, 0 kein Laden, 1 das nächste Steuerwort für den angewählten Kanal wird als Zeitkonstante interpretiert. Wird eine Zeitkonstante während eines Zeitmeßvorgangs eingegeben, so erfolgt die Übernahme der Zeitkonstanten in den Zähler erst, wenn der Meßvorgang beendet ist. |                                         |
| $D_1$ | Rücksetzen; 0 Kanalzähler zählt weiter, 1 Kanal unterbricht das Zählen, bis eine Zeitkonstante eingegeben wird.                                                                                                                                                                                |                                         |
|       | ZC/TO = inaktiv; Interrupt ist gesperrt.                                                                                                                                                                                                                                                       |                                         |
| $D_0$ | 1 Kanalsteuerwort, 0 Interruptvektor                                                                                                                                                                                                                                                           |                                         |

**c) Laden der Zeitkonstante**

Ist im Kanalsteuerwort das Laden einer Zeitkonstanten gefordert, so wird die Zeitkonstante nach dem Kanalsteuerwort eingeschrieben. Die Zeitkonstante ist ein Dualwert zwischen 0 und 255.

**d) Lesen des Rückwärtszählers**

Der momentane Wert des Rückwärtszählers kann zu jedem beliebigen Zeitpunkt vom Prozessor aus, mit Hilfe eines Eingabebefehls, gelesen werden. Der Wert repräsentiert die Anzahl der positiven Flanken vor der positiven Flanke von T2 des Lesezyklus.

**Interruptbildung**

Liegt im Baustein eine Bedingung zur Bildung einer Interruptanforderung vor (Nulldurchgang eines Zählers), so wird der INT-Ausgang aktiv, wenn die Interruptbildung erlaubt ist.

## 2.4. Serieller Ein-/Ausgabe-Baustein U856 D

### 2.4.1. Struktur des U856 D

Bild 2.8 zeigt die Grundstruktur des SIO-Bausteins (Seriell Input/Output). Der SIO-Baustein setzt das Rechnerinterface (Rechnerbus) in ein Interface zur Steuerung peripherer Geräte mit serieller Dateneingabe bzw. -ausgabe um. Der Baustein hat eine interne Steuerlogik, die mit Hilfe des Grundtakts arbeitet, sowie eine Interruptsteuerung mit der Möglichkeit zur Bildung von Prioritätsketten. Zur Peripheriesteuerung gibt es 2 Kanäle. Über jeden Kanal können Informationen bitseriell gesendet und empfangen werden.

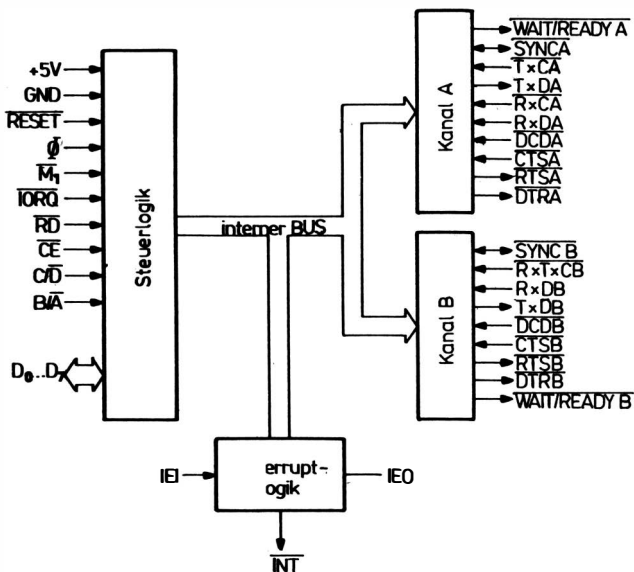


Bild 2.8 Grundstruktur des Bausteins U856 D

## 2.4.2. Beschreibung der Anschlüsse des U856D

Signale des Rechnerinterface:

|                                     |                                                                                                                                                                                                                                |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $D_0 \dots D_7$                     | Datenbus zum Prozessor U880.                                                                                                                                                                                                   |
| $B/\overline{A}$<br>Eingangssignal  | Kanalauswahl; «Low» $\triangleq$ Tor A; «High» $\triangleq$ Tor B.                                                                                                                                                             |
| $C/\overline{D}$<br>Eingangssignal  | Auswahl Steuerwort/Datenwort: Dieses Signal legt fest, ob das vom Prozessor kommende Wort ein Steuerwort («High») oder ein Datenwort («Low») ist.                                                                              |
| $\overline{CE}$<br>Eingangssignal   | Chipauswahlsignal (aktiv $\triangleq$ «Low»).                                                                                                                                                                                  |
| $\overline{M1}$<br>Eingangssignal   | Signal für M1-Zyklus: Dieses Signal entspricht dem Signal $\overline{M1}$ des Prozessors.                                                                                                                                      |
| $\overline{RD}$<br>Eingangssignal   | Lesesignal: Dieses Signal entspricht dem $\overline{RD}$ -Signal des Prozessors (aktiv $\triangleq$ «Low»).                                                                                                                    |
| $\overline{IORQ}$<br>Eingangssignal | Eingabe-/Ausgabe-Anforderung vom Prozessor (aktiv = «Low»).                                                                                                                                                                    |
| IEI<br>Eingangssignal               | Interruptfreigabesignal: Dieses Signal dient zur Bildung von Interruptprioritätsschaltungen.                                                                                                                                   |
| IEO<br>Ausgangssignal               | Signal für die Weitergabe der Interruptfreigabe: Bei diesem Signal handelt es sich um das Ausgangssignal zu IEI. Es ist «High», wenn IEI «High» ist und keine Interruptanforderung vom SIO-Baustein kommt. Sonst ist es «Low». |
| $\overline{INT}$<br>Ausgangssignal  | Interruptanforderungssignal (aktiv $\triangleq$ «Low»).                                                                                                                                                                        |
| $\Phi$<br>Eingangssignal            | Systemtakt.                                                                                                                                                                                                                    |

RESET                      Rücksetzen der Empfänger- und Senderlogik. Aus  $T \times DA$  und  $T \times DB$  entsteht das Grundsignal. Steuerregister sind gelöscht. Alle Interrupts sind gesperrt (aktiv  $\triangleq$  «Low»).

Signale der Kanäle A und B:

$R \times DA, R \times DB$     Empfangsdaten (aktiv  $\triangleq$  «High»).

Eingangssignale

$T \times DA, T \times DB$     Sendedaten (aktiv  $\triangleq$  «High»).

Ausgangssignale

$R \times CA, R \times CB^*)$  Empfängertakt (aktiv  $\triangleq$  «Low»).

Eingangssignale

$T \times CA, T \times CB^*)$  Sendertakt (aktiv  $\triangleq$  «Low»).

Eingangssignale

CTSA, CTSB

Eingangssignale

Steuersignal für Datensender. Im Arbeitsmodus AUTOENABLE werden bei  $\overline{CTS} =$  «High» Daten nicht gesendet. In anderen Arbeitsmode können diese Anschlüsse als Eingänge anderweitig verwendet werden. Die beiden Leitungen haben *Schmitt*-Trigger-Eingänge (aktiv  $\triangleq$  «Low»).

DCDA, DCDB

Eingangssignale

Steuersignal für Datenempfang. Das Signal  $\overline{DCD}$  arbeitet analog dem Signal  $\overline{CTS}$ , jedoch wird durch  $\overline{DCD}$  der Empfänger des zugehörigen Kanals gesperrt (aktiv  $\triangleq$  «Low»).

RTSA, RTSB

Ausgangssignale

Programmierbares Quittungssignal. Sobald das RTS-Bit eines Kanals gesetzt ist, geht der zugehörige  $\overline{RTS}$ -Ausgang in den «Low»-Zustand. Wird das RTS-Bit im Asynchronmodus rückgesetzt, geht der zugehörige  $\overline{RTS}$ -Ausgang in den «High»-Zustand, sobald das Senderegister leer ist. Im Synchronmodus folgt der Anschluß dem RTS-Bit (aktiv  $\triangleq$  «Low»).

|                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\overline{\text{DTRA}}, \overline{\text{DTRB}}^*)$<br>Ausgangssignale | Programmierbares Quittungssignal. Der Ausgang folgt dem Wert des DTR-Bit (aktiv $\triangleq$ «Low»).                                                                                                                                                                                                                                                                                                                                                                                                                            |
| $\overline{\text{SYNCA}}, \overline{\text{SYNCB}}$                     | Synchronisationssignal (aktiv $\triangleq$ «Low»). Bei interner Synchronisation: «Ausgang». Das Signal ist in dem Takt aktiv, in dem ein Synchronisationszeichen erkannt wird.<br>Bei externer Synchronisation: «Eingang». Der Aufbau eines Zeichens beginnt mit der nach SYNC folgenden steigenden Flanke von $\overline{\text{R} \times \text{C}}$ . Zwischen 2 aktiven SYNC-Signalen müssen mindestens 3 Takte liegen. Im Asynchronmodus sind diese Anschlüsse direkte Eingänge für das HUNT SYNC-Bit im Statusregister BRO. |

### 2.4.3. Aufbau eines Kanals des U856D

Bild 2.9 zeigt den Aufbau eines Kanals. Zu einem Kanal gehören folgende Register:

Verschieberegister für den Empfang,  
 Verschieberegister für das Senden,  
 drei Empfangspufferregister,  
 Sendepufferregister,  
 Register für die Speicherung des SYNC-Zeichens,  
 acht Register für Steuerinformationen ( $\text{WR}_0$  bis  $\text{WR}_7$ ),  
 drei Register für Statusinformationen ( $\text{RR}_0$  bis  $\text{RR}_2$ ).

#### *Senden von seriellen Daten:*

Die zu sendende Information gelangt zuerst in den Sendepuffer, anschließend kommt sie in das Verschieberegister und wird von da aus seriell über den Ausgang  $\text{T} \times \text{D}$  gesendet. Die Sende-Bits erscheinen mit der fallenden Flanke von  $\overline{\text{T} \times \text{C}}$ , wobei durch den «Taktzähler Senden» eine Untersetzung programmierbar ist.

\* Aus Gründen begrenzter Pin-Anzahl stehen für  $\overline{\text{T} \times \text{CB}}$ ,  $\overline{\text{R} \times \text{CB}}$  und  $\overline{\text{DTRB}}$  nur 2 Pins zur Verfügung.

Standardversion:  $\overline{\text{T} \times \text{CB}}$  und  $\overline{\text{R} \times \text{CB}}$  gleich 1 Pin  $\overline{\text{R} \times \text{T} \times \text{CB}}$ .

Sonderversion:  $\overline{\text{T} \times \text{CB}}$ ,  $\overline{\text{R} \times \text{CB}}$  2 Pins, DTRB fehlt.

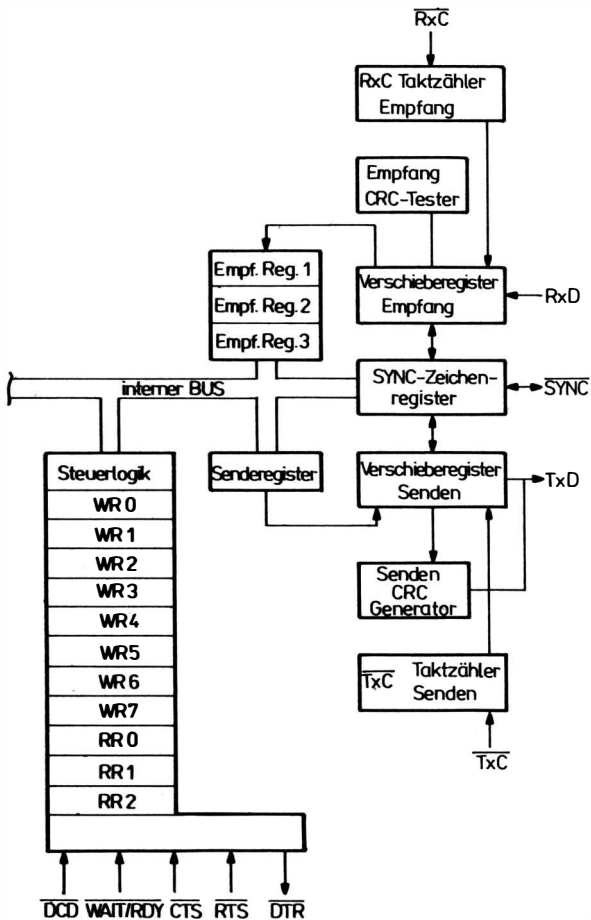


Bild 2.9 Logische Struktur eines Kanals des Bausteins U856 D

### *Empfang von seriellen Daten:*

Die Daten kommen über den Eingang  $R \times D$  ins Verschieberegister und von da aus in die 3 Empfängerpuffer. Die Eingangsdaten werden mit der steigenden Flanke von  $\overline{R \times C}$  abgetastet. Durch den «Taktzähler Empfang» läßt sich eine Untersetzung des Empfangstaktes programmieren.

### *Programmierung des SIO:*

Die Initialisierung des SIO geschieht mit Hilfe von Steuerbytes (C/D-Pin = «High»). Das 1. Steuerbyte nach RESET geht in das Register  $WR_0$ . Das 2. Steuerbyte wird in einem WR-Register gespeichert, dessen Nummer in den Bit-Stellen  $D_2, D_1, D_0$  von  $WR_0$  steht. Das nächste Steuerbyte geht wieder nach  $WR_0$ , dann wieder in das durch die Bit-Stellen  $D_2, D_1, D_0$  angezeigte Register und so fort. Beim Lesen der RR-Register muß nach RESET zunächst ein Steuerbyte in das Register  $WR_0$  gebracht werden, in dem die Nummer des zu lesenden RR-Registers angegeben ist.

## **2.4.4. Arbeitsweise des SIO-Bausteins**

### *Asynchronmodus*

#### **Sendeformat**

| Nullinie | Start | $D_0$ | $D_1$ | $D_2$                     | $\dots$ | $D_n$ | Parity                                                | STOP                   |
|----------|-------|-------|-------|---------------------------|---------|-------|-------------------------------------------------------|------------------------|
|          |       |       |       | 5, 6, 7, 8<br>Bit/Zeichen |         |       | Program-<br>mierbar<br>ja/nein<br>gerade/<br>ungerade | 1, 1½<br>oder<br>2 Bit |

#### **Senden**

- Senden beginnt mit TRANSMIT ENABLE,
- bei AUTOENABLE muß  $\overline{CTS}$  «Low» sein,
- bei  $n < 8$  Bit müssen vordere Bit 0 sein.

#### **Empfang**

- Empfang beginnt mit RECEIVER ENABLE,
- bei AUTOENABLE muß  $\overline{DCD}$  «Low» sein.

Die empfangenen Bit werden in der Mitte einer Bit-Zeit abgeta-

stet. Ist  $\frac{1}{2}$ -Bit-Zeit nach der fallenden Flanke des Eingangssignals noch Tiefpegel, so wird das als Start-Bit erkannt, und die folgenden Bit werden empfangen.

- **BREAK** kann durch Setzen eines Bit in einem WR-Register gesendet werden. Es entsteht «Low»-Pegel, solange das Bit gesetzt ist (Nulllinie ist «High»-Pegel).

### *Synchronmodus*

Im Synchronmodus beginnt der Empfang erst nach Erkennen der Synchronisation. Vom Sender kommt entweder ein SYNC-Signal (externe Synchronisation) oder ein Synchronisationszeichen im Text (interne Synchronisation). Im Synchronmodus existiert nach RESET und RECEIVER ENABLE der HUNTMODE. In diesem Zustand werden keine Daten empfangen. Der HUNTMODE wird beendet, wenn Synchronisation eingetreten ist. Danach beginnt der Datentransfer. Ist die Synchronisation verlorengegangen, so kann der HUNTMODE gesetzt werden.

### *Monosynchronisation*

Der Empfang beginnt, wenn das SYNC-Zeichen (8 Bit) erkannt ist. Das Vergleichszeichen steht in WR<sub>7</sub>. Bei Erkennen des SYNC-Zeichens wird der SYNC-PIN aktiv (bis zum Ende des Taktzyklus, in dem das SYNC-Zeichen erkannt wurde).

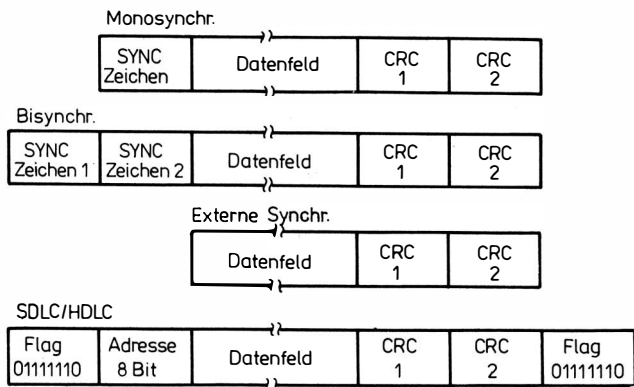


Bild 2.10 Datenformate im Synchronmodus des Bausteins U 856 D

### *Bisynchronisation*

Der Empfang beginnt, wenn ein 16-Bit-SYNC-Wort (2 aufeinanderfolgende Zeichen) erkannt ist. Die Vergleichszeichen stehen in  $WR_6$  und  $WR_7$ . Bei Erkennen der SYNC-Zeichen wird der  $\overline{SYNC}$ -PIN aktiv (vom Erkennen bis zum Ende des Taktzyklus, in dem sie erkannt wurden).

### *Externe Synchronisation*

Der Empfang beginnt mit der 1. steigenden Flanke  $\overline{R \times C}$ , nachdem  $\overline{SYNC}$  ( $\overline{SYNC}$ -PIN) aktiv geworden ist.

Bild 2.10 zeigt die Datenformate für die Betriebsarten des Synchronmodus.

### *Synchronmodus (Zusammenfassung)*

#### Sender

- Mit der fallenden Flanke von  $\overline{T \times C}$  wird ein Bit gesendet.
- Grundzustand (nach RESET) ist Markierungslinie.
- Nach Modusauswahl und TRANSMIT ENABLE ist Grundzustand «Senden von Synchronisationszeichen».
- BREAK ergibt Grundpegel («Low»-Pegel)

#### Empfänger

- Mit der steigenden Flanke von  $\overline{R \times C}$  wird die Information abgetastet.
- Der Empfänger ist nach RESET im HUNTMODE. Er beginnt mit RECEIVER ENABLE.
- Nach Synchronisation beginnt die Datenübernahme.
- Synchronisation liegt vor, wenn
  - bei Monosynchronisation das Zeichen  $WR_7$  erkannt ist;
  - bei Bisynchronisation das 16-Bit-Wort in  $WR_6$  und  $WR_7$  erkannt ist;
  - bei externer Synchronisation der  $\overline{SYNC}$ -PIN von außen «Low» gesetzt ist.
- Abbruch des Empfangs erfolgt, wenn:
  - RESET erscheint;
  - Bit RECEIVER ENABLE rückgesetzt wird;
  - im AUTOENABLE-Zustand  $\overline{DCD}$  «High» wird;
  - der HUNTMODE gesetzt wird.

### *CRC-Bildung*

- CRC-Berechnung beginnt 8-Bit-Zeiten, nachdem das Wort im Empfängerpuffer ist.
- CRC Codes:  $x^{16} + x^{15} + x^2 + 1$  bzw.  
 $x^{16} + x^{12} + x^5 + 1$  bei SDLC.
- CRC-Berechnung läßt sich sperren und freigeben durch das Bit TRANSMIT CRC ENABLE.
- Während CRC-Sendung ist das CRC/SYNC-Bit gesetzt. Das TRANSMIT-BUFFER-EMPTY-Bit ist gesetzt.
- SYNC-Zeichen gehen nicht in CRC ein.

### **2.4.5 Interruptbildung und Statusbildung**

Es gibt 4 Hauptinterruptursachen:

1. Es kommt zum Interrupt, wenn der Sendepuffer keine Daten mehr zum Senden hat. Dieser Interrupt kann durch Nullsetzen des Bit TRANS INTERRUPT ENABLE im Steuerregister  $WR_1$  gesperrt werden.
2. Externer Interrupt bzw. Interrupt tritt auf bei Statuswechsel. Hierzu gehören 5 Ursachen, die im Statusregister  $RR_0$  durch die entsprechenden Bit angezeigt werden:
  - Zustandswechsel («Low»-«High» oder umgekehrt) am CTS-Eingang.
  - Zustandswechsel am DCD-Eingang.
  - Zustandswechsel am SYNC-Eingang (Asynchronmodus und Synchronmodus mit externer Synchronisation) bzw. am SYNC-Ausgang (Monosyncmodus und Bisyncmodus).
  - BREAK oder eine Abbruch-Bit-Folge ist erkannt worden.
  - Beginn des Sendens des CRC-Wortes.Dieser Interrupt kann durch Nullsetzen des Bit EXT INTERRUPT ENABLE im Steuerregister  $WR_1$  gesperrt werden.
3. Je nach eingestelltem Interruptmodus tritt ein Interrupt dann auf, wenn entweder das 1. Zeichen nach Einstellung des Interruptmodus in den Empfängerpuffer gelangt ist oder wenn mindestens 1 Zeichen im Empfängerpuffer zur Verfügung steht. Durch Nullsetzen der Bit  $D_3$  und  $D_4$  im Steuerregister  $WR_1$  kann dieser Interrupt gesperrt werden.

4. Zum Interrupt kommt es bei einer besonderen Empfangsbedingung. Hierzu gehören 4 Ursachen, die im Statusregister  $RR_1$  durch die entsprechenden Bit angezeigt werden:

- Paritätsfehler,
- Empfängerpufferüberlauf,
- Formatfehler,
- Blockende (nur SDLC).

Wann diese Interrupt zugelassen oder gesperrt sind, hängt vom eingestellten Interruptmodus ab.

Nach jedem Interrupt wird die Hauptinterruptursache in den Interruptvektor (Steuerregister  $WR_2$ ) Bit 1 und 2 eingetragen, sofern das Bit STATUS AFFECTS VECTOR im Steuerregister  $WR_1$  gesetzt ist.

### *Interruptbildung Senden*

- TRANSMIT BUFFER EMPTY (bei ASYNC):

Interrupt tritt auf, wenn der Sendepuffer gerade leer geworden ist, d. h., wenn keine zu sendenden Daten mehr im Sendepuffer sind.

- CTS (bei ASYNC und SYNC):

Es kommt zum externen Interrupt, wenn ein Zustandswechsel («Low»-«High» oder umgekehrt) am CTS-Eingang aufgetreten ist.

- CRC/SYNC (bei SYNC):

Während des Sendens des CRC-Wortes ist das Bit TRANSMIT BUFFER EMPTY noch nicht gesetzt. Nachdem das CRC-Wort gesendet ist, wird das Bit TRANSMIT BUFFER EMPTY gesetzt, das wiederum einen Interrupt erzeugt. Wenn das CRC-Senden beginnt, wird das Bit SENDING CRC/SYNC gesetzt und ein Statusinterrupt erzeugt.

- TRANSMIT BUFFER EMPTY (bei SYNC):

Wenn der CRC-Generator nicht arbeitet, kommt es zum Interrupt, wenn der Sendepuffer leer geworden und das 1. SYNC-Zeichen in den Sendepuffer geladen ist. Es beginnt das automatische Senden von SYNC-Zeichen (Monosynchronmodus, Bisynchronmodus).

Interrupt tritt auf, wenn der CRC-Generator arbeitet und wenn der Sendepuffer leer geworden sowie das CRC-Wort gesendet worden ist. Dann beginnt das automatische Senden von SYNC-Zeichen.

## *Interruptbildung Empfang*

### Interruptmodus 1

– RECEIVE CHARACTER AVAILABLE (bei ASYNC und SYNC):

Interrupt tritt auf, wenn das 1. Zeichen eines zu übertragenden Blockes, d. h. das 1. Zeichen nach Einstellen des Interruptmodus, im Empfängerpuffer angekommen ist.

– SYNC/HUNT (bei SYNC):

Es kommt zum externen Interrupt bei einem Zustandswechsel am SYNC-Ausgang vom Zustand «Zeichensynchronisation erreicht» in den Zustand «Herstellen Zeichensynchronisation» und umgekehrt.

– SYNC/HUNT (bei ASYNC):

Es kommt zum Statusinterrupt, wenn ein Zustandswechsel am SYNC-Eingang aufgetreten ist.

– BREAK/ABORT (bei ASYNC):

Ein Statusinterrupt tritt auf, wenn ein Break erkannt worden ist.

– DCD (bei ASYNC und SYNC):

Es kommt zum externen Interrupt, wenn ein Zustandswechsel am DCD-Eingang aufgetreten ist.

– Besondere Empfangsbedingung:

Interrupt tritt auf bei

- Empfängerpufferüberlauf (bei ASYNC und SYNC),
- Formatfehler (bei ASYNC).

Paritätsfehler führen nicht zum Interrupt.

### Interruptmodus 2

– RECEIVE CHARACTER AVAILABLE (bei ASYNC und SYNC):

Interrupt tritt auf, wenn mindestens 1 Zeichen im Empfängerpuffer verfügbar ist. Dieser Interrupt kommt bei jedem Zeichen.

– DCD:

wie Interruptmodus 1

– SYNC/HUNT:

wie Interruptmodus 1

– Besondere Empfangsbedingung:

Interrupt bei Paritätsfehler (bei ASYNC und SYNC).

Der Paritätsfehlerinterrupt ist mit einer Eintragung im Interruptvektor verbunden.

## Interruptmodus 3

Wie Interruptmodus 2, jedoch verursacht der Paritätsfehlerinterrupt keine Eintragung im Interruptvektor.

### *Statusbildung Senden*

#### – TRANSMIT BUFFER EMPTY (bei ASYNC):

Das Bit wird gesetzt, wenn der Sendepuffer leer geworden ist. Es bleibt gesetzt, solange der Sendepuffer leer ist.

#### – TRANSMIT BUFFER EMPTY (bei SYNC):

Bei nicht arbeitendem CRC-Generator wird das Bit gesetzt, wenn der Sendepuffer leer geworden ist, d. h., wenn keine zu sendenden Zeichen mehr im Sendepuffer sind. Es wird mit dem automatischen Senden von SYNC-Zeichen begonnen (Monosyncmodus, Bisyncmodus). Das Bit ist gesetzt, solange der Sendepuffer leer ist, d. h., solange er keine Daten zum Senden hat.

Bei arbeitendem CRC-Generator wird das Bit gesetzt, wenn der Sendepuffer leer geworden und das CRC-Wort gesendet worden ist. Es beginnt das automatische Senden von SYNC-Zeichen.

– (Monosyncmodus, Bisyncmodus). Wenn man den CRC-Generator nicht abschaltet, so wird über die SYNC-Zeichen ein neues Wort berechnet. Das Bit bleibt so lange gesetzt, bis der Sendepuffer leer ist, d. h., solange er keine Daten zum Senden hat.

#### – CTS (bei ASYNC und SYNC):

Das Bit wird gesetzt entsprechend dem Zustand des CTS-Eingangs zum Zeitpunkt des letzten Wechsels eines der 5 Status DCD, CTS, SYNC/HUNT, BREAK/ABORT oder SENDING CRC/SYNC.

#### – ALL SENT (bei ASYNC):

Das Bit wird gesetzt, wenn alle Bit des Schieberegisters den Sender verlassen haben. Es verursacht keinen Interrupt.

#### – CRC/SYNC (bei SYNC):

Das Bit wird gesetzt, wenn, nachdem der Sendepuffer leer geworden ist, mit dem Senden des CRC-Wortes begonnen wird. Nach dem Senden des CRC-Wortes wird das Bit TRANSMIT BUFFER EMPTY gesetzt und mit dem Senden von SYNC-Zeichen begonnen (Monosyncmodus, Bisyncmodus). Das Bit bleibt so lange gesetzt, bis es durch das Kommando «RESET CRC/SYNC», «RESET EXTERNAL/STATUS INTERRUPT» oder «CHANNEL RESET» zurückgesetzt wird. Voraussetzung für diese Statusanzeige ist, daß der CRC-Generator arbeitet.

|                                                                        |   |                              |
|------------------------------------------------------------------------|---|------------------------------|
| Bit CRC/SYNC gesetzt und<br>Bit TRANSMIT BUFFER<br>EMPTY nicht gesetzt | } | CRC-Wort wird gesendet       |
| Bit CRC/SYNC gesetzt und<br>Bit TRANSMIT BUFFER<br>EMPTY gesetzt       |   |                              |
|                                                                        | } | SYNC-Zeichen werden gesendet |

### *Statusbildung Empfang*

- RECEIVER CHARACTER AVAILABLE (ASYNC und SYNC):

Das Bit ist so lange gesetzt, wie mindestens 1 Zeichen im Empfängerpuffer verfügbar ist.

- DCD (ASYNC und SYNC):

Das Bit wird entsprechend dem Zustand des DCD-Signals zum Zeitpunkt des letzten Wechsels eines der 5 Status DCD, CTS, SYNC/HUNT, BREAK/ABORT oder CSC/SYNC gesetzt.

- SYNC/Hunt (ASYNC):

Das Bit wird entsprechend dem Zustand am SYNC-Eingang zum Zeitpunkt des letzten Wechsels eines der 5 Status DCD, CTS, SYNC/HUNT, BREAK/ABORT oder CRC/SYNC gesetzt.

- SYNC/HUNT (SYNC):

Das Bit wird gesetzt, wenn durch Setzen des Bit ENTER HUNT-MODE der Zustand «Herstellen Zeichensynchronisation» eingestellt wird. Es wird rückgesetzt, wenn die Zeichensynchronisation erreicht ist, und bleibt rückgesetzt, auch wenn die Zeichensynchronisation abbricht, bis zum erneuten Setzen des Bit ENTER HUNTMODE. Bei abgeschaltetem Empfänger oder nach einem CHANNEL-RESET-Kommando ist dieses Bit gesetzt.

- INTERRUPT PENDING (ASYNC und SYNC):

Dieses Bit wird gesetzt, wenn irgendeine Interruptbedingung im gesamten SIO vorhanden ist. Es existiert nur im Kanal A.

- Besondere Empfangsbedingungen:

- Empfängerüberlauf (ASYNC und SYNC): Das Bit ist gesetzt bei Empfängerpufferüberlauf. Danach bleibt es so lange gesetzt, bis es durch das Kommando ERROR RESET rückgesetzt wird.

- Format- oder CRC-Fehler (SYNC): Das Bit zeigt das Vergleichsergebnis der CRC-Prüfung an.

- Format- oder CRC-Fehler (ASYNC): Das Bit wird gesetzt, wenn ein Formatfehler aufgetreten ist. Es wird nicht blockiert (bleibt für das Zeichen gesetzt, das den Fehler ausgelöst hat).

- **Paritätsfehler (ASYNC und SYNC):** Dieses Bit wird gesetzt, wenn ein Paritätsfehler aufgetreten ist. Danach bleibt es so lange gesetzt, bis das Kommando ERROR RESET dieses Bit wieder rücksetzt.

- **BREAK/ABORT (ASYNC):**

Das Bit wird gesetzt, wenn ein Break erkannt worden ist. Erst wenn dieser BREAK-Zustand nicht mehr besteht, kann dieses Bit durch das Kommando RESET EXTERNAL/STATUS INTERRUPT rückgesetzt werden.

#### **2.4.6. Beschreibung der Bit-Stellen der Steuerregister WR<sub>0</sub> bis WR<sub>7</sub> und der Statusregister RR<sub>0</sub> bis RR<sub>2</sub> (s. Bild 2.11)**

WR<sub>0</sub>:

D<sub>0</sub> . . . D<sub>2</sub>      Nr. des nächsten zu ladenden oder zu lesenden Registers.

D<sub>3</sub> . . . D<sub>5</sub>      RESET-Kommandos  
SEND ABORT (nur SDLC). Es wird eine Folge von 8 bis 13 «1» gesendet.

RESET EXT STATUS INT löscht die im RR<sub>0</sub> bei Statusinterrupt eingetragenen Bit.

CHANNEL RESET führt zum Rücksetzen eines Kanals. Alle Steuer- und Statusregister sind gelöscht. RESET R × INT ON FIRST CHAR. Nach Erscheinen eines Interrupt im Interruptmodus 1 muß durch RESET R × INT ON FIRST CHAR der Kanal für weitere Interrupt freigegeben werden. RESET T × INT PENDING. Im Interruptmodus 2 erscheint bei leerem Sendepuffer ein Interrupt. Dieses Kommando verhindert, daß dieser Interrupt mehrmals kommt. Nach dem Kommando tritt so lange kein Interrupt auf, bis wieder ein Zeichen in den Sendepuffer gekommen ist. ERROR RESET veranlaßt das Rücksetzen der Fehler-Bit (Parität und Empfängerüberlauf) in RR<sub>1</sub>.

RETURN FROM INT erzeugt einen RETI-Befehl auf dem Datenbus.

D<sub>6</sub> . . . D<sub>7</sub>      setzt die CRC-Logik zurück.

WR 0

D7 D6 D5 D4 D3 D2 D1 D0

Register-Nr.

0 0 —  
 0 1 RESET R \* CRC  
 1 0 RESET T \* CRC  
 1 1 RESET CRC / SYNC  
 0 0 0 —  
 0 0 1 SEND ABORT (SDLC)  
 0 1 0 RESET EXT STATUS INT  
 0 1 1 CHANNEL RESET  
 1 0 0 RESET R \* INT ON FIRST CHAR.  
 1 0 1 RESET T \* INT PENDING  
 1 1 0 ERROR RESET  
 1 1 1 RETURN FROM INT (nur Kanal A)

WR 1

D7 D6 D5 D4 D3 D2 D1 D0

EXT. INTERRUPT  
 ENABLE  
 TRANS INTERRUPT  
 ENABLE  
 STATUS AFFECTS  
 VECTOR (nur B)

0 0 Empfänger-Interrupt  
 gesperrt  
 0 1 Interruptmodus 1  
 1 0 Interruptmodus 2  
 1 1 Interruptmodus 3  
 WAIT/READY als READY  
 WAIT/READY als WAIT  
 WAIT/READY-Freigabe

WR 2

D7 D6 D5 D4 D3 D2 D1 D0

V0  
 V1  
 V2  
 V3  
 V4  
 V5  
 V6  
 V7

WR 3

D7 D6 D5 D4 D3 D2 D1 D0

RECEIVER  
 ENABLE  
 kein SYNC-  
 Zeichen laden  
 Adreß-Such-  
 Mode (SDLC)  
 RECEIVER  
 CRC-ENABLE  
 HUNTMODE  
 AUTO ENABLE

0 0 5  
 0 1 6  
 1 0 7  
 1 1 8 } Bit / Zeichen bei Empfang

WR 4

D7 D6 D5 D4 D3 D2 D1 D0

Einfügen  
 Paritätsbit  
 1 Parität gerade  
 0 Parität ungerade

0 0 SYNC MODE  
 0 1 1 STOP-BIT  
 1 0 1 1/2 STOP-BIT  
 1 1 2 STOP-BIT  
 0 0 MONOSYNC  
 0 1 BISYNC  
 1 0 SDLC  
 1 1 externe SYNC

0 0 X1  
 0 1 X16  
 1 0 X32  
 1 1 X64

WR 5

D7 D6 D5 D4 D3 D2 D1 D0

TRANSMIT  
 CRC ENABLE  
 RTS  
 CRC / SDLC  
 TRANSMIT-ENABLE  
 BREAK

0 0 5  
 0 1 6  
 1 0 7  
 1 1 8 } BIT / Zeichen bei Senden  
 DTR

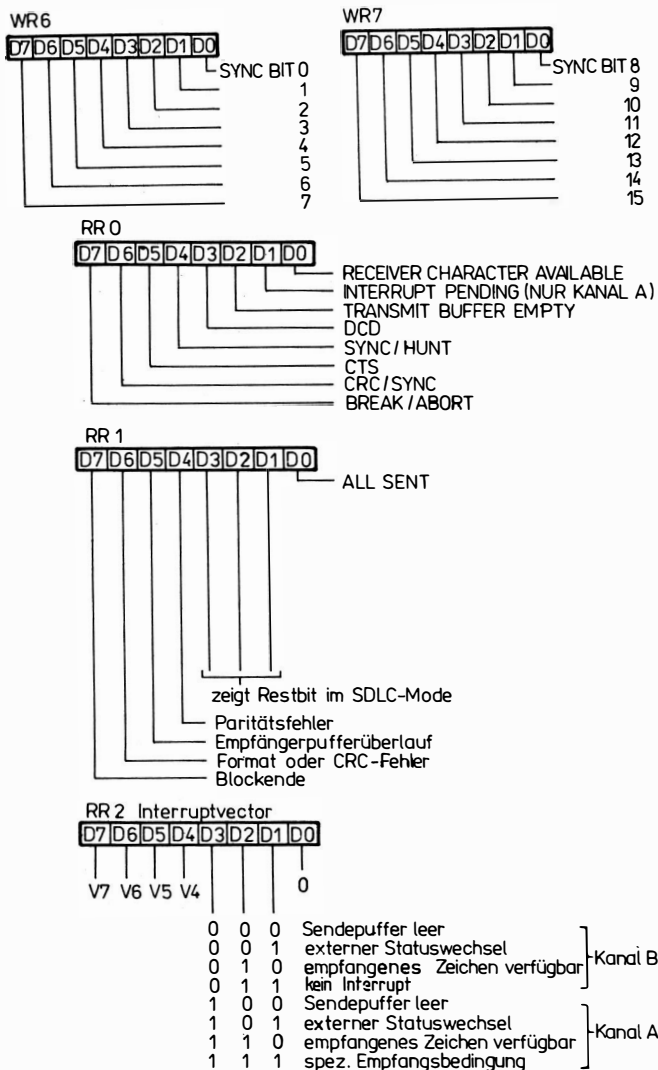


Bild 2.11 Bedeutung der Bitstellen der Steuerregister WR<sub>0</sub> bis WR<sub>7</sub> und der Statusregister RR<sub>0</sub> bis RR<sub>2</sub> im Baustein U856 D

WR<sub>1</sub>:

- D<sub>0</sub> EXT INTERRUPT ENABLE. Gestattet Interrupt bei Wechsel von  $\overline{\text{DCD}}$ ,  $\overline{\text{CTS}}$ ,  $\overline{\text{SYNC}}$ , bei BREAK, bei Sendebeginn von CRC oder SYNC-Zeichen.
- D<sub>1</sub> TRANS INTERRUPT ENABLE. Erlaubt INT, wenn der Sendepuffer leer ist.
- D<sub>2</sub> STATUS AFFECTS VECTOR (nur Kanal B). Ist dieser Modus ausgewählt, so wird der Interruptvektor in WR<sub>2</sub> wie folgt modifiziert:

| V <sub>3</sub> | V <sub>2</sub> | V <sub>1</sub> |                               |           |
|----------------|----------------|----------------|-------------------------------|-----------|
| 0              | 0              | 0              | Sendepuffer leer              | } Kanal B |
| 0              | 0              | 1              | externer Statuswechsel        |           |
| 0              | 1              | 0              | empfangenes Zeichen verfügbar |           |
| 0              | 1              | 1              | spezielle Empfangsbedingung*) |           |
| 1              | 0              | 0              | Sendepuffer leer              | } Kanal A |
| 1              | 0              | 1              | externer Statuswechsel        |           |
| 1              | 1              | 0              | empfangenes Zeichen verfügbar |           |
| 1              | 1              | 1              | spezielle Empfangsbedingung*) |           |

D<sub>3</sub>...D<sub>4</sub> Durch D<sub>3</sub> und D<sub>4</sub> werden spezielle Interruptbildungsmöglichkeiten für Empfänger eingestellt.

D<sub>5</sub> D<sub>5</sub> = 1 Der  $\overline{\text{WAIT}}/\overline{\text{READY}}$ -Ausgang wird aktiv, wenn der Empfänger leer ist.

D<sub>5</sub> = 0 Der  $\overline{\text{WAIT}}/\overline{\text{READY}}$ -Ausgang wird aktiv, wenn der Sendepuffer voll ist.

D<sub>6</sub> D<sub>6</sub> = 0  $\overline{\text{WAIT}}/\overline{\text{READY}}$ -Ausgang arbeitet als WAIT (Ausgang).

D<sub>6</sub> = 1  $\overline{\text{WAIT}}/\overline{\text{READY}}$ -Ausgang arbeitet als READY (Ausgang).

WAIT-Funktion: Das SIO kann keine Daten aufnehmen oder abgeben.

READY-Funktion: Das SIO ist zur Arbeit bereit.

D<sub>7</sub> Freigabe des WAIT-/READY-Ausgangs.

WR<sub>2</sub>: Dieses Register enthält den Interruptvektor. Er existiert nur im Kanal B. Bei STATUS AFFECTS VECTOR werden die Bit-Stellen V<sub>1</sub> bis V<sub>3</sub> entsprechend der Interrupterzeugung eingetragen.

\*) spezielle Empfangsbedingung: Paritätsfehler, Empfängerüberlauf, CRC-/Format-Fehler, Blockende (SDLC)

### WR<sub>3</sub>:

- D<sub>0</sub> Erlaubt Empfangsfunktion.
- D<sub>1</sub> Bei D<sub>1</sub> = 1 werden die SYNC-Zeichen nicht in die Empfangspufferregister geladen. CRC erfolgt normal.
- D<sub>2</sub> D<sub>2</sub> = 1 verhindert im SDLC-Modus Interrupt, wenn keine Übereinstimmung zwischen empfangener und programmierter Adresse oder wenn keine Globaladresse (11111111) vorliegt.
- D<sub>3</sub> Mit D<sub>3</sub> = 1 beginnt die CRC-Kontrolle beim Start des letzten Zeichens vom Empfangsregister zum Puffer.
- D<sub>4</sub> HUNTMODE. Die Übertragung wird abgebrochen, bis ein neues SYNC-Zeichen kommt.
- D<sub>5</sub> AUTOENABLE. Bei D<sub>5</sub> = 1 arbeiten die DCD- und CTS-Eingänge als Empfangs- und Sendesteuerung.
- Bei D<sub>5</sub> = 0 wirken DCD und CTS nur auf ihre Bit im Statusregister.

D<sub>6</sub>/D<sub>7</sub> Bestimmt empfangene Bit/Zeichen.

### WR<sub>4</sub>:

- D<sub>0</sub> D<sub>0</sub> = 1, SIO arbeitet mit Paritäts-Bit,
- D<sub>1</sub> D<sub>1</sub> = 0, Parität gerade,  
D<sub>1</sub> = 1, Parität ungerade.
- D<sub>2</sub>/D<sub>3</sub> Bestimmt die Anzahl der Stop-Bit bei Senden.
- D<sub>4</sub>/D<sub>5</sub> Bestimmt die Art des Synchronmodus.
- D<sub>6</sub>/D<sub>7</sub> Bestimmung des Multiplikationsfaktors zwischen Takt und Übertragungsrate.

### WR<sub>5</sub>:

- D<sub>0</sub> TRANSMIT CRC-ENABLE. CRC-Freigabe bei Senden.
- D<sub>1</sub> D<sub>1</sub> = 1,  $\overline{\text{RTS}}$  geht auf 0.  
D<sub>1</sub> = 0,  $\overline{\text{RTS}}$  geht auf 1, wenn der Sender leer ist.
- D<sub>2</sub> CRC/SDLC  
D<sub>2</sub> = 0, CRC-Kontrolle mit SDLC-Polynom  
$$x^{16} + x^{12} + x^5 + 1.$$
  
D<sub>2</sub> = 1, CRC-Kontrolle mit Polynom  
$$x^{16} + x^{15} + x^2 + 1.$$
- D<sub>3</sub> TRANSMIT ENABLE.  
Sendeerlaubnis. Bei Rücksetzen wird das letzte Zeichen noch voll ausgegeben.

|                                                |                                                                                                                                                                                                                                           |
|------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| D <sub>4</sub>                                 | BREAK. Bewirkt Aussenden von «0» auf T × D.                                                                                                                                                                                               |
| D <sub>5</sub> /D <sub>6</sub>                 | Bestimmt auszugebende Bit/Zeichen.<br>Bei weniger als 5 Bit/Zeichen gilt:<br>1 1 1 1 0 0 0 D sendet 1 Bit<br>1 1 1 0 0 0 D D sendet 2 Bit<br>1 1 0 0 0 D D D sendet 3 Bit<br>1 0 0 0 D D D D sendet 4 Bit<br>0 0 0 D D D D D sendet 5 Bit |
| D <sub>7</sub>                                 | DTR<br>D <sub>7</sub> = 1 bewirkt $\overline{\text{DTR}} = 0$ (aktiv).                                                                                                                                                                    |
| WR <sub>6</sub>                                | SYNC-Zeichen bei Monosynchronisation, die ersten 8 Bit bei Bisynchronisation.                                                                                                                                                             |
| WR <sub>7</sub>                                | 2. Zeichen bei Bisynchronisation oder das Synchronzeichen bei Monosynchronisation. Bei SDLC muß WR <sub>7</sub> = 01111110 sein.                                                                                                          |
| RR <sub>0</sub> :                              |                                                                                                                                                                                                                                           |
| D <sub>0</sub>                                 | RECEIVE CHARACTER AVAILABLE; gesetzt, wenn mindestens 1 Zeichen im Puffer.                                                                                                                                                                |
| D <sub>1</sub>                                 | INTERRUPT PENDING; wird bei jeder INT-Anmeldung gesetzt (nur Kanal A).                                                                                                                                                                    |
| D <sub>2</sub>                                 | TRANSMIT BUFFER EMPTY; gesetzt, wenn der Sendepuffer leer ist (außer bei CRC).                                                                                                                                                            |
| D <sub>3</sub>                                 | DCD; zeigt Zustand $\overline{\text{DCD}}$ .                                                                                                                                                                                              |
| D <sub>4</sub>                                 | SYNC/HUNT<br>D <sub>4</sub> = 0, Zeichensynchronisation erreicht,<br>D <sub>4</sub> = 1, keine Synchronisation.                                                                                                                           |
| D <sub>5</sub>                                 | CTS; zeigt Zustand $\overline{\text{CTS}}$ .                                                                                                                                                                                              |
| D <sub>6</sub>                                 | BREAK/ABORT; wird gesetzt, wenn mit dem Senden von CRC-Worten begonnen wird, nachdem der Sendepuffer leer geworden ist.                                                                                                                   |
| D <sub>7</sub>                                 | CRC/SYNC<br>D <sub>7</sub> = 1 bei Erkennen von BREAK (bei ASYNC).<br>Im SDLC-Format bei Erkennen einer Abbruchfolge (7 oder mehr Einsen).                                                                                                |
| RR <sub>1</sub> :                              |                                                                                                                                                                                                                                           |
| D <sub>0</sub>                                 | ALL SENT wird gesetzt, wenn alle Zeichen im Asynchronmode gesendet sind.                                                                                                                                                                  |
| D <sub>1</sub> /D <sub>2</sub> /D <sub>3</sub> | Falls bei Empfang im SDLC-Modus die Anzahl der Bit/Zeichen nicht aufgehen, zeigt D <sub>1</sub> /D <sub>2</sub> /D <sub>3</sub> die restlichen Bit an.                                                                                    |

|                   |                                                                                        |
|-------------------|----------------------------------------------------------------------------------------|
| D <sub>4</sub>    | Paritätsfehler,                                                                        |
| D <sub>5</sub>    | Empfängerüberlauf,                                                                     |
| D <sub>6</sub>    | Formatfehler im Asynchronmode,<br>CRC-Fehler im Synchronmode.                          |
| D <sub>7</sub>    | Blockende bei SDLC mit gültigem CRC erkannt.                                           |
| RR <sub>2</sub> : | Enthält Interruptvektor. Die Bit sind entsprechend<br>STATUS AFFECTS VECTOR verändert. |

## 2.5. DMA-Steuerbaustein *U880* – *DMA*

### 2.5.1. Funktion und Struktur des DMA-Bausteins

Der DMA-Baustein dient zur Steuerung der Übertragung von Daten von einer Funktionseinheit des Mikrorechners zu einer anderen. Als Funktionseinheiten können dabei periphere Einheiten oder der Arbeitsspeicher auftreten. Damit ist es möglich, mit Hilfe des DMA folgende Datenübertragungen durchzuführen:

- von einer peripheren Einheit zu einer anderen,
- von einem Bereich im Arbeitsspeicher zu einem anderen,
- von einer peripheren Einheit zum Arbeitsspeicher,
- vom Arbeitsspeicher zu einer peripheren Einheit.

Die Übertragung geschieht blockweise. Die Übertragungsgeschwindigkeit beträgt bis zu 1,25 Megabyte/s.

Bild 2.12 zeigt die Grundstruktur des DMA-Bausteins. Der Baustein hat folgende Funktionseinheiten, die über einen internen Bus gekoppelt sind:

- 1 Interface mit Steuerung zum Systembus des Mikrorechners,
- 2 Tore; sie werden mit A und B gekennzeichnet.

Jedes dieser Tore kann die Steuerung einer Funktionseinheit (periphere Einheit oder Arbeitsspeicher) übernehmen. Es kann zum Lesen und zum Einschreiben benutzt werden.

- 1 Steuerlogik mit einem Registersatz,
- 1 Interrupt- und Bussteuerung.

Der DMA-Baustein hat folgende Register:

**Steuerregister**

Dieses Register enthält die notwendigen Steuerinformationen.

**Zeitablaufregister**

In diesem Register ist die Steuerinformation für die Art und Weise der Zyklensteuerung enthalten.

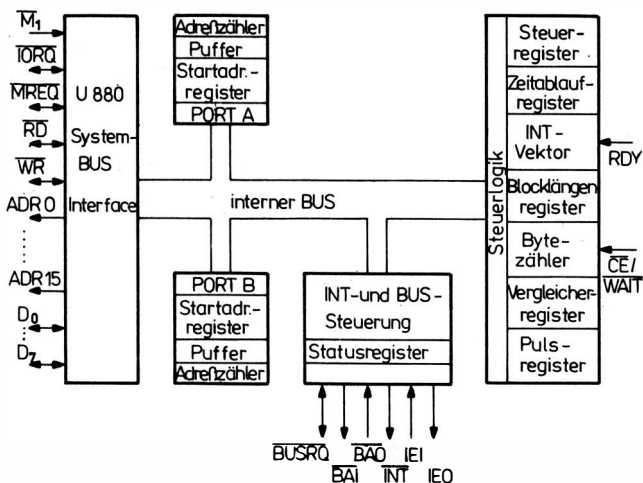


Bild 2.12 Grundstruktur des Bausteins U 880 - DMA

### Interruptvektorregister

Dieses Register enthält den Interruptvektor.

### Blocklängenregister

Dieses Register enthält die gesamte zu suchende oder zu übertragende Blocklänge.

### Bytezähler

Die gelesenen Bytes werden im Bytezähler gezählt. Bei Übereinstimmung mit dem Blocklängenregister wird ein Bit im Statusregister gesetzt.

### Vergleichsregister

Es enthält ein Byte, nach dem beim Suchvorgang gesucht wird.

### Maskenregister

Dieses Register enthält eine Maske. Ein Vergleich eines Bit zwischen gelesenen Byte und Vergleichsbyte findet statt, wenn im Maskenregister an der entsprechenden Stelle eine «0» steht.

### Startadreßregister für Tor A und B

Sie enthalten die Startadressen (16 Bit) für die beiden Tore. Bei E/A-Operationen sind es nur 8 Bit.

### Adreßzähler für Tor A und B

Diese Zähler enthalten die laufenden Adressen der beiden Tore (16 Bit).

## Pulssteuerregister

Sobald die im Pulssteuerregister angegebene Anzahl Byte abgearbeitet ist, wird am Ausgang  $\overline{\text{INT}}$  ein Impuls ausgegeben. Die CPU reagiert nicht, da durch  $\overline{\text{BUSRQ}}$  der DMA aktiviert ist.

## Statusregister

Dieses Register enthält die gesammelte Statusinformation.

### 2.5.2. Beschreibung der Anschlüsse des DMA-Bausteins

|                             |                                                                                                                                                                                           |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $A_0 \dots A_{15}$          | Adreßbus (Tri-state-Ausgang, aktiv «High»).                                                                                                                                               |
| $D_0 \dots D_7$             | Datenbus (Tri-state, bidirectional, aktiv «High»).                                                                                                                                        |
| $\overline{\text{M1}}$      | Signal für M1-Zyklus (Eingabesignal, aktiv «Low»).                                                                                                                                        |
| $\overline{\text{IORQ}}$    | Eingabe-/Ausgabe-Anforderung (Ein- bzw. Ausgabesignal aktiv «Low»).                                                                                                                       |
| $\overline{\text{MREQ}}$    | Speicheranforderung (Ein- bzw. Ausgang, aktiv «Low»).                                                                                                                                     |
| $\overline{\text{RD}}$      | Lesen (Ein- bzw. Ausgang, aktiv «Low»).                                                                                                                                                   |
| $\overline{\text{WR}}$      | Schreiben (Ein- bzw. Ausgang, aktiv «Low»).                                                                                                                                               |
| $\overline{\text{CE/WAIT}}$ | (Eingang, aktiv «Low»); dient als Bausteinfreigabesignal. Es kann, wenn $\overline{\text{BAI}} = \text{«Low»}$ ist, auch als «WAIT» verwendet werden. Die Auswahl erfolgt durch Programm. |
| $\overline{\text{BUSRQ}}$   | Busanforderung (Ein-/Ausgang, aktiv «Low»); Signal zur Anforderung einer DMA-Funktion.                                                                                                    |
| $\overline{\text{BAI}}$     | Busanforderungsbestätigung (Eingang, aktiv «Low»); zeigt an, daß die Buskontrolle an den DMA-Baustein übergeben wurde.                                                                    |
| $\overline{\text{BAO}}$     | Weitergabe $\overline{\text{BAI}}$ (Ausgang, aktiv «Low»); gibt $\overline{\text{BAI}}$ -Signal an den nächsten Baustein weiter.                                                          |
| $\overline{\text{INT}}$     | Interruptanforderung (Ausgang, aktiv «Low»).                                                                                                                                              |
| $\overline{\text{IEI}}$     | Interruptfreigabe (Eingang, aktiv «High»).                                                                                                                                                |
| $\overline{\text{IEO}}$     | Weitergabe Interruptfreigabe (Ausgang, aktiv «High»).                                                                                                                                     |
| $\overline{\text{RDY}}$     | Quittung (Eingang aktiv «High» oder «Low», je nach Programmierung). RDY teilt dem DMA mit, ob die periphere Einheit zum Lesen oder Schreiben bereit ist.                                  |

### 2.5.3. Arbeitsweise des DMA-Bausteins

#### *Busanforderung*

Wenn RDY aktiv ist, erzeugt die DMA das Signal  $\overline{\text{BUSRQ}}$ . Die CPU aktiviert danach BUSAK. Mit BUSAK muß BAI aktiv werden. Nach BAI beginnt mit der nächsten Taktflanke von T1 ein DMA-Zyklus. Nach der Busanforderung läuft die eingestellte Betriebsart ab.

#### *Betriebsarten*

Alle Betriebsarten können in den Funktionsarten Übertragung, Suchen, Übertragung und Suchen laufen. Es gibt folgende Betriebsarten:

**Blockweise** Die Übertragung ist abgeschlossen, wenn der Block übertragen oder das gesuchte Byte gefunden ist (unabhängig von RDY).

**Andauernd (burst)** Die Blockübertragung arbeitet nur, wenn RDY aktiv ist.

**Transparent** Die DMA-Operation wird während der Refresh-Zyklen ausgeführt.

**Byteweise** Die CPU übernimmt die Buskontrolle nach jeder Einzelbyteoperation.

#### *Blockübertragung*

Bei Speicherzugriff steuert die DMA einen Speicherlesezyklus und speichert das Byte im Lesetor. Danach erfolgen bei «Übertragung» die Übergabe zum Schreiber und ein Speicherschreibzyklus. Bei «Suchen» kommt es zum Vergleich mit dem Byte im Vergleicheregister, und bei Gleichheit wird ein Status-Bit gesetzt. Adreß- und Bytezähler werden dabei erhöht. Bei Ein-/Ausgabe laufen Ein- bzw. Ausgabezyklen ab. Die Zyklenlänge läßt sich programmieren, jedoch muß die Mindestlänge analog den CPU-Zyklen eingehalten werden. Durch die Verlängerung spart man eine zusätzliche WAIT-Logik. Nach RESET arbeiten die Zyklen analog den CPU-Zyklen.

Die Busfreigabe geschieht bei «Übertragung» nur am Blockende, bei «Suchen» bei Gleichheit oder am Blockende. Bei Gleichheit wird vor der Busfreigabe noch eine Byte-Operation ausgeführt.

## Andauernde Übertragung

Der Ablauf in dieser Betriebsart unterscheidet sich von der «Blockübertragung» dadurch, daß es während  $RDY = \langle 0 \rangle$  zu einer Busfreigabe kommt.

## Byteweise Übertragung

Die Busfreigabe ist unabhängig von  $RDY$  nach jedem Lesezyklus bei «Suchen» und nach jedem Schreibzyklus bei «Übertragen». Nachdem  $\overline{BUSRQ}$  inaktiv ist, kann ein erneutes  $\overline{BUSRQ}$  erst kommen, wenn vorher  $\overline{BAI}$  inaktiv geworden ist.

## Programmierung des DMA (Initialisierung)

Das Einstellen der Funktion des DMA geschieht durch eine Reihe Steuerbytes, mit deren Hilfe die einzelnen Register gefüllt werden. Die unterschiedlichen Steuerbytes sind durch die Bit-Stellen  $D_0$ ,  $D_1$  und  $D_7$  gekennzeichnet. Die Steuerbytes werden in Steuerregister gespeichert. Sie werden 1A, 1B, 2A, 2B, 2C, 2D genannt.

### Kennzeichnung der Steuerbytes:

|    | $D_7$ | $D_1$    | $D_0$    |
|----|-------|----------|----------|
| 1A | 0     | $\neq 0$ | $\neq 0$ |
| 1B | 0     | 0        | 0        |
| 2A | 1     | 0        | 0        |
| 2B | 1     | 0        | 1        |
| 2C | 1     | 1        | 0        |
| 2D | 1     | 1        | 1        |

### Übersicht über die Steuerbytes

#### Steuerbyte 1A

|       |                                        |                                                    |
|-------|----------------------------------------|----------------------------------------------------|
| $D_7$ | 0                                      |                                                    |
| $D_6$ | Blocklänge, «High»-Byte folgt          |                                                    |
| $D_5$ | Blocklänge, «Low»-Byte folgt           |                                                    |
| $D_4$ | Adresse A, «High»-Byte folgt           |                                                    |
| $D_3$ | Adresse A, «Low»-Byte folgt            |                                                    |
| $D_2$ | $0 B \rightarrow A; 1 A \rightarrow B$ |                                                    |
| $D_1$ | 0                                      | } -; 0 } Transfer; 1 } Suche; 1 } Suche + Transfer |
| $D_0$ | 0                                      |                                                    |

### Steuerbyte 1B

|                |                                                                    |
|----------------|--------------------------------------------------------------------|
| D <sub>7</sub> | 0                                                                  |
| D <sub>6</sub> | Zeitablaufbyte folgt                                               |
| D <sub>5</sub> | Toradresse ist fest                                                |
| D <sub>4</sub> | 0 Toradresse wird decrementiert<br>1 Toradresse wird incrementiert |
| D <sub>3</sub> | 0 } B = Speicher; 0 } A = Speicher; 1 } B = E/A;                   |
| D <sub>2</sub> | 0 } B = Speicher; 1 } A = Speicher; 0 } B = E/A;                   |
|                | 1 } A = E/A                                                        |
|                | 1 }                                                                |
| D <sub>1</sub> | 0                                                                  |
| D <sub>0</sub> | 0                                                                  |

### Zeitablaufbyte

|                |                                                 |
|----------------|-------------------------------------------------|
| D <sub>7</sub> | 0 = $\overline{WR}$ -Ende 1/2 Zyklus früher     |
| D <sub>6</sub> | 0 = $\overline{RD}$ -Ende 1/2 Zyklus früher     |
| D <sub>5</sub> | frei                                            |
| D <sub>4</sub> | frei                                            |
| D <sub>3</sub> | 0 = $\overline{MREQ}$ -Ende 1/2 Zyklus früher   |
| D <sub>2</sub> | 0 = $\overline{IORQ}$ -Ende 1/2 Zyklus früher   |
| D <sub>1</sub> | 0 } Zykluslänge 0 } Zykluslänge 1 } Zykluslänge |
| D <sub>0</sub> | 0 } = 4; 1 } = 3; 0 } = 2                       |

Bei «Übertragung» muß das Zeitablaufbyte zweimal folgen, erst für Tor A, dann für Tor B.

### Steuerbyte 2A

|                |                      |
|----------------|----------------------|
| D <sub>7</sub> | 1                    |
| D <sub>6</sub> | DMA-Freigabe         |
| D <sub>5</sub> | Interruptfreigabe    |
| D <sub>4</sub> | Vergleichsbyte folgt |
| D <sub>3</sub> | Maskenbyte folgt     |
| D <sub>2</sub> | Stop bei Vergleich   |
| D <sub>1</sub> | 0                    |
| D <sub>0</sub> | 0                    |

### Steuerbyte 2B

|                |                                          |
|----------------|------------------------------------------|
| D <sub>7</sub> | 1                                        |
| D <sub>6</sub> | Be- 0 } byte- 0 } block- 1 } burst 1 } - |
| D <sub>5</sub> | triebsart; 0 } weise; 1 } weise; 0 }     |
| D <sub>4</sub> | Interruptsteuerbyte folgt                |

|                |                              |
|----------------|------------------------------|
| D <sub>3</sub> | Adresse B, «High»-Byte folgt |
| D <sub>2</sub> | Adresse B, «Low»-Byte folgt  |
| D <sub>1</sub> | 0                            |
| D <sub>0</sub> | 1                            |

#### Interruptsteuerbyte

|                |                                  |
|----------------|----------------------------------|
| D <sub>7</sub> | frei                             |
| D <sub>6</sub> | Interrupt vor BUSRQ              |
| D <sub>5</sub> | Status verändert Interruptvektor |
| D <sub>4</sub> | Interruptvektor folgt            |
| D <sub>3</sub> | Pulszählerbyte folgt             |
| D <sub>2</sub> | Pulse werden erzeugt             |
| D <sub>1</sub> | Interrupt bei Blockende          |
| D <sub>0</sub> | Interrupt bei Gleichheit         |

#### Interruptvektor

|                |                |                            |   |                                   |   |                                  |   |                                                 |
|----------------|----------------|----------------------------|---|-----------------------------------|---|----------------------------------|---|-------------------------------------------------|
| D <sub>7</sub> | V <sub>7</sub> |                            |   |                                   |   |                                  |   |                                                 |
| D <sub>6</sub> | V <sub>6</sub> |                            |   |                                   |   |                                  |   |                                                 |
| D <sub>5</sub> | V <sub>5</sub> |                            |   |                                   |   |                                  |   |                                                 |
| D <sub>4</sub> | V <sub>4</sub> |                            |   |                                   |   |                                  |   |                                                 |
| D <sub>3</sub> | V <sub>3</sub> |                            |   |                                   |   |                                  |   |                                                 |
| D <sub>2</sub> | 0              | } bei 0 Interrupt auf RDY; | 0 | } bei 1 Interrupt auf Gleichheit; | 1 | } bei 1 Interrupt auf Blockende; | 1 | } bei 1 Interrupt auf Gleichheit oder Blockende |
| D <sub>1</sub> | 0              |                            | 1 |                                   | 0 |                                  | 1 |                                                 |
| D <sub>0</sub> | 0              |                            |   |                                   |   |                                  |   |                                                 |

#### Steuerregister 2C

|                |                                                                  |
|----------------|------------------------------------------------------------------|
| D <sub>7</sub> | 1                                                                |
| D <sub>6</sub> | frei                                                             |
| D <sub>5</sub> | 0 <u>Stop bei Blockende</u> ; 1 <u>wiederholen bei Blockende</u> |
| D <sub>4</sub> | 0 <u>CE</u> nur; 1 <u>CE/WAIT</u>                                |
| D <sub>3</sub> | 0 RDY aktiv «Low»; 1 RDY aktiv «High»                            |
| D <sub>2</sub> | frei                                                             |
| D <sub>1</sub> | frei                                                             |
| D <sub>0</sub> | 0                                                                |

## Steuerregister 2D

|                |                |                |                |                |                |                |                |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|

- 1 0 0 0 0 Chip zurücksetzen;
- 1 0 0 0 1 gleicht Zeitverhalten DMA-Tor A an CPU an;
- 1 0 0 1 0 gleicht Zeitverhalten DMA-Tor B an CPU an;
- 1 0 0 1 1 0 → Bytezähler, laden Startadressen\*);
- 1 0 1 0 0 Ausführung wird bei momentanem Adreßstand und voller Blocklänge begonnen (Restart);
- 0 1 0 1 0 Interruptfreigabe;
- 0 1 0 1 1 Interrupt sperren;
- 0 1 0 0 0 alle Interrupt werden zurückgesetzt;
- 0 0 0 0 1 Chipfreigabe, Operationen werden freigegeben;
- 0 0 0 0 0 Chipsperre, Operationen werden blockiert;
- 0 1 1 1 0 Lesemaske folgt;
- 0 1 0 0 1 das erste Register, das durch Lesemaske frei ist, wird als nächstes gelesen;
- 0 1 1 1 1 als nächstes Register wird das Statusregister gelesen
- 0 1 1 0 0 erzwungenes RDY, es wird intern ein RDY erzeugt;
- 0 1 1 0 1 vom DMA kommt keine Busanforderung, solange das RETI nicht kommt;
- 0 0 0 1 0 Rücksetzen Vergleichs-Bit und Blockende-Bit im Statusregister.

## Statusregister

- D<sub>7</sub> frei
- D<sub>6</sub> frei
- D<sub>5</sub> Blockende-Bit, 0 = Blockende
- D<sub>4</sub> Vergleichs-Bit, 0 = Vergleich
- D<sub>3</sub> 0INT-Pending

\* Bekommt der betreffende Kanal eine Festadresse, so ist das Laden der Startadresse (für Festadresse) nur möglich, wenn dieser Kanal als Eingabekanal definiert ist.

|                |                                                              |
|----------------|--------------------------------------------------------------|
| D <sub>2</sub> | frei                                                         |
| D <sub>1</sub> | 0 RDY = aktiv                                                |
| D <sub>0</sub> | 0 DMA-Zyklus nicht erschienen<br>1 DMA-Zyklus ist erschienen |

### *Lesen der DMA-Register*

Folgende DMA-Register können sequentiell in der aufgezählten Reihenfolge gelesen werden:

Statusregister,  
 Bytezähler niederwertiger Teil,  
 Bytezähler höherwertiger Teil,  
 Adresse A niederwertiger Teil,  
 Adresse A höherwertiger Teil,  
 Adresse B niederwertiger Teil,  
 Adresse B höherwertiger Teil.

Das Weiterschalten von Register zu Register geschieht intern. Soll ein Register nicht gelesen werden, so kann es durch Setzen von «0» an der entsprechenden Stelle der Lesemaske übersprungen werden. Nach RESET und Lesen Statusregister weist der interne Zeiger immer auf das Statusregister.

Lesemaske:

|                |                |
|----------------|----------------|
| D <sub>7</sub> |                |
| D <sub>6</sub> | Adresse B HWT  |
| D <sub>5</sub> | Adresse B NWT  |
| D <sub>4</sub> | Adresse A HWT  |
| D <sub>3</sub> | Adresse A NWT  |
| D <sub>2</sub> | Bytezähler HWT |
| D <sub>1</sub> | Bytezähler NWT |
| D <sub>0</sub> | Statusregister |

### **2.5.4. Beispiel zum DMA-Baustein**

Ein DMA-Baustein soll die Ausgabe eines Datenblocks aus dem Speicher auf einen über ein PIO angeschlossenen Drucker steuern. Bild 2.13 zeigt die notwendigen Verbindungen zwischen U880, PIO und DMA. Der Drucker ist am Kanal A des PIO angeschlossen und arbeitet im *Hand-shake*-Betrieb mit den Signalen ARDY und ASTB, Kanal A des PIO wird in die Betriebsart «Ausgabe» in-

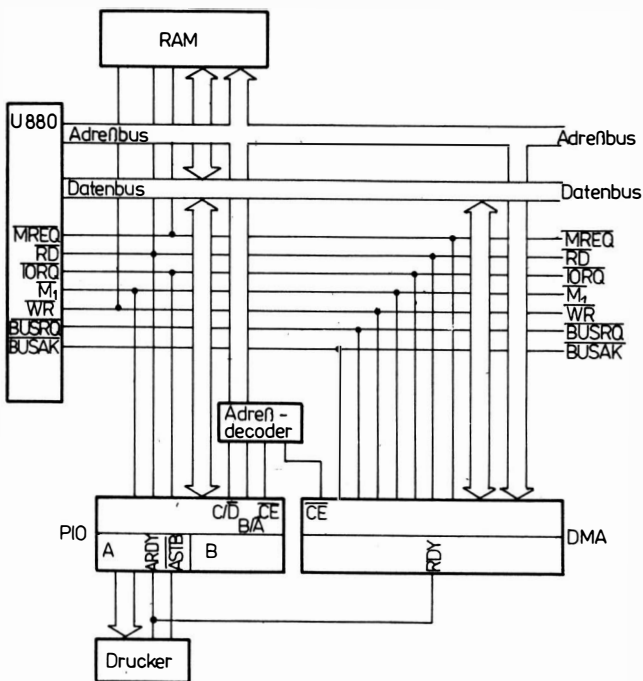


Bild 2.13 Anschluß eines DMA-Bausteins an den Systembus U880

initialisiert. Interrupts werden gesperrt. Der DMA wird in die Betriebsart «Byteweise» und in die Funktionsart «Übertragung» initialisiert. Kanal A soll den Speicher, Kanal B das PIO steuern. Solange am PIO das Signal ARDY aktiv ist (Daten sind nicht vom Drucker abgeholt), soll der DMA den Bus freigeben. Dazu wird ARDY mit RDY des DMA verbunden und für RDY aktiv «Low» festgelegt. Durch den Adreßdecoder sollen folgende Adressen festgelegt werden:

|     |              |         |             |
|-----|--------------|---------|-------------|
| PIO | Datenwort    | Kanal A | Adresse 00H |
|     | Steuerwort   | Kanal A | Adresse 40H |
|     | Datenwort    | Kanal B | Adresse 80H |
|     | Steuerwort   | Kanal B | Adresse C2H |
| DMA | Bausteinwahl |         | Adresse 30H |

Zur Initialisierung der Übertragung vom Speicher zum Drucker müssen folgende Bit-Muster von der CPU *U 880* ausgegeben werden:

|                                                                                                                                                          | Befehl   | Wort<br>auf Datenbus |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|----------|----------------------|
| Ausgabe Steuerwort zur<br>Festlegung der Betriebsart<br>«Ausgabe» im PIO Kanal A                                                                         | OUT 40 H | 00001111             |
| Interruptsperre für<br>PIO Kanal A                                                                                                                       | OUT 40 H | 00000011             |
| Steuerbyte 1A für DMA<br>Übertragung B → A<br>Da Kanal B mit einer festen<br>Adresse versehen wird, muß<br>er erst als Eingabekanal<br>definiert werden. | OUT 30 H | 00000001             |
| Steuerbyte 1B für<br>DMA Kanal B<br>Kanal B zur Peripherie-<br>steuerung mit fester Adresse                                                              | OUT 30 H | 00101000             |
| Steuerbyte 2B<br>Byteweise Übertragung<br>Festlegen Adresse für<br>Kanal B                                                                               | OUT 30 H | 10000101             |
| Adresse PIO Datenkanal A<br>zum DMA Kanal B                                                                                                              | OUT 30 H | PIO-Adresse          |
| Steuerbyte 2D<br>Laden Toradresse B                                                                                                                      | OUT 30 H | 11001111             |
| Steuerbyte 1A für DMA<br>Übertragung A → B                                                                                                               | OUT 30 H | 01111101             |
| Kanal A «Low»-Byte                                                                                                                                       | OUT 30 H | «Low»-Byte           |
| Kanal A «High»-Byte                                                                                                                                      | OUT 30 H | «High»-Byte          |
| Blocklänge «Low»-Byte                                                                                                                                    | OUT 30 H | «Low»-Byte           |
| Blocklänge «High»-Byte                                                                                                                                   | OUT 30 H | «High»-Byte          |
| Steuerbyte 1B für<br>DMA-Kanal A<br>Kanal steuert Speicher,<br>Adresse wird incrementiert,<br>kein Zeitsteuerbyte                                        | OUT 30 H | 00010100             |

|                       |          |          |
|-----------------------|----------|----------|
| Steuerbyte 2C         | OUT 30 H | 10000010 |
| RDY aktiv «Low»       |          |          |
| kein WAIT-Status      |          |          |
| Steuerbyte 2D         | OUT 30 H | 11001111 |
| Laden Toradresse A    |          |          |
| Löschen Blockzähler   |          |          |
| Steuerbyte 2D         | OUT 30 H | 10000111 |
| Starten DMA-Operation |          |          |

## 2.6. Serieller Ein-/Ausgabeschaltkreis 8251

### 2.6.1. Funktionen des 8251

Ein neben dem SIO häufig verwendeter Baustein, der parallel in serielle Information umgewandelt wird, ist der serielle Schaltkreis 8251 (USART = Universal Synchron A Transmitter). Er hat nicht so viele Möglichkeiten, ist aber für viele Anwendungsfälle ausrechenend. Er ermöglicht den Anschluß peripherer Geräte über serielle Schnittstellen. Mit dem 8251 lassen sich folgende Betriebsarten realisieren:

#### Asynchron

- 5 bis 8 Zeichen
- Taktuntersetzung  $\times 1$ ,  $\times 16$  oder  $\times 64$
- Erzeugung BREAK
- 1,  $1\frac{1}{2}$  oder 2 Stopbits

#### Synchron

- 5 bis 8 Zeichen
  - 1 oder 2 Synchronisationszeichen
  - externe Synchronisation
- (jedoch kein SDLC und keine CRC-Prüfung)

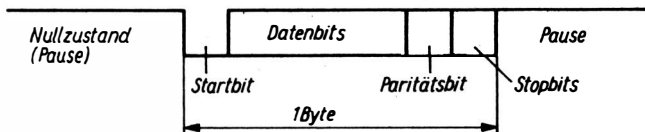
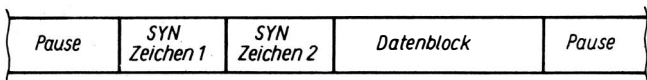


Bild 2.14 Asynchrones Datenprotokoll



- Es sind ein- oder zwei Synchronisationszeichen möglich
- In der Pause werden die Synchronisationszeichen übertragen
- Es existiert nur 1 Kanal für serielle Send- und Empfangsdaten

Bild 2.15 Synchrones Datenprotokoll

## 2.6.2. Anschlußbelegung

### Beschreibung der Anschlüsse

|                  |                                                                                                                                 |
|------------------|---------------------------------------------------------------------------------------------------------------------------------|
| $D_0-D_7$        | Datenbus (Bidirektional).                                                                                                       |
| $\overline{C/D}$ | Unterscheidung Steuerwort – Datenwort<br>1 = Steuerwort, 0 = Datenwort.                                                         |
| $\overline{RD}$  | aus dem Baustein Daten lesen (aktiv «Low»).                                                                                     |
| $\overline{WR}$  | in den Baustein Daten schreiben (aktiv «Low»).                                                                                  |
| $\overline{CS}$  | Bausteinfreigabe (aktiv «Low»).                                                                                                 |
| CLK              | Takteingang.                                                                                                                    |
| RESET            | Rücksetzen der Betriebsarteneinstellung (aktiv «High»).                                                                         |
| $T \times RDY$   | Sender ist zur Aufnahme eines Bytes bereit, wird beim Laden eines Zeichens zurückgesetzt (aktiv «High»).                        |
| $T \times E$     | kein Zeichen zum Senden im Baustein (Sender leer), wird beim Empfang eines Zeichens vom Prozessor zurückgesetzt (aktiv «High»). |
| $R \times RDY$   | Baustein hat ein Zeichen empfangen, wird beim Lesen des Zeichens zurückgesetzt (aktiv «High»).                                  |
| $\overline{DTR}$ | (Data Terminal Ready).<br>$\overline{DTR}$ ist Ausgangssignal und läßt sich durch Programm einstellen (aktiv «Low»).            |

Systembus

Ausgang

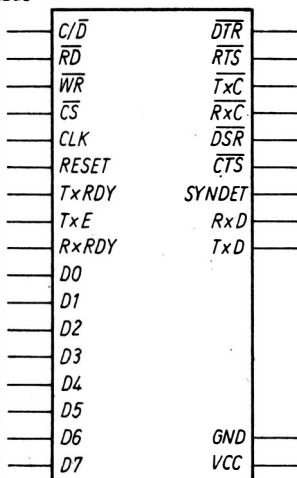


Bild 2.16  
Anschlußbelegung  
des 8251

 $\overline{RTS}$ 

(Request To Send).

$\overline{RTS}$  ist Ausgangssignal und läßt sich durch Programm einstellen (aktiv «Low»).

 $\overline{CTS}$ 

(Clear To Send).

$\overline{CTS}$  ist Eingangssignal und läßt sich vom Programm testen (aktiv «Low»).

 $\overline{DSR}$ 

(Data Set Ready).

$\overline{DSR}$  ist Eingangssignal und läßt sich vom Programm testen (aktiv «Low»).

 $TxC$ 

Sendetakt.

 $RxC$ 

Empfangstakt.

SYNDET

(Synchronisation Detekt)

SYNDET läßt sich als Ausgang oder Eingang programmieren, es ist bei externer Synchronisation «Eingang» und erklärt die Gültigkeit der empfangenen Information;  
der High-Pegel kann nach 1–2 Taktperioden abgeschaltet werden;

bei interner Synchronisation ist es Ausgang, nach Erkennen des Synchronisationszeichens wird SYNDET «High» und nach Zustandslesen «Low» (aktiv «High»).

R×D Empfangseingang.

T×D Sendeausgang.

### 2.6.3. Logische Struktur des Bausteines 8251

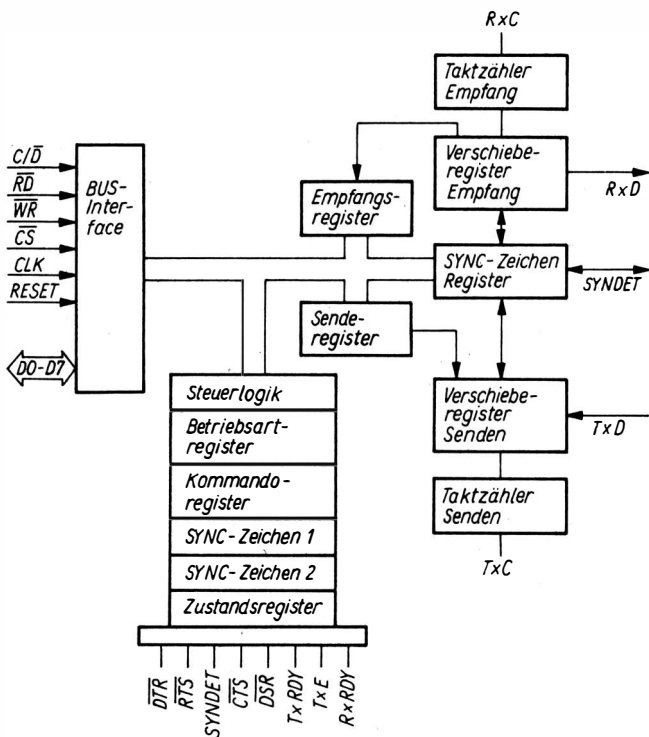


Bild 2.17 Logische Struktur des 8251

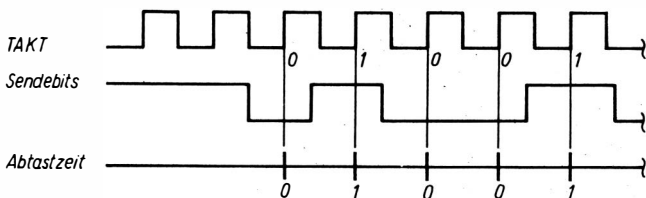


Bild 2.18 Senden und Empfangen von Bits im 8251

#### 2.6.4. Arbeitsweise

Das Senden eines Bits erfolgt mit der H/L-Flanke von  $T \times D$ . Mit der L/H-Flanke von  $R \times C$  werden Empfangsleitungen getestet. Für Kurzschlußbetrieb (Sender mit Empfänger verbunden und  $T \times C$  mit  $R \times C$  verbunden) würde sich Bild 2.18 zum Senden und Empfangen eines Bit-Musters ergeben.

Im Asynchronbetrieb wird ein Datenprotokoll nach Bild 2.14 erzeugt. Zum Senden und Empfangen läßt sich eine Untersetzung  $\times 16$  oder  $\times 64$  programmieren. Damit wird nur jeder 16. oder 64. Takt zur Übertragung ausgewertet.

Im Synchronbetrieb wird ein Datenprotokoll nach Bild 2.15 erzeugt. Eine Untersetzung des Taktes  $T \times C$  oder  $R \times C$  ist nicht möglich. In der Betriebsart interne Synchronisation werden in der Pause SYNC-Zeichen gesendet. Werden die SYNC-Zeichen im Empfänger erkannt, wird das Signal SYNDDET eingeschaltet. Bei externer Synchronisation erfolgt die Synchronisation mit SYNDDET. Ein Eingangsimpuls an SYNDDET gibt den Beginn der Übertragung an.

#### 2.6.5. Programmierung

Zur Programmierung des 8251 ist diese Reihenfolge einzuhalten:

1. Einstellen der Betriebsart.
2. Eingabe der SYNC-Zeichen, wenn die entsprechende SYNC-Betriebsart eingestellt ist. Es werden nur die SYNC-Zeichen angenommen, die von der eingestellten Betriebsart verlangt werden.

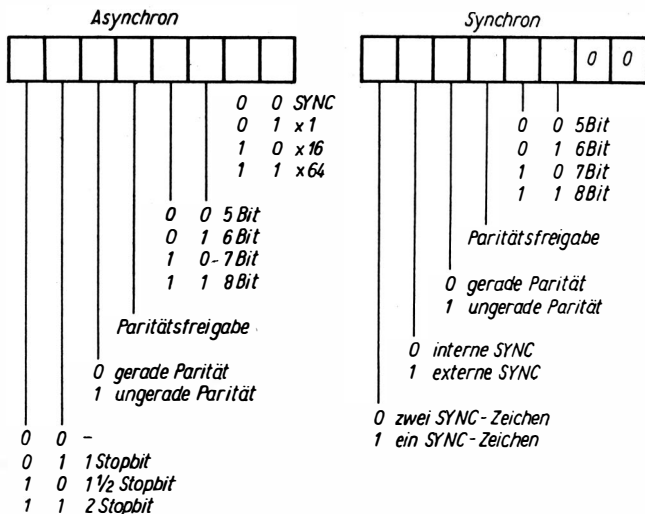


Bild 2.19 Einstellung der Betriebsarten des 8251

3. Alle nun folgenden Steuerworte sind Kommandos. Jede neue Eingabe in den 8251 ändert das Kommando. Eine Rückkehr zum Anfang erfolgt nur mit dem Rücksetzbit im Kommando. Kommandos ( $C/\overline{D} = 1$ ) und Datenworte ( $C/\overline{D} = 0$ ) können sich abwechseln.

### 3. Assemblersprache MAPS K 1520

#### 3.1. Format einer Assembleranweisung

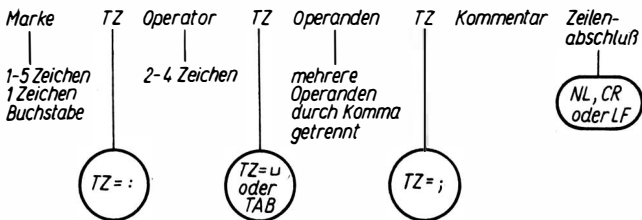


Bild 3.1 Aufbau einer Quellzeile

Quellzeile = 71 Zeichen ohne Zeilenabschluß

- Marke: frei wählbarer Name / 1–5 Zeichen / 1. Zeichen = Buchstabe
- Trennzeichen (TZ) zwischen Marke und Operator: (Doppelpunkt)
- Operator: Festgelegte Bezeichnung für einen Befehl
- Trennzeichen zwischen Operator und Operanden: Leerzeichen oder Tabulator
- Operanden:

|                       | Darstellung                              | Erläuterung<br>(Beispiele)                                   |
|-----------------------|------------------------------------------|--------------------------------------------------------------|
| 1. Zahlen:            | Dezimal                                  | 14                                                           |
|                       | Oktal                                    | 140 oder 14Q                                                 |
|                       | Hexadezimal                              | (die erste Ziffer darf kein Buchstabe A–F sein)<br>25H, 0A3H |
|                       | Binär                                    | 1011B                                                        |
| 2. Symbolische Größen | Entspricht Marke oder definiertem Symbol |                                                              |

- |                                              |                                     |                                                                                   |
|----------------------------------------------|-------------------------------------|-----------------------------------------------------------------------------------|
| 3. aktueller Stand des Speicherplatzzählers: | #                                   | Das Zeichen # entspricht der Adresse in der das 1. Byte des Operationscodes steht |
| 4. Zeichen im SIF-1000-Code:                 | 'A' oder 'AX'                       | statt A bzw. X wird der entsprechende SIF-1000-Code abgespeichert                 |
| 5. Registernamen:                            | einfache Register<br>Doppelregister | A, B, C, usw.<br>DE, HL, IX, usw.                                                 |
| 6. Ausdrücke:                                | Zulässige Operationen + -           |                                                                                   |

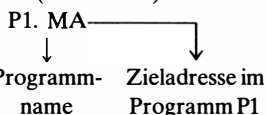
Satz: Die Anzahl der positiven minus der Anzahl der negativen verschiebbaren Symbole (Namen und #) darf nur 0 oder 1 sein.

Beispiele:  $A + 5$  erlaubt (Anzahl = 1)

$A + \# - C$  erlaubt (Anzahl = 1)

$A + \#$  nicht erlaubt (Anzahl = 2)

7. externe Adressen: Beispiel



- Trennzeichen zwischen mehreren Operanden , (Komma)
- Trennzeichen zwischen Operanden und Kommentar ; (Semikolon)
- Kommentar: beliebiger Text, der nicht mit verarbeitet wird
- Zeilenabschluß: Neue Zeile (NL)  
Wagenrücklauf (CR)  
Zeilenvorschub (LF)

### 3.2. Pseudoanweisung der Sprache MAPS K 1520

Pseudoanweisungen sind Anweisungen der Assemblersprache, die sofort bei der Übersetzung ausgeführt werden. Das Ergebnis wird in das Maschinencodeprogramm eingefügt. Die Pseudoanweisungen sind nicht in der Maschinensprache enthalten.

## *Zahlen- und Textdefinitionen*

### **Bytedefinition:**

**DB n** Die Zahl n (8 Bit) wird ins Programm eingefügt.

### **Adressendefinition:**

**DA nn** Die Zahl nn (16 Bit) wird ins Programm eingefügt, niederwertiger Teil zuerst.

### **Textdefinition:**

**DB 'TEXT'** Die Zeichenkette TEXT wird in der Reihenfolge von links nach rechts ins Programm eingefügt.

### **Bereitstellung von RAM-Speicher im Programm:**

**MA: BER nn** Im Programm werden nn Zellen unter der Adresse MA bereitgestellt.

## *Programmorganisation*

**PN name** Anfang eines Programms  
Vom Programmnamen sind nur 2 Zeichen signifikant.

**END** Ende des Programms

**ORG nn** Der Speicherplatzzähler wird auf die Adresse nn gestellt. Die nächste Anweisung kommt in die Zelle nn.

**SYMB: EQU nn** Dem Namen SYMB wird die Zahl nn zugeordnet. (Symboldefinition)  
Diese Zuordnung gilt für das gesamte Programm.

**SYMB: DEF nn** Wie EQU wird dem SYMB die Zahl nn zugeordnet. Die Zuordnung gilt aber nur bis zur nächsten DEF-Anweisung

## *Übersetzungssteuerung*

**IF A** Die Anweisung 1 bis n wird übersetzt, wenn  $A \neq 0$ .

**Anweisung 1** Wenn  $A = 0$ , werden die Anweisungen beim Übersetzen weggelassen

.

.

.

**Anweisung n**

**ENIF**

**TITL 'TEXTKETTE'** Die Textkette erscheint als Überschrift in der Programmliste.

### 3.3. Makroorganisation

Ein MAKRO ist eine Befehlsfolge, die der Assembler bei der Übersetzung in das Programm einsetzt.

Makrodefinition

Ein MAKRO wird definiert durch:

```
NAME:  MACR      Liste der formalen Parameter
       Anweisung 1
       .
       .
       .
       Anweisung n
       ENDM
```

Die formalen Parameter können in den Anweisungen 1 bis n beliebig auftreten. In der Spalte «Marke» steht der Name des MAKRO. Er ist unter gleichen Bedingungen wie eine Marke frei wählbar. In der Spalte «Operator» steht der Pseudobefehl MACR. Er sagt aus, daß die folgenden Anweisungen 1 bis n zu dem MAKRO mit diesem definierten Namen gehören. Die formalen Parameter sind Zeichenkettenparameter, die innerhalb des Makros verwendet werden können.

Beispiel:

Dezimale Addition der Zelleninhalte A1 und A2. Es soll durch das MAKRO mit Namen «DEZA» der Inhalt der Zellen A1 und A2 addiert und in Zelle A1 gespeichert werden.

$(A1) + (A2) \rightarrow (A1)$

Das MAKRO hat folgendes Aussehen:

```
DEZA:  MACR Z1, Z2;    Definitionszeile, Z1 und Z2 sind
                       formale Parameter

      LD A, (Z1)
      LD HL, Z2
      ADC (HL)
      DAA
      LD (Z1), A
      ENDM;            Makrokörper
                       Ende des MAKRO
```

### Makroaufruf:

Der Aufruf erfolgt durch den Namen des MAKRO als Pseudobefehl und der folgenden Liste der aktuellen Parameter. Soll zum Beispiel der Inhalt der Zellen 30H und 31H addiert werden, so geschieht dies durch den Aufruf DEZA 30H, 31H.

Durch den Aufruf wird an die Stelle im Programm, an der der Aufruf steht, ein Programmstück (Makroerweiterung genannt) eingesetzt, in dem die formalen Parameter Z1 und Z2 durch die aktuellen Zeichenketten 30H und 31H ersetzt werden. Das Programmstück hat daher dieses Aussehen:

LD A, (30H)

LD HL, 31H

ADC (HL)

DAA

LD (30H), A

### Sätze:

– Bei der Übersetzung wird statt des Makroaufrufes ein Programmteil eingesetzt, in dem die formellen Parameter der Makrodefinition durch die aktuellen Parameter in der gleichen Reihenfolge ersetzt werden. Den Programmteil nennt man Makroerweiterung. Sind beim Aufruf mehr aktuelle Parameter als formelle vorhanden, werden die überflüssigen weggelassen. Sind beim Aufruf weniger aktuelle Parameter als formelle vorhanden, werden die fehlenden mit «0» belegt.

– Makrobefehle müssen vor ihrem Aufruf definiert sein. Es ist möglich, eine Makrobibliothek anzulegen. Dann genügt es, vor dem Aufruf das Makro aus der Bibliothek ins Programm zu holen.

## 3.4. Gegenüberstellung der Schreibweisen von Befehlen und Pseudobefehlen UDOS/MAPS K 1520

**Tabelle 3.1.** UDOS/MAPS K 1520

| UDOS                 | MAPS K 1520 <sup>1)</sup> |
|----------------------|---------------------------|
| Transportoperationen |                           |
| LD r, (HL)           | LD r, M                   |
| LD (HL), r           | LD M, r (statt (HL); M)   |
| LD (HL), n           | LD M, n                   |
| EXAF, AF'            | EXAF                      |

| UDOS              | MAPS K 1520 <sup>1)</sup> |
|-------------------|---------------------------|
| Rechenoperationen |                           |
| OP A, r           | OP r                      |
| OP A, n           | OP n                      |
| CP s              | CMP s                     |
| Sprünge           |                           |
| JP ADR            | JMP ADR                   |
| JPNZ, ADR         | JPNZ ADR                  |
| JP Z, ADR         | JPZ ADR                   |
| usw.              | usw.                      |
| CALLNZ, ADR       | CANZ ADR                  |
| CALL Z, ADR       | CAZ ADR                   |
| usw.              | usw.                      |
| RET NZ            | RNZ                       |
| RET Z             | RZ                        |
| usw.              | usw.                      |
| Relative Sprünge  |                           |
| JRC, MARKE        | JRC MARKE - #             |
| JRC, \$ + 5       | JRC # + 5                 |
| E/A-Befehle       |                           |
| IN A, (n)         | IN n                      |
| IN r, (C)         | IN r                      |
| OUT (n), A        | OUT n                     |
| OUT (C), r        | OUT r                     |
| Pseudooperationen |                           |
| ORG nn            | ORG nn                    |
| EQU nn            | EQU nn                    |
| DEFL nn           | DEF nn                    |
| DEFB n            | DB n                      |
| DEFW nh           | DA nn                     |
| DEFM 'TEXT'       | TITL 'TEXT'               |
| DEFS nn           | BER nn                    |
| IF nn             | IF nn                     |
| ENDIF             | ENIF                      |
| MARCRu# P0u# P1   | MACR P0, P1               |
| ENDM              | ENDM                      |
| END               | END                       |

<sup>1)</sup> Die Schreibweise der Befehle eines Rechners ist nur vom Übersetzerprogramm, nicht vom Rechner selbst abhängig.

Übersetzerprogramme sind meistens Betriebssystemen zugeordnet. UDOS ist das Betriebssystem der Bürocomputer A 5110 – A 5130, MAPS – K 1520, die Assemblersprache zum Betriebssystem MEOS für den MRES 20.

## 4. Programmaufbereitungssysteme

### 4.1. Schritte der Programmaufbereitung

Hat man ein Programm geschrieben, wird es vor der Verarbeitung durch einen Rechner zunächst auf einen Datenträger (Lochband, Lochkarte, Magnetband usw.) gebracht. Dasso erfaßte Programm ist als Text auf dem Datenträger und heißt Quellcode (QC). Wenn das Programm arbeiten soll, muß es in einer arbeitsfähigen Form im Arbeitsspeicher stehen. Diese Form heißt Maschinencode (MC). Zwischen QC und MC gibt es noch Zwischenformen, so u. a.:

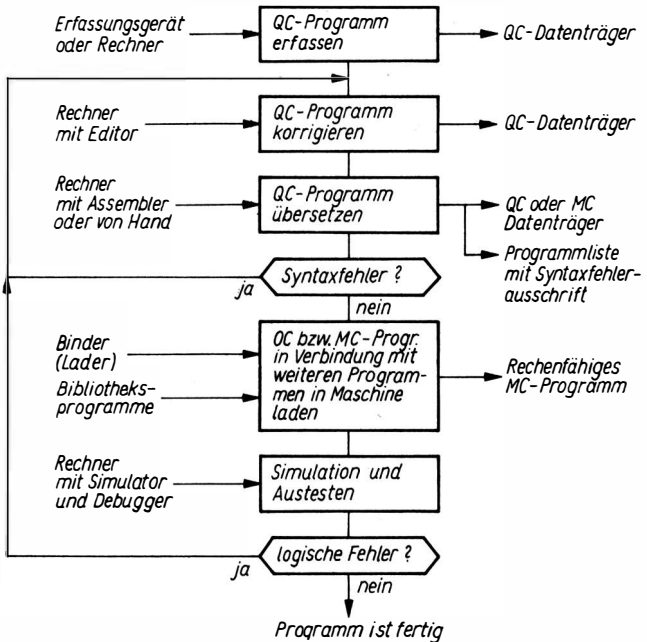


Bild 4.1 Übersicht der Schritte für die Programmaufbereitung und der dazu notwendigen Geräte und Systemprogramme

### Objectcode (OC)

Dieser Code des Programms kann an beliebiger Stelle des Speichers geladen werden. Die Adressen ändern sich entsprechend der Anfangsladeadresse (Leitadresse des Programms).

### Bibliothekscod

Dieser Code ist durch Merkmale versehen, die zur Einordnung in eine Bibliothek benötigt werden.

## 4.2. Systemprogramme zur Programmaufbereitung

### *Editor*

Der Editor ist ein Programm mit dem man einen Quellcode, d. h. einen Text, korrigieren kann. Das Programm enthält Kommandos zum Streichen, Auswechseln und Hinzufügen von Zeichen, Worten und Zeilen. Als Ergebnis entsteht ein neuer Quellcode.

### *Assembler*

Der Assembler ist ein Programm mit dem der Quellcode in den Objektcode oder Maschinencode umgewandelt wird. Zusätzlich entsteht u. a. eine Programmliste.

### *Binder (Linker, Taskbildner)*

Der Binder ist ein Programm mit dem man mehrere Programme (Hauptprogramm, Unterprogramme und Programme aus der Bibliothek) zu einem rechenfähigen Programm zusammenstellen kann.

### *Debugger*

Der Debugger ist ein Programm zum Korrigieren des Maschinencodes. Er enthält Kommandos für das Lesen und Beschreiben von Speicherzellen zur Überprüfung von Programmteilen.

### *Simulator*

Der Simulator ist ein Programm mit dem man ein Maschinenprogramm rechnen lassen kann. Während der Rechnung kann man den Verlauf im Rechner protokollieren.

### *Bibliothekar*

Der Bibliothekar ist ein Programm zur Verwaltung einer Programmbibliothek.

### *Fileprozessor*

Der Fileprozessor ist ein Programm zur Verwaltung und zum Transfer von Filestrukturen. Mit ihm lassen sich Files löschen, kopieren, verändern und hinzufügen.

### *Dienstprogramme*

Dienstprogramme sind kleine Systemprogramme, mit deren Hilfe man spezielle Funktionen ausführen kann;

z. B.

- Doppeln von Lochbändern, Lochkarten, Magnetbändern, Magnetplatten.
- Initialisieren von Magnetplatten.
- Kopieren von Band nach Platte, Speicher nach Platte usw.
- Drucken von Verzeichnissen.

### *Monitor*

Der Monitor ist der Steuerteil des Betriebssystems. Er verwaltet sowohl die einzelnen Systemprogramme und Nutzerprogramme als auch die Ressourcen der Maschine wie Speicher, Ein-/Ausgabegeräte und externe Speicher.

## **5. Basiskonfiguration eines Mikrorechners**

### **5.1. Übersicht**

Ein Mikrorechner dient im Gegensatz zum universellen Klein- oder Großrechner zur Realisierung einer speziellen Aufgabe. Diese Aufgabe kann eine Gerätesteuerung, Meßwerterfassung, Prozeßsteuerung, Konstruktionsplatzunterstützung oder spezielle Numeriklösung sein. Je nach Aufgabenstellung wird eine spezielle Konfiguration zusammengestellt, für die die Software aufgestellt werden muß. Da ein solcher Mikrorechner nicht die universellen Möglichkeiten zur Programmentwicklung hat, wird diese meistens auf einem speziellen Programmentwicklungsrechner (Mikrorechnerentwicklungssystem) oder einem Klein- oder Großrechner (Wirtsrechner, Crossrechner) durchgeführt.

Auf der Mikrorechnerkonfiguration erfolgt dann zum Schluß die Inbetriebnahme der Software und das Suchen der letzten logischen Fehler. Für diesen Zweck muß der Mikrorechner mindestens folgende Funktionen ausführen können:

- Eingabe von Maschinencode-Programmen;
- Eingabe und Ausgabe von Hex-Zahlen in Register und Speicher;
- Starten von Programmen;
- Setzen von Haltepunkten (ein Haltepunkt ist eine Adresse, an der der Programmlauf anhält);
- Einzelschrittbetrieb (Anhalten nach Abarbeitung eines Befehls oder nach einem Maschinenzyklus);
- Simulationsbetrieb (Anzeige der Registerinhalte bei jedem Halt oder Ausdrucken der Registerinhalte bei Halt oder nach bestimmten Befehlen).

Zur Realisierung der Grundfunktion eines Mikrorechners benötigt man zunächst eine Minimalkonfiguration. Sie ist unbedingt notwendig, um mit dem Mikroprozessor einfache Aufgaben lösen zu können.

Zu dieser Minimalkonfiguration gehören (Bild 5.1)

- der Mikroprozessor mit Stromversorgung, Ansteuerung und Taktgenerator;

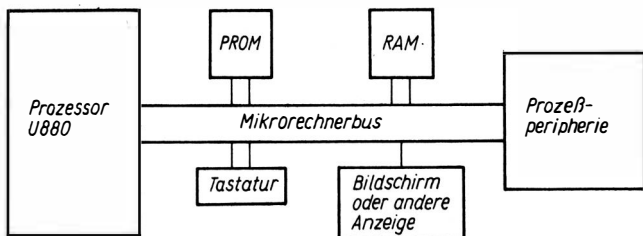


Bild 5.1 Basiskonfiguration eines Mikrorechners

- ein Speicher, bestehend aus ROM- und RAM-Bereich;
- ein Anzeigefeld (hexadezimal oder dual);
- eine Eingabetastatur.

Im folgenden Teil sollen einige Beispiele zum Aufbau der Minimalconfiguration sowie einige Anwendungsbeispiele dieser Konfiguration beschrieben werden.

## 5.2. Steuerung des Prozessors

Ausgehend vom Prozessor legt man zunächst fest, welche Signale zum Systembus gehören sollen. Dazu zählen die Adreßsignale, die Datensignale und die Steuersignale des Prozessors. Bei Minimalconfigurationen kann man dabei einige Signale, die nicht benötigt werden, weglassen (z. B.  $\overline{RFSH}$ , wenn ohne dynamische Speicher gearbeitet wird,  $\overline{BUSRQ}$  und  $\overline{BUSAK}$ , wenn kein DMA-Betrieb vorgesehen ist). Weitere Leitungen können spezielle Funktionen übernehmen, die im Prozessor nicht benötigt werden (z. B. andere Betriebsspannungen, Sperrsignale für den Speicher).

Alle an den Systembus anzuschließenden Funktionseinheiten müssen so aufgebaut sein, daß sie sich mit den Signalen des Systembusses vollständig steuern lassen.

Zur Steuerung des Prozessors benötigt man außer der Betriebsspannung und dem Taktgenerator eine Logik, mit der ein Programm zyklusweise oder befehlsweise abgearbeitet werden kann. Kommen keine dynamischen Speicher zur Anwendung, so läßt sich das sehr einfach über die  $\overline{WAIT}$ -Leitung realisieren. (Bild 5.2) Die Schaltung TG soll als Eintaktgenerator arbeiten. Am Ausgang TG

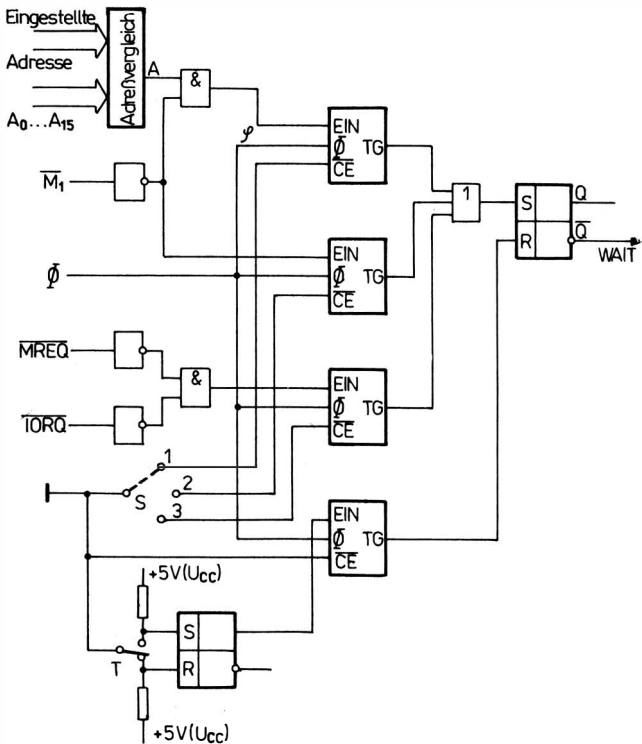


Bild 5.2 Erzeugung des WAIT-Signals zur Realisierung einer taktweisen und befehlswweisen Abarbeitung

wird, wenn  $\overline{CE}$  Tiefpegel hat, der erste Takt  $\Phi$ , nachdem der Eingang EIN Hochpegel erhält, durchgelassen. Die Schaltung «Adreßvergleich» muß so arbeiten, daß am Ausgang A Hochpegel liegt, wenn die eingestellte Adresse mit der an  $A_0$  bis  $A_{15}$  liegenden Adresse übereinstimmt. Mit Hilfe des Schalters S kann man wählen, ob das Programm in jedem Zyklus (Stellung S3), nach jedem Befehl (Stellung S2) oder bei einem Befehl mit eingestellter Adresse (Stellung S1) anhält. In Stellung S3 erzeugt die Schaltung mit jedem MREQ oder IORQ ein WAIT-Signal, d. h., der Prozessor unterbricht bei jedem Speicherzugriffs- und E/A-Zyklus. In

Stellung S2 hält der Prozessor nach dem Erscheinen eines  $\overline{M1}$  und in Stellung S1 nach dem Erscheinen eines  $\overline{M1}$ , wenn gleichzeitig die eingestellte Adresse mit der Busadresse übereinstimmt, an.

### 5.3. Bedienelemente

Die Bedienelemente dienen zum Anzeigen der Register der CPU, Füllen von Speicherplätzen des RAM und Starten eines Programms. Angezeigt bzw. verändert werden können unmittelbar zunächst nur die Informationen auf dem Systembus. Inhalte von Registern oder Speicherzellen müssen zur Anzeige erst auf den Systembus gebracht werden bzw. zur Eingabe vom Systembus geholt

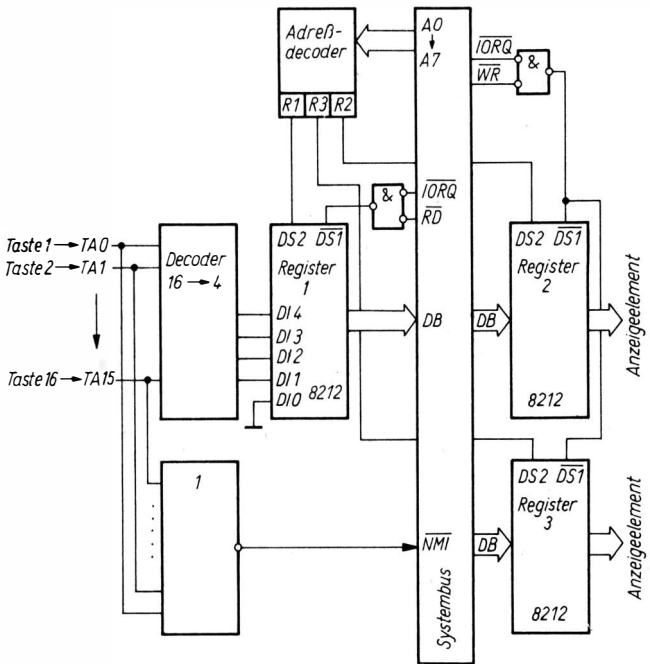


Bild 5.3 Logische Prinzipschaltbild zur Realisierung von Tastenfunktionen am Mikrorechner

werden. Dazu gibt es die Befehle zur Ein- bzw. Ausgabe von Bit-Mustern. Um die oben genannten Funktionen zu realisieren, müssen kleine Programme für diese Funktionen in einem speziellen Speicherbereich (ROM) stehen. Der Zugang zu diesen Programmen ist z. B. über den nichtmaskierten Interrupt oder RESET möglich.

### *Beispiel*

Jeder Funktion (z. B. Anzeige Register A, Füllen Register A) ist ein Programm zugeordnet, das über eine Taste angesprungen wird. Durch das Drücken der Taste wird ein NMI ausgelöst und gleichzeitig die Nummer der Taste in einem Register gespeichert (z. B. 8212). Mit Hilfe der Nummer läßt sich das zur Taste gehörende Programm finden. Bild 5.3 zeigt das logische Prinzipschaltbild für 16 Tastenfunktionen. Beim Drücken einer Taste entsteht eines der Tastensignale TA0 bis TA15. Am Ausgang des «Decoders 16 → 4» entsteht die Nummer der Taste. Sie wird ins Register 1 gebracht. Gleichzeitig entsteht das Signal NMI, das einen Interrupt im Prozessor erzeugt und damit einen Sprung in das Bedienprogramm zum nichtmaskierten Interrupt auslöst. Das Bedienprogramm liest Register 1 und geht damit zum Programm «Anzeige Register A» über. Dieses Programm bringt den Inhalt von Register A zum Register 2 und von da aus zur Anzeige. Je nach Art der Anzeige (dual oder hexadezimal) erfolgt die Auswertung des Inhalts von Register 2. Bei hexadezimaler Anzeige kann der Hexadezimalcode in die beiden Register (Register 2 und Register 3) gebracht werden und von dort über Anzeigeverstärker zu den LED-Elementen. Die folgende Befehlsliste zeigt den Aufbau des Bedienprogramms und des Programms «Anzeige Register A», wenn der Inhalt von Register 2 dual (z. B. durch Leuchtdioden) angezeigt wird.

Zur Aufstellung des Programms sei festgelegt:

Adresse von Register 1: 20H

Adresse von Register 2: 21H

Adresse von Register 3: 22H

Bedienprogramm NMI

PN AW ; Bedienprogramm NMI

ORG 66H ; Startadresse für NMI-Programm

EXAF ; Retten der alten Registerinhalte in den

EXX ; Registern A' bis H'

IN 20H ; Tastennummer in Register DE

```

LD HL, TTARP ; Die Anfangsadresse der Tabelle der
                Adressen für die einzelnen Tastenprogramme
                kommt nach HL
ADD HL, DE    ; In HL steht die Adresse, in der die Startadresse
                des Tastenprogramms steht, in unserem Beispiel
                die Adresse für «Anzeige Register A»

LD E, M
INC HL
LDD, M        ; In DE steht jetzt die Startadresse von «Anzeige
                Register A»

EX DE, HL
JMP M         ; Sprung ins Programm «Anzeige Register A»;
TTAPR: DA ATAO ; Startadresse für Tastenprogramm Taste 1
        DA ATA1 ; Startadresse für Tastenprogramm Taste 2
        .
        .
        .
        DA ATA15 ; Startadresse für Tastenprogramm
                    Taste 16
Programm «Anzeige Register A»
ATAO:OUT21H    ; Inhalt Register A nach Register 2
WS: NOP        ; Warteschleife auf die nächste Tasten-
                funktion

JR WS - #
END

```

## 5.4. Anzeige mit LED-Elementen

Lichtemitteranzeigen mit LED dienen dazu, die im Rechner gespeicherten Zahlen anzuzeigen. Dazu werden in den meisten Fällen 7-Segment-Elemente verwendet. Bild 5.4 zeigt den Anschluß von 6 LED-Anzeigen an den Systembus eines Mikrorechners. Der Systembus erhält die Bussignale eines Prozessors (Adreß-, Daten- und Steuerbus). Zu den einzelnen Anzeigen führen Ausgabetore, über die die Information vom Datenbus zur LED-Anzeige gelangt. Die Ausgabe läßt sich im einfachsten Fall über eine Torschaltung realisieren. Besser ist es jedoch, wenn das zur LED-Anzeige erforderliche Bit-Muster in einem Register zwischengespeichert wird. Dazu eignet sich z. B. der Baustein 8212. Für die Anzeige ist es

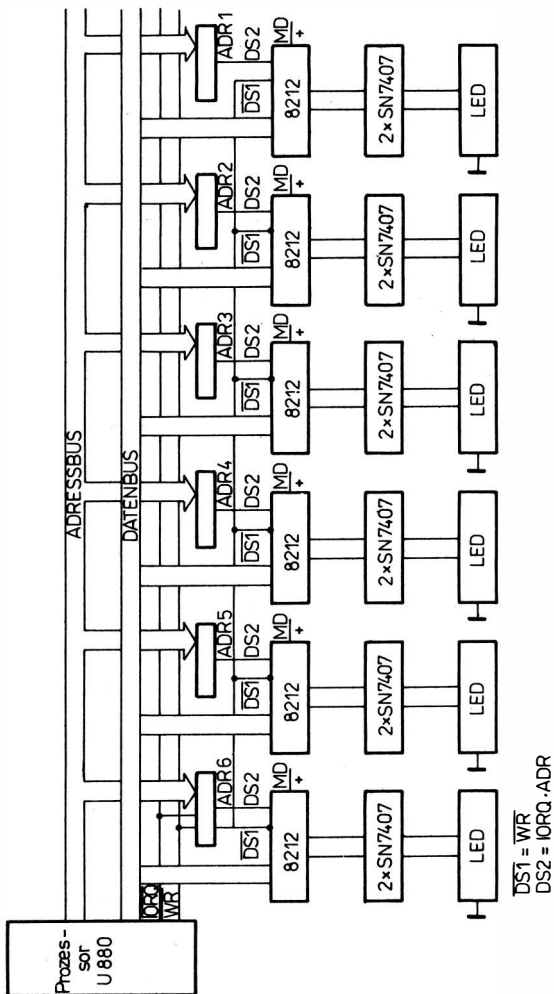


Bild 5.4: Anschluß von 6 LED-Anzeigen an den Systembus eines Mikrorechners

nicht notwendig, daß die ausgegebenen Ziffern über einen BCD-zu-7-Segment-Decoder umgewandelt werden. Statt dessen werden die zur Ansteuerung eines Anzeigeelements notwendigen Bit-Muster vom Prozessor direkt ausgegeben. Die Umwandlung vom BCD-Code in das notwendige Bit-Muster läßt sich dann durch ein Programm erreichen. Bild 5.5 zeigt, wie den einzelnen Segmenten die Bits eines Bytes zugeordnet werden können. Daraus ergibt sich folgende Bit-Muster-Tabelle für die Hexadezimalziffern 0 bis F:

| Zeichen | Bit-Muster (dual) | hexadezimal |
|---------|-------------------|-------------|
| 0       | 0 1 0 1 1 1 1 1   | 5F          |
| 1       | 0 0 0 0 0 1 1 0   | 06          |
| 2       | 0 0 1 1 1 0 1 1   | 3B          |
| 3       | 0 0 1 0 1 1 1 0   | 2E          |
| 4       | 0 1 1 0 0 1 1 0   | 66          |
| 5       | 0 1 1 0 1 1 0 1   | 6D          |
| 6       | 0 1 1 1 1 1 0 1   | 7D          |
| 7       | 0 0 0 0 0 1 1 1   | 07          |
| 8       | 0 1 1 1 1 1 1 1   | 7F          |
| 9       | 0 1 1 0 1 1 1 1   | 6F          |
| A       | 0 1 1 1 0 1 1 1   | 77          |
| B       | 0 1 1 1 1 1 0 0   | 7C          |
| C       | 0 1 0 1 1 0 0 1   | 59          |
| D       | 0 0 1 1 1 1 1 0   | 3E          |
| E       | 0 1 1 1 1 0 0 1   | 79          |
| F       | 0 1 1 1 0 0 0 1   | 71          |

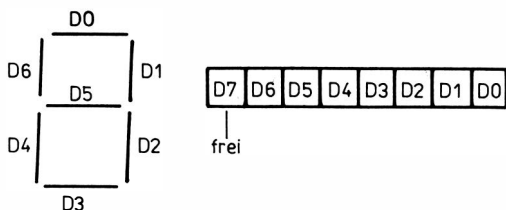


Bild 5.5 Zuordnung der Bit-Stellen eines Bytes zu den Segmenten einer LED-Anzeige

## 5.5. Programm zur LED-Anzeige ohne Konvertierung und Zwischenspeicher

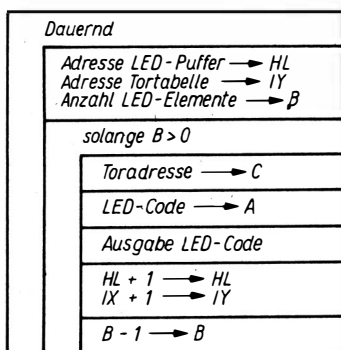
Die zur Anzeige kommenden Daten stehen im LED-Puffer. Für jede LED gibt es eine Zelle des LED-Puffers. Jedem Balken einer LED ist ein Bit dieser Zelle (nach Bild 5.6) zugeordnet. Ist dieses Bit «1», leuchtet der Balken.

Das folgende Programm realisiert die Anzeige des LED-Pufferinhalts auf n-LED-Elementen durch ständige Wiederholung der Datenausgabe (ohne Zwischenpufferung in einem Register).

Dazu sei in HL die LED-Pufferadresse, in IY die Tortabellenadresse und in B die Anzahl der LED-Elemente.

Für das Programm gilt:

a – Struktogramm



b – Assembler-Programm

```

ABS. MACKASSEMBLER K1520 /1 MEOS 1521 V4.2
LOC   OBJ.CODE   STMT   SOURCE PROGRAM
-----84/00/15=====001-----
0001      FN      C5
0002
0003      ;PROGRAMM LED-ANZEIGE OHNE KONVERTIERUNG UND ZWISCHENSPEICHER
0004      ;ADRESSE LED-PUFFER IN HL
0005      ;ADRESSE TORTABELLE IN IY
0006      ;ANZAHL LED-ELEMENTE IN B
0007      LEDA:   PUSH IY
0008              PUSH HL
0009              PUSH BC
0010      ZYK:    LD   C, <IY>
0011              LD   A, <HL>
0012              OUT  C
0013              INC  HL
0014              INC  IY
0015      DJNZ    ZYK-#
0016              POP  BC
0017              POP  HL
0018              POP  IY
0019      JR      LEDA-#
0020      END
0015      PROGRAM CONTAINS 0000 ERROR(S)
  
```

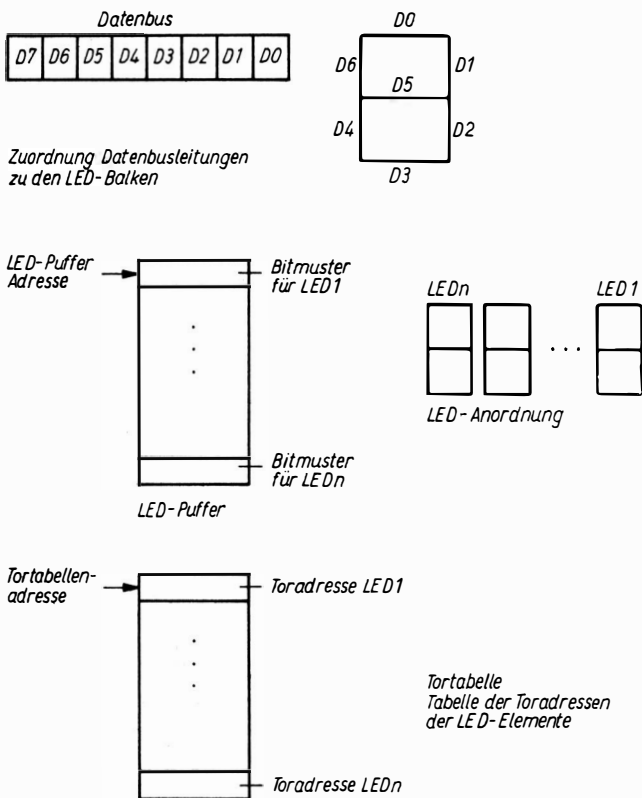


Bild 5.6 Schema zur Programmierung einer LED-Anzeige

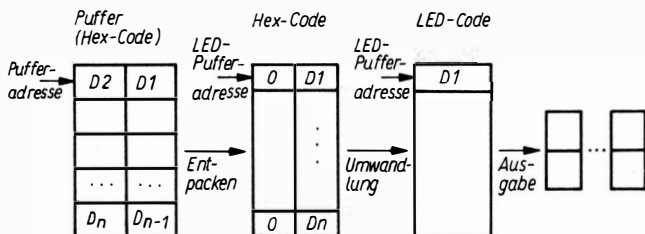


Bild 5.7 Schritte zur hexadezimalen Anzeige eines Puffers auf LED-Elementen

Um den Inhalt eines Speicherpuffers auf LED-Elementen hexadecimal anzuzeigen, müssen die Daten folgende Schritte durchlaufen. (Bild 5.7)

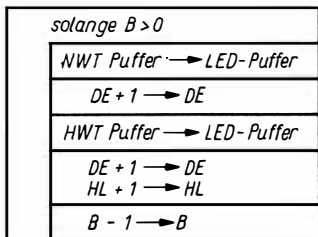
Das Programm zum Entpacken der Daten hat mit  
Pufferadresse in HL

LED-Pufferadresse in DE und

Anzahl Pufferzellen in B

folgende Form:

a – Struktogramm



b – Assembler-Programm

; Entpacken von HEX-Zahlen

; Pufferadresse in HL

; LED-Pufferadresse in DE

; Anzahl Pufferzellen in B

ENTP: XOR A

ZYK: RRD ; NWT Puffer → A

LD (DE), A

INC DE

RRD

; HWT Puffer → A

LD (DE), A

RRD

; Pufferinhalt wiederherstellen

INC DE

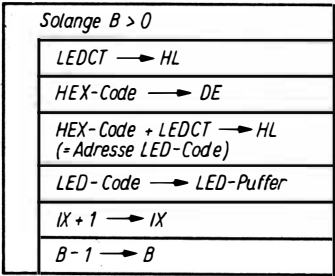
INCHL

DJNZ ZYK - #

RET

Die Umwandlung von HEX-Code in den LED-Code kann im LED-Puffer erfolgen. Dazu benötigen wir eine Tabelle, in der der LED-Code für die 16-HEX-Ziffern 0–F der Reihe nach gespeichert ist. Die Anfangsadresse dieser Tabelle habe den Namen

LEDCT. Steht die Anzahl der LED-Elemente in B und die Adresse des LED-Puffers in IX, so gilt:  
a – Struktogramm



b – Assembler-Programm

```

ABS. MACROASSEMBLER K1520 /1 M6801521 U4.2
L OC   OBJ.CODE   STMT   SOURCE PROGRAM
=====
0001      FN      CG
0002
0003      ;LED-AUFBEREITUNG FUER HEX-ZAHLEN
0004      ;UMWANDLUNG HEX-CODE IN LED-CODE
0005      ;ADRESSE DATENPUFFER IN IX
0006      ;ANZAHL DER ELEMENTE IN B
0007 UHL: LD      HL,LEDCT      ;TABELLE LED CODE
0008      LD      E,(IX)
0009      LD      D,0
0010      ADD     HL,DE
0011      LD      A,(HL)
0012      LD      (IX),A
0013      INC     IX
0014      DJNZ   UHL,-#
0015      RET
0016 LEDCT: EQU 0      ;ADRESSE FUER LED-CODE TABELLE
0017      ;DIE ADRESSE IST ANZUGEBEN
0018      END
0012      PROGRAM CONTAINS 0000 ERROR(S)

```

Eine einfache Möglichkeit eine Dualzahl auf einer LED-Anzeige darzustellen, ergibt sich durch spezielle Auswahl der Balken. (in Bild 5.8 stark gezeichnet)

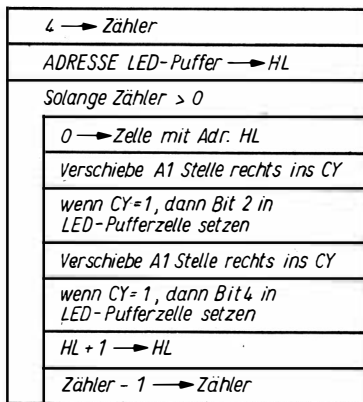
Soll z. B. der Inhalt von Register A auf dieser LED-Anzeige angezeigt werden, ist der LED-Puffer mit 4.Bit-Mustern so aufzubereiten, daß die Bit-Stellen für die zu leuchtenden Balken «1» gesetzt sind.



Bild 5.8  
Auswahl von Balken einer  
4stelligen LED-Anzeige  
zur Darstellung einer Dualzahl

## Lösung:

### a – Struktogramm



### b – Assembler-Programm

```

=====04/08/15==001=====
      ABSL MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
      LOC  OBJ CODE  STMT  SOURCE PROGRAM
0000 06 04 0001 FN C7
0002 36 00 0002 ;LED-AUFBEREITUNG
0004 1F 0003 ;BITMUSTER VON REGISTER A WIRD AUF 4LED-ELEMENTEN ANGEZEIGT
0005 30 02 0004 ;BITMUSTER IN REGISTER A
0007 CB D6 0005 ;ADRESSE LED-PUFFER IN HL
0009 1F 0006 DUALA: LD E,4
000A 30 02 0007 ZYK: LD <HL>,0
000C CB E6 0008 RRA
000E 23 0009 JRNC M1-#
000F 10 F1 0010 SET 2,<HL>
0011 C9 0011 M1: RRA
0012 0012 JRNC M2-#
0013 0013 SET 4,<HL>
0014 0014 M2: INC HL
0015 0015 DJNZ ZYK-#
0016 0016 RET
0017 0017 END
0018 0018
PROGRAM CONTAINS 0000 ERROR(S)

```

Nach Setzen der Bitstellen im LED-Puffer muß zur Anzeige das Programm «LED-Anzeige ohne Konvertierung» aufgerufen werden.

## 6. Programmbeispiele

### 6.1. Numerische Programme und Programme zur Zahlenumwandlung

Die Darstellung der Zahlen im Mikrorechner wird durch den Anwendungsfall bestimmt. Sie wird durch das betreffende Programm definiert. Dabei treten oft rein duale Darstellungen in Verbindung mit BCD-Darstellungen oder Textdarstellungen (ASCII-Code) auf. Zur Realisierung einheitlicher Verarbeitung sind Umwandlungen von einer in die andere Darstellung unumgänglich. Im folgenden werden einige solcher Umwandlungen und Operationen für die betreffenden Zahlendarstellungen gezeigt.

#### Umwandlung BCD-DUAL

Bei der Umwandlung BCD (Dezimal) – DUAL unterscheidet man u. a. die Verfahren für den ganzen Teil und den gebrochenen Teil der Zahl. Daher erfolgt die Umsetzung von ganzem und gebrochenem Teil meistens getrennt.

#### *Beispiel 1*

Umwandlung einer ganzen BCD-Zahl (4 Stellen ohne Vorzeichen) in eine Dualzahl

Eingangsparameter: BCD-Zahl in HL

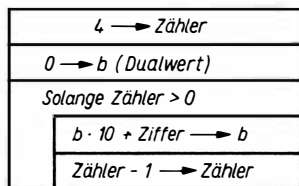
Ausgangsparameter: DUAL-Zahl in HL

Verfahren:

Schreibt man die Dezimalzahl mit den Ziffern  $d_3 d_2 d_1 d_0$  in der Form

$$\begin{aligned} b &= d_3 \cdot 10^3 + d_2 \cdot 10^2 + d_1 \cdot 10^1 + d_0 = \\ &= (((0 \cdot 10 + d_3) \cdot 10 + d_2) \cdot 10 + d_1) \cdot 10 + d_0 \end{aligned} \quad (2)$$

so erkennt man folgenden Algorithmus:



Dabei ist mit der höchstwertigen Ziffer ( $d_3$ ) zu beginnen. Vor dem Abarbeiten des Algorithmus sind die Ziffern von der Zahl abzutrennen.

```

ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
LOC  OBJ.CODE  STMT  SOURCE PROGRAM
=====84/07/26=====001=====
0001          FN  A1
0002          ;UMWANDLUNG BCD-DUAL
0003          ;UMWANDLUNG EINER INTEGER BCD-ZAHL (4 STELLEN UNSIGNIERT)
0004          ;IN EINE DUALZAHL (16 BIT)
0005
0000 E5        0006 DB12: PUSH HL
0001 21 00 00  0007 LD HL,0
0004 39        0008 ADD HL,SP
0005 23        0009 INC HL           ;HL ZEIGT AUF BCD-ZAHL
0006 AF        0010 XOR A           ;ABLEGEN DER ZIFFERN D3,D2,D1,D0
0007 ED 67     0011 RRD             ;IN DEN STACK
0009 F5        0012 PUSH AF         ;ZIFFER D0 IN DEN STACK
000A ED 67     0013 RRD
000C F5        0014 PUSH AF         ;ZIFFER D1 IN DEN STACK
000D 23        0015 INC HL
000E ED 67     0016 RRD
0010 F5        0017 PUSH AF         ;ZIFFER D2 IN DEN STACK
0011 ED 67     0018 RRD
0013 F5        0019 PUSH AF         ;ZIFFER D3 IN DEN STACK
0014 21 00 00  0020 ;ALGORITHMUS
0017 06 04     0021 LD HL,0
0019 5D        0022 LD B,4
001A 54        0023 LD E,L           ;ZAEHLER =4
001B 29        0024 LD D,H
001C 29        0025 ADD HL,HL       ;MULTIPLIKATION MIT 10
001D 19        0026 ADD HL,HL
001E 29        0027 ADD HL,DE
001F D1        0028 ADD HL,HL
0020 5A        0029 POP DE           ;ZIFFER HOLEN
0021 16 00     0030 LD E,D
0023 19        0031 LD D,0
0024 10 F3     0032 ADD HL,DE       ;ZIFFER ADDIEREN
0026 D1        0033 DJNZ ZYK-#
0027 C9        0034 POP DE
0028          0035 RET
0028          0036 END
PROGRAM CONTAINS 0000 ERROR(S)

```

### Beispiel 2

Umwandlung einer gebrochenen BCD-Zahl (2 Stellen ohne Vorzeichen) in eine Dualzahl mit 8 Stellen.

Eingangsparameter: BCD-Zahl in Register A

Ausgangsparameter: Dual-Zahl in Register A

Verfahren:

Durch Multiplikation mit 2 erscheinen die Dualziffern der Reihe nach vor dem Komma.

Für 8 Dualstellen ergibt sich folgender Algorithmus:

|                                     |
|-------------------------------------|
| 8 → Zähler                          |
| Solange Zähler > 0                  |
| BCD-Zahl × 2 → BCD-Zahl             |
| Bitstelle vor dem Komma<br>abnehmen |
| Zähler - 1 → Zähler                 |

Durch  $(\text{BCD-Zahl} \cdot 2 = \text{BCD-Zahl} + \text{BCD-Zahl})$  wird die Bit-Stelle vor dem Komma in das CY-Bit gebracht. Diese Stellen brauchen nur noch in ein freies Register (z. B. Register C) der Reihe nach von rechts nach links eingeschoben werden.

```

ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
LOC  OBJ. CODE  STMT  SOURCE PROGRAM
=====84/07/26=====001=====
0001      FN      A2
0002      ;UMWANDLUNG EINER BCD-ZAHL /2<1 (2 STELLEN UNSIGNED)
0003      ;IN EINE DUALZAHL (8 BIT)
0004
0005  DB11: LD     B,B      ;ZAELER=0
0006  MA1:  ADD    A        ;WERT#2
0007      DAA
0008      RL      C
0009      DJNZ   MA1,##      ;RUECKSPRUNG SOLANGE ZAELER>0
000A      LD     A,C        ;RESULTAT NACH A
000B      RET
000C      END
PROGRAM CONTAINS 0000 ERROR(S)

```

### Umwandlung DUAL-BCD

Auch hier wird i. a. der ganze und der gebrochene Teil einer Zahl getrennt behandelt.

#### Beispiel 3

Umwandlung einer ganzen Dualzahl (16 Bit) in eine BCD-Zahl mit 5 Stellen.

Eingangsparameter: Dualzahlen in HL

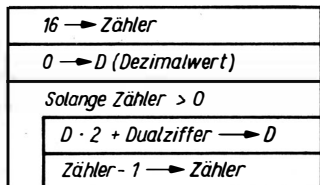
Ausgangsparameter: BCD-(Dezimal-)Zahl in AHL

Verfahren:

Schreibt man die Dualzahl mit den Ziffern  $b_{15} b_{14} b_{13} \dots b_1 b_0$  in der Form

$$\begin{aligned}
 D &= b_{15} \cdot 2^{15} + b_{14} \cdot 2^{14} + \dots + b_1 \cdot 2^1 + b_0 \\
 &= (\dots((0 \cdot 2 + b_{15}) \cdot 2 + b_{14}) \cdot 2 + \dots + b_1) \cdot 2 + b_0, \quad (3)
 \end{aligned}$$

so erkennt man folgenden Algorithmus:



Die Dualziffern müssen mit der höchstwertigen beginnen. Durch Dualwert + Dualwert (HL + HL) kommen die Dualziffern der Reihe nach in das CY-Flag, das nur noch zu 2D addiert zu werden braucht.

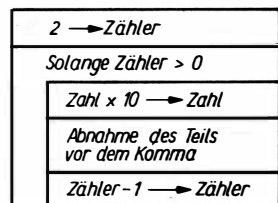
| LOC  | OBJ.CODE | ABS.                           | MACROASSEMBLER K1520 /1 MEOS 1521 V4.2               | SOURCE PROGRAM |
|------|----------|--------------------------------|------------------------------------------------------|----------------|
|      |          | 0001                           | FN                                                   | A3             |
|      |          | 0002                           | ; DUAL-BCD                                           |                |
|      |          | 0003                           | ; UMWANDLUNG EINER DUALZAHL (INTEGER 16BIT UNSIGNED) |                |
|      |          | 0004                           | ; IN EINE BCD-ZAHL (5 STELLEN)                       |                |
|      |          | 0005                           | ; DUALZAHL IN HL                                     |                |
|      |          | 0006                           | ; BCD-ZAHL IN AHL                                    |                |
|      |          | 0007                           |                                                      |                |
| 0000 | AF       | 0008                           | BDI2:                                                | XOR A          |
| 0001 | 57       | 0009                           |                                                      | LD D,A         |
| 0002 | 06 10    | 0010                           |                                                      | LD B,16        |
| 0004 | 29       | 0011                           | ZYK:                                                 | ADD HL,HL      |
| 0005 | 8F       | 0012                           |                                                      | ADC A          |
| 0006 | 27       | 0013                           |                                                      | DAA            |
| 0007 | 5F       | 0014                           |                                                      | LD E,A         |
| 0008 | 7A       | 0015                           |                                                      | LD A,D         |
| 0009 | 8F       | 0016                           |                                                      | ADD A          |
| 000A | 27       | 0017                           |                                                      | DAA            |
| 000B | 57       | 0018                           |                                                      | LD D,A         |
| 000C | CB 11    | 0019                           |                                                      | RL C           |
| 000E | 7B       | 0020                           |                                                      | LD A,E         |
| 000F | 10 F3    | 0021                           |                                                      | DJNZ ZYK-1     |
| 0011 | EB       | 0022                           |                                                      | EX DE,HL       |
| 0012 | 79       | 0023                           |                                                      | LD A,C         |
| 0013 | C9       | 0024                           |                                                      | RET            |
| 0014 |          | 0025                           |                                                      | END            |
|      |          | PROGRAM CONTAINS 0000 ERROR(S) |                                                      |                |

#### Beispiel 4

Umwandlung einer gebrochenen Dualzahl mit 8 Bit ohne Vorzeichen in eine BCD-Zahl mit 2 Stellen.

Eingangsparameter: Dualzahl in Register A

Ausgangsparameter: BCD-Zahl in Register A



**Verfahren:** Multipliziert man den gebrochenen Teil einer Zahl mit 10, so tritt die vorderste Dezimalziffer vor das Komma.

```

ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
LOC  OBJ.CODE  STMT      SOURCE PROGRAM
=====04/07/26==001=====
0001      FN      A4
0002      ;UMWANDLUNG EINER DUALZAHLECHT GEBROCHEN 8 BIT
0003      ;IN EINE DEZIMALZAHLE 2 STELLEN
0004      ;DEZIMALZAHLE UND DUALZAHLE IN A
0005
0006      UDDE1: LD    D,A      ;DUALZAHLE IN D
0007      LD    H,A      ;DUALZAHLE IN H
0008      LD    L,0
0009      LD    E,0
0010      XOR    A
0011      LD    B,2      ;ZAEHLER=2
0012      ZVK:  ADD   HL,HL    ;MULTIPLIKATION MIT 10
0013      ADC    A
0014      ADD   HL,HL
0015      ADC    A
0016      ADD   HL,DE
0017      ADC    0
0018      ADD   HL,HL
0019      ADC    A
0020      DEC    B
0021      RZ
0022      RLA      ;1 ZIFFER IN VORDERE 4 STELLEN VON A
0023      JR     ZVK-#
0024      END
PROGRAM CONTAINS 0000 ERROR(S)

```

### Beispiel 5

Umwandlung einer echt gebrochenen Dualzahl mit 16 Bit ohne Vorzeichen in eine BCD-Zahl mit 4 Stellen.

Eingangsparameter: Dualzahl in HL

Ausgangsparameter: BCD-Zahl in HL

Das Programm läuft nach dem gleichen Verfahren wie Beispiel 4, nur wird der Zähler am Anfang auf 4 gestellt.

```

ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
LOC  OBJ.CODE  STMT      SOURCE PROGRAM
=====04/07/31==001=====
0001      FN      A5
0002      ;UMWANDLUNG DUAL NACH BCD
0003      ;16 BIT DUALZAHLE UNSIGNIERT ERGIBT 4-STELLIGE BCD-ZAHLE(ECHT GEBROCHEN)
0004      ;DUALZAHLE IN HL
0005      ;BCD-ZAHLE IN HL
0006
0007      BDE2: LD    B,4      ;FUEH 4 BCD-STELLEN
0008      XOR    A
0009      LD    C,A
0010      ZVK:  LD    E,L
0011      LD    D,H
0012      ADD   HL,HL
0013      RLA      ;REGISTER A UND C UEBERNEHMEN
0014      RL     ;DIE DEZIMALSTELLEN
0015      RL     C
0016      ADD   HL,DE
0017      RLA
0018      RL     C
0019      ADD   HL,DE
0020      ADC    0
0021      ADD   HL,HL
0022      RLA
0023      RL     C
0024      DEC    B
0025      JRZ   MO-#
0026      SLA    A      ;DIE SCHON ERRECHNETEN STELLEN WERDEN
0027      ;IN DIE RICHTIGE STELLUNG GEBRACHT
0028      RL     C
0029      JR     ZVK-#
0030      MO:  LD    L,A
0031      LD    H,C
0032      RET
0033      END
PROGRAM CONTAINS 0000 ERROR(S)

```

## Beispiel 6

Zur Ausgabe einer Zahl auf Bildschirm oder Drucker muß die Zahl im ASCII-Code vorliegen.

Lösung:

*Umwandlung DUAL-BCD  
und Abspeichern in der  
auszugebenden Reihenfolge*

*BCD-Ziffer + 30H → ASCII-Ziffer*

*Speichern in Ausgabepuffer*

| LOC                            | OBJ.CODE | STMT | SOURCE PROGRAM                                    |
|--------------------------------|----------|------|---------------------------------------------------|
| =====84/87/31=====001=====     |          |      |                                                   |
|                                |          | 0001 | FN AG                                             |
|                                |          | 0002 |                                                   |
|                                |          | 0003 | ;UMWANDLUNG EINER GANZEN POSITIVEN DUALZAHL       |
|                                |          | 0004 | ;IN DIE ASCII-DUALSTELLUNG                        |
|                                |          | 0005 | ;EINGABEPARAMETER: DUALZAHL IN HL ADRESSE ASCII - |
|                                |          | 0006 | ;PUFFER IN DE                                     |
|                                |          | 0007 | ;AUSGABEPARAMETER: ASCII-ZAHL IM PUFFER           |
| 0000                           | D5       | 0008 | DAS2: PUSH DE                                     |
|                                |          | 0009 | ;;PROGRAMMSCHNITT 1##                             |
|                                |          | 0010 | ;DUAL-BCD                                         |
| 0001                           | 3E 00    | 0011 | LD A, 0                                           |
| 0003                           | 57       | 0012 | LD D, A                                           |
| 0004                           | 06 10    | 0013 | LD B, 16                                          |
| 0006                           | 29       | 0014 | ZYK: ADD HL, HL                                   |
| 0007                           | 8F       | 0015 | ADC A                                             |
| 0008                           | 27       | 0016 | DAA                                               |
| 0009                           | 5F       | 0017 | LD E, A                                           |
| 000A                           | 7A       | 0018 | LD A, D                                           |
| 000B                           | 8F       | 0019 | ADC A                                             |
|                                |          | 0020 |                                                   |
| 000C                           | 27       | 0021 | DAA                                               |
| 000D                           | 57       | 0022 | LD D, A                                           |
| 000E                           | 0B 11    | 0023 | RL C                                              |
| 0010                           | 7B       | 0024 | LD A, E                                           |
| 0011                           | 10 F3    | 0025 | DJNZ ZYK-#                                        |
| 0013                           | 79       | 0026 | LD A, C                                           |
| 0014                           | EB       | 0027 | EX DE, HL                                         |
|                                |          | 0028 | ;;PROGRAMMSCHNITT 2##                             |
|                                |          | 0029 | ;UMWANDLUNG BCD-ASCII                             |
| 0015                           | DD E1    | 0030 | POP IX                                            |
| 0017                           | C6 30    | 0031 | ADD 30H                                           |
| 0019                           | DD 77 00 | 0032 | LD <IX>, A                                        |
| 001C                           | 06 04    | 0033 | Z0: LD B, 4                                       |
| 001E                           | DD 23    | 0034 | Z1: INC IX                                        |
| 0020                           | AF       | 0035 | XOR A                                             |
| 0021                           | 0E 04    | 0036 | LD C, 4                                           |
| 0023                           | 29       | 0037 | ADD HL, HL                                        |
| 0024                           | 8F       | 0038 | ADC A                                             |
| 0025                           | 0D       | 0039 | DEC C                                             |
| 0026                           | 20 F6    | 0040 | JRNZ Z1-#                                         |
| 0028                           | C6 30    | 0041 | ADD 30H                                           |
| 002A                           | DD 77 00 | 0042 | LD <IX>, A                                        |
| 002D                           | 10 ED    | 0043 | DJNZ Z0-#                                         |
| 002F                           | C9       | 0044 | RET                                               |
| 0030                           |          | 0045 | END                                               |
| PROGRAM CONTAINS 0000 ERROR(S) |          |      |                                                   |

## 6.2. Rechenprogramme

Im Befehlsschlüssel des U880 sind keine Multiplikation und Division enthalten. Diese Operationen müssen durch Programme realisiert werden. Das betreffende Programm hängt natürlich von der Darstellung der Zahlen ab.

### Beispiel 7

Multiplikation von 2 ganzen Dualzahlen (16 Bit) ohne Vorzeichen.  
Das Ergebnis ist 32 Bit lang.

Eingangsparameter: Multiplikand in DE  
Multiplikator in BC

Ausgangsparameter: Produkt in HL BC

Verfahren:

Schreiben wir den Multiplikator (MR) in dualer Darstellung

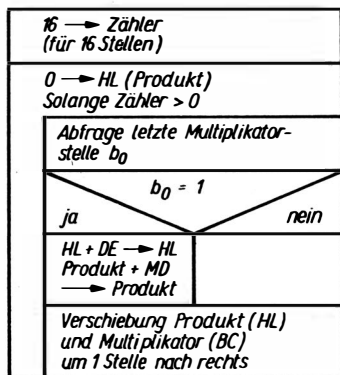
$$MR = b_{15} \cdot 2^{15} + b_{14} \cdot 2^{14} + \dots b_1 \cdot 2^1 + b_0, \quad (4)$$

so erkennen wir aus

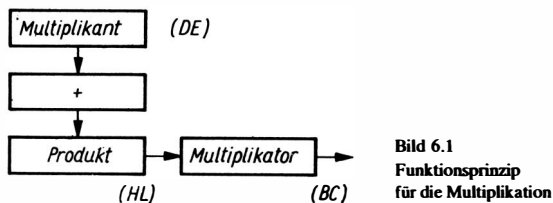
$$\begin{aligned} MD \cdot MR &= MD (b_{15} \cdot 2^{15} + b_{14} \cdot 2^{14} + \dots b_1 \cdot 2^1 + b_0) \\ &= 2^{15} b_{15} MD + 2^{14} b_{14} MD + \dots \\ &\quad \dots 2^1 b_1 MD + b_0 MD, \end{aligned} \quad (5)$$

daß wir zur Produktbildung MD mit einer Verschiebung  $2^i$  zum Teilprodukt addieren müssen, wenn die Stelle  $b_i = 1$  ist. Statt einer Verschiebung von MD nach links ( $\times 2^i$ ) können wir auch das Produkt nach rechts verschieben. Die Multiplikation läuft damit nach Schema (Bild 6.1) ab.

Der Ablauf läßt sich wie folgt darstellen:



Da sich die Abfrage der letzten Stelle des Multiplikators  $b_0$  leichter über das CY-Bit realisieren läßt, kann man vor Beginn des Verfah-



rens den Multiplikator so um eine Stelle nach rechts schieben, daß die letzte Stelle ins CY-Bit kommt. Dadurch entsteht der als Flußbild (Bild 6.2) dargestellte Ablauf.

Zur Multiplikation wird der Multiplikator eine Stelle nach rechts geschoben. Dabei gelangt das niederwertigste Bit in das C-Bit. Ist dieses Bit 1, wird der Inhalt von DE zu HL addiert, ist es 0, erfolgt keine Addition von DE zu HL. Anschließend wird der Inhalt von HL mit BC gemeinsam um eine Stelle nach rechts verschoben. Dabei kommt das nächste Bit des Multiplikators ins CY-Bit. Dieser Vorgang wird insgesamt 16mal wiederholt, da der Multiplikator 16 Stellen umfaßt.

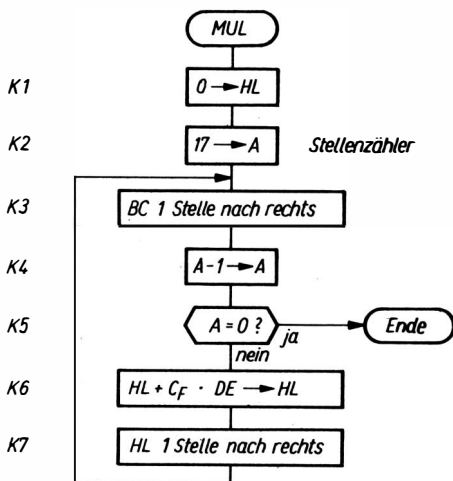


Bild 6.2 Flußdiagramm für die Multiplikation von 2 16stelligen Dualzahlen

Weil am Anfang einmal BC nach rechts verschoben werden muß, um die 1. Stelle ins CY-Bit zu bringen, wird der Zähler auf 17 gestellt.

```

ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
LOC  OBJ.CODE  STMT  SOURCE PROGRAM
=====84/07/27=====001=====
0001      FN      B1
0002      ;BASISPROGRAMM FUER MULTIPLIKATION
0003      ;16 X 16 BIT -- MULTIPLIKATION, UNSIGNIERT, ERGEBNIS 32 BIT
0004      ;MULTIPLIKANT IN DE
0005      ;MULTIPLIKATOR IN BC
0006
0007      MUL2:   LD      HL,0
0008      LD      A,17      ;ZAEHLER
0009      K3:     RR      B
0010      RR      C
0011      DEC     A
0012      RZ
0013      JNOC    K7-#
0014      ADD     HL,DE
0015      K7:     RR      H
0016      RR      L
0017      JR      K3-#
0018      END
PROGRAM CONTAINS 0000 ERROR(S)

```

## Division

Die Division ist die Umkehrung der Multiplikation. Es soll eine Division mit einem Dividenden (DV) von 32 Bit und einem Divisor (DR) von 16 Bit betrachtet werden. Der Quotient (Q) möge 16 Bit sein.

Dann gilt

$$DV : DR = Q \text{ Rest } R$$

$$\text{oder } \frac{DV}{DR} = Q + \frac{R}{DR} \quad (6)$$

Bringt man die Gleichung auf die Form

$$R = DV - Q \cdot DR \quad (7)$$

und schreibt den Quotienten Q als Dualzahl

$$Q = Q_{15} \cdot 2^{15} + Q_{14} \cdot 2^{14} + \dots + Q_1 \cdot 2^1 + Q_0, \quad (8)$$

so kann man unser Verfahren aus der Gleichung

$$R = DV - 2^{15} Q_{15} DR - 2^{14} Q_{14} DR - \dots - 2^1 Q_1 DR - Q_0 DR \quad (9)$$

ableiten.

Man probiert, ob  $2^{15} \cdot DR$  vom Dividenden DV abgezogen geht. Wenn «ja», ist  $Q_{15} = 1$ , und es werden  $2^{15} \cdot DR$  abgezogen; wenn «nein», ist  $Q_{15} = 0$ , und  $2^{15} \cdot DR$  wird nicht abgezogen. Danach verfährt man ebenso mit  $2^{14} \cdot DR$  bis  $2^0 \cdot DR$ . Für die Division ergibt sich damit folgendes Blockscheema (Bild 6.3).

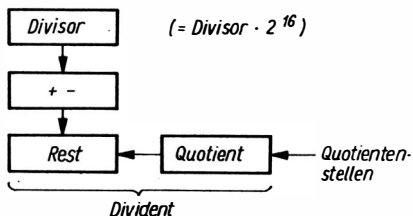


Bild 6.3  
Funktionsprinzip  
für die Division

### 1. Bildung von Dividend – Divisor

Ist das Ergebnis negativ, so wird der Divisor wieder zum Ergebnis dazugezählt (Rückstellung des Restes) und in die letzte Stelle des Quotienten eine 0 eingetragen.

Ist das Ergebnis positiv, so wird keine Rückstellung des Restes vorgenommen. In die letzte Quotientenstelle trägt man eine 1 ein.

2. Quotient und Dividend werden gemeinsam 1 Stelle nach links verschoben.

3. 1. und 2. werden so oft wiederholt, wie der Quotient Stellen haben soll.

Am Ende steht der Quotient im Quotientenregister und im Dividentenregister der Rest.

Da im Divisorregister der  $\text{Divisor} \cdot 2^{16}$  steht und wir mit einem 16-Bit-Divisor arbeiten, müssen Punkt 1 und 2 einmal zusätzlich durchlaufen und die nach der ersten Linksverschiebung aus dem Restregister austretende vorderste Stelle in einem weiteren Register aufgefangen werden.

### Beispiel 8

Division einer ganzen Zahl (32 Bit) durch eine ganze Zahl (16 Bit). Der Quotient soll 16 Bit Datenbreite haben.

Eingangsparameter: Divident in HL BC  
Divisor in DE

Ausgangsparameter: Quotient in BC

Bild 6.4 zeigt das Divisionsschema, wenn wir dem Dividenten, dem Divisor und dem Quotienten bestimmte Register zuweisen.

Als zusätzliches Register zum Auffangen der vordersten Dividentenstelle nehmen wir Register A'. Im Flag entsteht beim Addieren oder Subtrahieren das Vorzeichen des Restes. Ist dieses Vorzeichen 1, ist die Quotientenstelle 0 und umgekehrt. Daher können

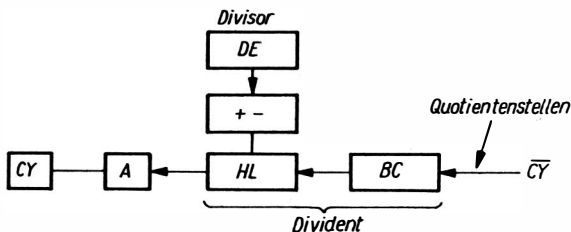


Bild 6.4 Blockschema für Division 32 Bit: 16 Bit → 16 Bit ohne Vorzeichen mit U 880

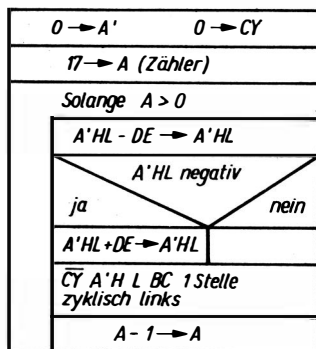


Bild 6.5  
Struktogramm für die Division

wir  $\overline{CY}$  als Quotientenstelle benutzen. Bild 6.5 zeigt den Ablauf als Struktogramm.

```

ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
LOC  OBJ.CODE  STMT  SOURCE PROGRAM
=====84/07/27==001=====
0001      PN      B2:
0002      ;32 BIT: 16 BIT DIVISION,UNSIGNIERT,ERGEBNIS 16 BIT
0003      ;DIVIDENT 32 BIT IN HL,BC
0004      ;DIVISOR 16 BIT IN DE
0005      ;QUOTIENT 16 BIT IN BC
0006      DIV:    XOR  A
0007      EXAF
0008      LD      A,17      ;A=0
0009      K3:     EXAF      ;ZAEHLER=17
0010      SBC    HL,DE      ;SUBTRAKTION
0011      SBC    0          ;A HL= DE-A HL
0012      JNC    K6-#
0013      ADD    HL,DE      ;ADDITION WENN ERGEBNIS NEGATIV
0014      ADC    0
0015      K6:     CCF
0016      RL     C          ;CY = CY
0017      RL     B          ;VERSCHIEBUNG
0018      RL     L
0019      RL     H
0020      RLA
0021      EXAF
0022      DEC    A          ;A-1-A
0023      JNZ    K3-#
0024      RET
0025      END
0026      PROGRAM CONTAINS 0000 ERROR(S)

```

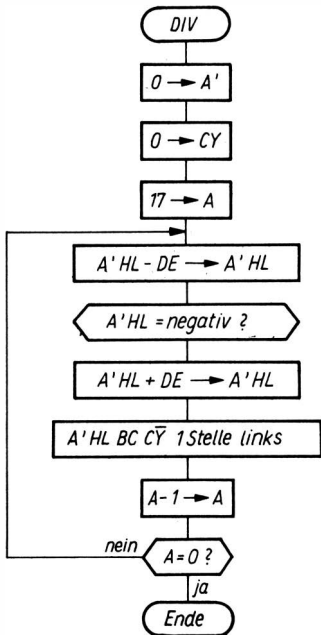


Bild 6.6  
Flußdiagramm zur Division

In Bild 6.6 ist der Ablauf als Flußbild gegenübergestellt. Zur Beschreibung von Programmabläufen wird sowohl das Flußbild als auch die Darstellung als Struktogramm verwendet. Die Flußbild-darstellung ist für den Anfänger leichter verständlich. Man sollte jedoch die Struktogrammdarstellung anstreben, da diese schärfer den Algorithmus wiedergibt und außerdem zu übersichtlicheren Beschreibungen und Programmen führt.

Das zusätzliche Register A' wird nicht benötigt, wenn wir statt 16-Bit-Divisor nur einen 15-Bit-Divisor und statt einem 32-Bit-Di-videnden einen 30-Bit-Dividenden nehmen.

### Beispiel 9

Division einer ganzen Zahl 30 Bit durch eine ganze Zahl 15 Bit. Der Quotient ist 15 Bit.

Eingangsparameter: Dividend in HL BC  
Divisor in DE

## Ausgangsparameter: Quotient in BC

Das Divisionsschema bleibt wie in Bild 6.4.

```

ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
LOC  OBJ.CODE  STMT  SOURCE PROGRAM
=====84/07/27=====001=====
0001      FN      B3
0002
0003      ;DIVISION, GANZE ZAHLEN 30 BIT:15 BIT GIBT 15 BIT
0004      ;DIVIDENT IN HL BC
0005      ;DIVISOR IN DE
0006      ;QUOTIENT IN BC
0000 3E 11 0007 DIV2: LD  A, 17      ;ZAEHLER
0002 87      0008      OR  A      ;CV=0
0003 ED 52 0009 ZVK:  SBC  HL, DE
0005 30 01 0010      JRNK KRUE-#    ;BEI CV=0 KEINE RUECKSTELLUNG
0007 19      0011      ADD  HL, DE  ;RESTRUECKSTELLUNG
0008 3F      0012 KRUE:  CCF
0009 08 11 0013      RL  C      ;VERSCHIEBUNG DIVIDENT
000B 08 10 0014      RL  B
000D 29      0015      ADD  HL, HL
000E 3D      0016      DEC  A
000F 20 F2 0017      JRNZ ZVK-#
0011 09      0018      RET
0012      0019      END
PROGRAM CONTAINS 0000 ERROR(S)

```

Da der Divisor gegenüber dem Dividenten um 16 statt um 14 Stellen nach links verschoben ist, bewirkt, daß der Zähler am Anfang auf 17 und nicht auf 15 gestellt werden muß.

Im anderen Fall müßte vorher der Divident um 2 Stellen nach links verschoben werden, da sich der Divisor in DE nicht weiter nach rechtschieben läßt.

$$Q = Q_{14} \cdot 2^{14} + \dots Q_0. \quad (10)$$

Bis jetzt haben wir die Multiplikation und Division ohne Vorzeichen betrachtet. Bei einer Multiplikation und Division mit Vorzeichen wird meistens Vorzeichen und Betrag getrennt berechnet. Liegt eine negative Zahl als Zweierkomplement vor, muß zunächst der Betrag gebildet werden. Die Betragsbildung kann durch die Operation  $0 - \text{Zahl}$ ,

wobei die Zahl im Zweierkomplement vorliegt, erfolgen. Die Vorzeichenberechnung ist für Multiplikation und Division gleich, und zwar gilt:

| 1. Operand | 2. Operand | Ergebnis |
|------------|------------|----------|
| +          | +          | +        |
| +          | -          | -        |
| -          | +          | -        |
| -          | -          | +        |

Da bei negativen Vorzeichen die Vorzeichenstelle «1» und bei positiven Vorzeichen «0» ist, gilt:

Ergebnisvorzeichen = Vorzeichen 1. Operand  $\oplus$  Vorzeichen 2. Operand, wobei  $\oplus$  das Exklusiv-Oder (XOR-Befehl) ist.

### Beispiel 10

Multiplikation zweier Zahlen mit Vorzeichen

(Zweierkomplement) 15 Bit plus Vorzeichen

Ergebnis 30 Bit plus Vorzeichen

Eingangsparameter: Multiplikator in BC

Multiplikand in DE

Ausgangsparameter: Produkt in HL BC

```
ABS. MACROASSEMBLER K1520 /1 ME05 1521 V4.2
LOC   OBJ.CODE   SMT      SOURCE PROGRAM
=====04/07/31=====001=====
0001      PH      B4
0002      ;MULTIPLIKATION MIT VORZEICHEN (ZWEIERKOMPLEMENT)
0003      ;DE X BC NACH HLEB
0004
0005      MUL2U: LD      A,D          ;BERECHNUNG ERGEBNISVORZEICHEN
0006      XOR      B
0007      AND      80H
0008      PUSH     AC
0009      BIT      7,D          ;BETRAG MULTIPLIKANT
0010      JRZ      M3-#
0011      LD      HL,0
0012      SBC      HL,DE
0013      EX      DE,HL
0014      BIT      7,B          ;BETRAG MULTIPLIKATOR
0015      JRZ      M4-#
0016      LD      HL,0
0017      XOR      A
0018      SBC      HL,BC
0019      LD      B,H
0020      LD      C,L
0021      CALL    MUL2          ;PRODUKT DER BETRÄGE
0022      POP      AF          ;KOMPLEMENT PRODUKT
0023      BIT      7,A
0024      JRZ      M5-#
0025      SUB      C
0026      LD      C,A
0027      LD      A,0
0028      SBC      B
0029      LD      B,A
0030      LD      A,0
0031      SBC      L
0032      LD      L,A
0033      LD      A,0
0034      SBC      H
0035      LD      H,A
0036      M5:      RET
0037      ;MULTIPLIKATIONSROUTINE
0038      MUL2: LD      HL,0
0039      LD      A,17
0040      K3:      RR      B
0041      RR      C
0042      DEC      A
0043      RZ
0044      JRNC     K7-#
0045      ADD      HL,DE
0046      K7:      RR      H
0047      RR      L
0048      JR      K3-#
0049      END
PROGRAM CONTAINS 0000 ERROR(S)
```

Für die Multiplikation der Beträge wurde die Routine von Beispiel 7 genommen.

### Beispiel 11

Division zweier Zahlen mit Vorzeichen (Zweierkomplement)

Eingangsparameter: Dividend in HL BC

(30 Bit + Vorzeichen)

Divisor in DE

(15 Bit + Vorzeichen)

## Ausgangsparameter: Quotient in BC (15 Bit + Vorzeichen)

| LOC                            | OBJ. CODE | ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2<br>STMT SOURCE PROGRAM |
|--------------------------------|-----------|--------------------------------------------------------------------|
| =====84/07/31=====001=====     |           |                                                                    |
|                                |           | 0001 FN 05                                                         |
|                                |           | 0002 ;DIVISIONSRoutine MIT VORZEICHEN (ZWEIERKOMPLEMENT)           |
|                                |           | 0003 ;32 BIT:16 BIT ERGIBT 16 BIT                                  |
|                                |           | 0004 ;HLBC:DE NACH BC                                              |
|                                |           | 0005                                                               |
| 0000                           | 7C        | 0006 DIV2: LD A,H ;BERECHNUNG ERGEBNISVORZEICHEN                   |
| 0001                           | AA        | 0007 XOR D                                                         |
| 0002                           | E6 00     | 0008 AND 80H                                                       |
| 0004                           | F5        | 0009 PUSH AF                                                       |
| 0005                           | C8 7A     | 0010 BIT 7,D ;BETRAG DIVISOR                                       |
| 0007                           | 28 08     | 0011 JRZ M8-#                                                      |
| 0009                           | E5        | 0012 PUSH HL                                                       |
| 000A                           | 21 00 00  | 0013 LD HL,0                                                       |
| 000D                           | ED 52     | 0014 SBC HL,DE                                                     |
| 000F                           | EB        | 0015 EX DE,HL                                                      |
| 0010                           | E1        | 0016 POP HL                                                        |
| 0011                           | C8 7C     | 0017 M8: BIT 7,H ;BETRAG DIVIDEND                                  |
| 0013                           | 28 18     | 0018 JRZ M9-#                                                      |
| 0015                           | 3E 00     | 0019 LD A,0                                                        |
| 0017                           | 91        | 0020 SUB C                                                         |
| 0018                           | 4F        | 0021 LD C,A                                                        |
| 0019                           | 3E 00     | 0022 LD A,0                                                        |
| 001B                           | 98        | 0023 SBC B                                                         |
| 001C                           | 47        | 0024 LD B,A                                                        |
| 001D                           | 3E 00     | 0025 LD A,0                                                        |
| 001F                           | 9D        | 0026 SBC L                                                         |
| 0020                           | 6F        | 0027 LD L,A                                                        |
| 0021                           | 3E 00     | 0028 LD A,0                                                        |
| 0023                           | 9C        | 0029 SBC H                                                         |
| 0024                           | 67        | 0030 LD H,A                                                        |
| 0025                           | CD 36 00  | 0031 M9: CALL DIV2 ;DIVISION DER BETRÄGE                           |
| 0028                           | F1        | 0032 POP AF ;BETRAG QUOTIENT                                       |
| 0029                           | C8 7F     | 0033 BIT 7,A                                                       |
| 002B                           | 28 1A     | 0034 JRZ M10-#                                                     |
| 002D                           | 3E 00     | 0035 LD A,0                                                        |
| 002F                           | 91        | 0036 SUB C                                                         |
| 0030                           | 4F        | 0037 LD C,A                                                        |
| 0031                           | 3E 00     | 0038 LD A,0                                                        |
| 0033                           | 98        | 0039 SBC B                                                         |
| 0034                           | 47        | 0040 LD B,A                                                        |
| 0035                           | C9        | 0041 ;DIVISIONSRoutine                                             |
| 0036                           | 3E 11     | 0042 DIV2: LD A,17                                                 |
| 0038                           | B7        | 0043 OR A                                                          |
| 0039                           | ED 52     | 0044 ZVK: SBC HL,DE                                                |
| 003B                           | 38 01     | 0045 JRNK KRUE-#                                                   |
| 003D                           | 19        | 0046 ADD HL,DE                                                     |
| 003E                           | 3F        | 0047 KRUE: CCF                                                     |
| 003F                           | CB 11     | 0048 RL C                                                          |
| 0041                           | CB 18     | 0049 RL B                                                          |
| 0043                           | 29        | 0050 ADD HL,HL                                                     |
| 0044                           | 3D        | 0051 DEC A                                                         |
| 0045                           | 28 F2     | 0052 JRNZ ZVK-#                                                    |
| 0047                           | C9        | 0053 M10: RET                                                      |
| 0048                           |           | 0054 END                                                           |
| PROGRAM CONTAINS 0000 ERROR(S) |           |                                                                    |

Für die Division der Beträge wurde die Routine aus Beispiel 9 verwendet.

Für die Arbeit mit BCD-Zahlen soll ein Programm zur Realisierung des kleinen Einmaleins dienen.

### Beispiel 12

Multiplikation von 2 ganzen Dezimalzahlen (BCD-Code) mit 2 Stellen. Das Produkt darf maximal ebenfalls 2 Stellen betragen. Da der Rechner sehr schnell arbeitet (pro Befehl ca.  $5\mu s$ ), führen wir die Multiplikation mit kleinen Zahlen durch einfache Addition durch.

Eingangsparameter: Multiplikand 2 BCD-Stellen in C  
 Multiplikator 2 BCD-Stellen in B  
 Ausgangsparameter: Produkt 2 BCD-Stellen in D  
 Bild 6.7 zeigt das Flußbild

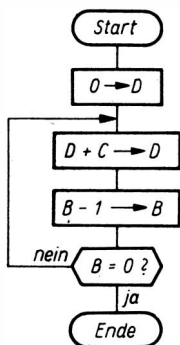


Bild 6.7  
 Flußbild zur Multiplikation  
 von 2 BCD-Zahlen mit 2 Stellen  
 (C · B → D)

| LOC  | OBJ.CODE | HEX. INSTRUCTIONS                                  | SOURCE PROGRAM          |
|------|----------|----------------------------------------------------|-------------------------|
|      |          | 0001 FN B6                                         |                         |
|      |          | 0002 MULTIPLIKATION VON 2 BCD-ZAHLEN MIT 2 STELLEN |                         |
|      |          | 0003 MULTIPLIKAND IN REGISTER C                    |                         |
|      |          | 0004 MULTIPLIKATOR IN REGISTER B                   |                         |
|      |          | 0005 PRODUKT IN REGISTER D (INITIALE 2 STELLEN)    |                         |
| 0000 | 16 00    | 0007 LD B=0                                        | PRODUKT=0 STELLEN       |
| 0002 | 7H       | 0008 M1: LD A=D                                    | ADDITION MULTIPLIKAND   |
| 0003 | 61       | 0009 ADD C                                         |                         |
| 0004 | 27       | 0010 DRR                                           |                         |
| 0005 | 57       | 0011 LD D=A                                        |                         |
| 0006 | 76       | 0012 LD A=B                                        | VEREHLUNG MULTIPLIKATOR |
| 0007 | 06 01    | 0013 SUB 1                                         |                         |
| 0009 | 27       | 0014 DRR                                           |                         |
| 000A | 47       | 0015 LD B=A                                        |                         |
| 000B | 20 F5    | 0016 JNZ M1+6                                      |                         |
| 000D | C9       | 0017 RET                                           |                         |
| 000E |          | 0018 END                                           |                         |

PROGRAM CONTAINS: 0000: ERROR 0.

Für die Arbeit mit Gleitkommazahlen betrachten wir die Multiplikation. Da eine Gleitkommazahl aus Mantisse und Exponent besteht, gilt:

$$z1 = m_1 \cdot 2^{E_1} \quad (11)$$

$$z2 = m_2 \cdot 2^{E_2} \quad (12)$$

und für das Produkt

$$z = z1 \cdot z2 = m1 \cdot 2^{E_1} \cdot m_2 \cdot 2^{E_2} = m_1 \cdot m_2 \cdot 2^{E_1+E_2}, \quad (13)$$

d. h., wir müssen die Mantissen multiplizieren und die Exponenten addieren. Die Mantisse ist normalisiert. Eine Zahl Normalisieren

heißt, sie so umformen, daß die erste Stelle nach dem Komma 1 wird.

### Beispiel 13

| LOC                           |          | OBJ. CODE | STMT | SOURCE PROGRAM                                               |
|-------------------------------|----------|-----------|------|--------------------------------------------------------------|
| =====84/07/01=====001=====    |          |           |      |                                                              |
|                               |          |           | 0001 | PN B7                                                        |
|                               |          |           | 0002 |                                                              |
|                               |          |           | 0003 | ;GLEITKOMMAMULTIPLIKATION:2 BYTE MANTISSE,1BYTE EXPONENT     |
|                               |          |           | 0004 | ;ZAHLENDARSTELLUNG:MANTISSE NORMALISIERT,RECHN. 4 VORZEICHEN |
|                               |          |           | 0005 | ; (15 + 1817)                                                |
|                               |          |           | 0006 | ;EXPONENT: ZWEIERKOMPLEMENT (7 +1 BIT)                       |
|                               |          |           | 0007 | ;EINGANGSPARAMETER:MANTISSE MULTIPLIKANT IN DE               |
|                               |          |           | 0008 | ; EXPONENT MULTIPLIKANT IN H                                 |
|                               |          |           | 0009 | ; MANTISSE MULTIPLIKATOR IN EL                               |
|                               |          |           | 0010 | ; EXPONENT MULTIPLIKATOR IN HL                               |
|                               |          |           | 0011 | ;AUSGANGSPARAMETER:MANTISSE PRODUKT IN HL                    |
|                               |          |           | 0012 | ; EXPONENT PRODUKT IN H                                      |
|                               |          |           | 0013 | ;BEI UEBERLAUF IST DAS P/V-FLAG GEGSETZT, SONST RUECKGESETZT |
|                               |          |           | 0014 | ;BERECHNUNG VORZEICHEN                                       |
| 0000                          | 74       |           | 0015 | GMU2: LD A:D                                                 |
| 0001                          | H8       |           | 0016 | XOR B                                                        |
| 0002                          | E6 80    |           | 0017 | AND 80H ;VORZEICHEN ERGEBNIS:MANTISSE AUF BIT                |
| 0004                          | CB 68    |           | 0018 | RES 7:0 ;VORZEICHEN MULTIPLIKAND LUESCHEN                    |
| 0006                          | CB 84    |           | 0019 | RES 7:D ;VORZEICHEN MULTIPLIKANT LUESCHEN                    |
| 0008                          | E5       |           | 0020 | PUSH HL                                                      |
| 0009                          | F5       |           | 0021 | PUSH AF                                                      |
|                               |          |           | 0022 | ;MULTIPLIKATION MANTISSEN                                    |
| 000A                          | 21 00 00 |           | 0023 | MU2: LD HL:0                                                 |
| 000B                          | 3E 11    |           | 0024 | LD A:11H                                                     |
| 000F                          | CB 15    |           | 0025 | K3: RR B                                                     |
| 0011                          | CB 19    |           | 0026 | RR C                                                         |
| 0013                          | 2D       |           | 0027 | DEC A                                                        |
| 0014                          | 29 09    |           | 0028 | JRZ NRH-#                                                    |
| 0016                          | 30 01    |           | 0029 | JRNC K7-#                                                    |
| 0018                          | 19       |           | 0030 | ADD HL,DE                                                    |
| 0019                          | CB 1C    |           | 0031 | RR H                                                         |
| 001B                          | CB 1D    |           | 0032 | RR L                                                         |
| 001D                          | 18 F0    |           | 0033 | JR K3-#                                                      |
|                               |          |           | 0034 | ;NORMALISIERUNG UND STELLENPUNKT FÜR VERSCHIEBUNG            |
| 001F                          | 6E 01    |           | 0035 | NRH: LD C:1                                                  |
| 0021                          | CB 18    |           | 0036 | SHR HL B                                                     |
| 0023                          | CB 15    |           | 0037 | RR L                                                         |
| 0025                          | 2E 14    |           | 0038 | RR H                                                         |
| 0027                          | 00       |           | 0039 | DEC C                                                        |
| 0028                          | CB 74    |           | 0040 | BIT 6:H                                                      |
| 002A                          | 28 F5    |           | 0041 | JRZ SH-#                                                     |
|                               |          |           | 0042 | ;RUNDUNG                                                     |
| 002C                          | CB 18    |           | 0043 | RR B                                                         |
| 002E                          | 11 00 00 |           | 0044 | LD DE:0                                                      |
| 0031                          | ED 54    |           | 0045 | ADD HL,DE                                                    |
| 0033                          | CB 7C    |           | 0046 | BIT 7:H                                                      |
| 0035                          | 28 03    |           | 0047 | JRZ M2-#                                                     |
|                               |          |           | 0048 | ;NORMALISIERUNG NACH DER RUNDUNG                             |
| 0037                          | CB 1C    |           | 0049 | RR H                                                         |
| 0039                          | 00       |           | 0050 | INC C                                                        |
|                               |          |           | 0051 | ;VORZEICHEN EINSETZEN                                        |
| 003A                          | F1       |           | 0052 | M2: PUSH AF                                                  |
| 003B                          | B4       |           | 0053 | OR H                                                         |
| 003C                          | 67       |           | 0054 | LD H:H                                                       |
|                               |          |           | 0055 | ;BERECHNUNG DES EXPONENTEN                                   |
| 003D                          | 701      |           | 0056 | EXP: PUSH DE                                                 |
| 003E                          | 76       |           | 0057 | LD H:E                                                       |
| 003F                          | 82       |           | 0058 | ADD D                                                        |
| 0040                          | EA 44 00 |           | 0059 | JPFE END ;UEBERLAUF                                          |
| 0043                          | 81       |           | 0060 | ADD C                                                        |
| 0044                          | C9       |           | 0061 | END RET                                                      |
| 0045                          |          |           | 0062 | END                                                          |
| PROGRAM CONTAINS ZERO ERRORS. |          |           |      |                                                              |

### Beispiel 14

#### Digitaluhr mit Prozessor und einer LED-Anzeige

Zur Realisierung einer Digitaluhr werde die in Bild 6.6 beschriebene LED-Anzeige verwendet. Die Zuordnung der LED-Elemente zur Uhrzeit zeigt die folgende Darstellung:

LED-Element       $\underbrace{\quad 6 \quad 5 \quad}$        $\underbrace{\quad 4 \quad 3 \quad}$        $\underbrace{\quad 2 \quad 1 \quad}$   
                          Stunden      Minuten      Sekunden

Das Programm besteht aus folgenden Teilschritten:

1. Schritt – Realisierung des eigentlichen Uhrenprogramms;
2. Schritt – Speichern der Zahlenwerte für Sekunden, Minuten und Stunden in den Datenpuffer für die Anzeige;
3. Schritt – Aufruf des Programms für die LED-Anzeige.

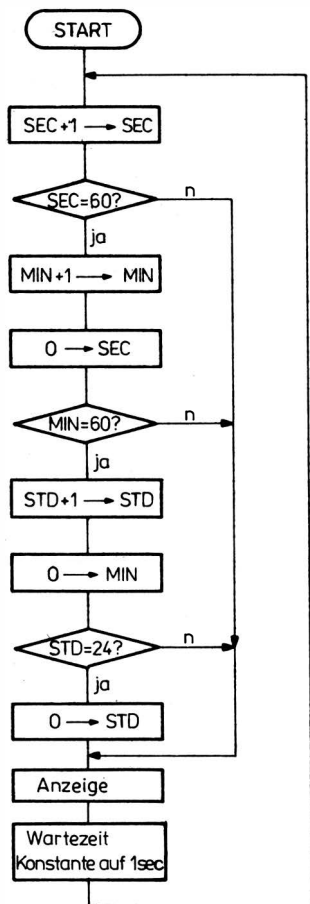


Bild 6.8  
 Flußdiagramm zur Errechnung  
 von Sekunden, Minuten und Stunden  
 eines Uhrenprogramms



| LOC                            | OBJ.CODE    | STMT | SOURCE                     | PROGRAM     | C103<br>04/08/15==002                 |
|--------------------------------|-------------|------|----------------------------|-------------|---------------------------------------|
| 0070                           | 50 01       | 0067 | ZK:                        | DA 150H     | ; ZAEHLKONSTANTE                      |
| 0072                           | 0C          | 0068 | TOR:                       | DB 0CH      | ; TABELLE DER PERIPHEREN LED-ADRESSEN |
| 0073                           | 0D          | 0069 |                            | DB 0DH      |                                       |
| 0074                           | 0E          | 0070 |                            | DB 0EH      |                                       |
| 0075                           | 0F          | 0071 |                            | DB 0FH      |                                       |
| 0076                           | 1C          | 0072 |                            | DB 1CH      |                                       |
| 0077                           | 1D          | 0073 |                            | DB 1DH      |                                       |
| 0078                           | 5F          | 0074 | STAB:                      | DB 5FH      | ; SEGMENTTABELLE                      |
| 0079                           | 06          | 0075 |                            | DB 6H       |                                       |
| 007A                           | 38          | 0076 |                            | DB 38H      |                                       |
| 007B                           | 2E          | 0077 |                            | DB 2EH      |                                       |
| 007C                           | 66          | 0078 |                            | DB 66H      |                                       |
| 007D                           | 6D          | 0079 |                            | DB 6DH      |                                       |
| 007E                           | 7D          | 0080 |                            | DB 7DH      |                                       |
| 007F                           | 07          | 0081 |                            | DB 7H       |                                       |
| 0080                           | 7F          | 0082 |                            | DB 7FH      |                                       |
| 0081                           | 6F          | 0083 |                            | DB 6FH      |                                       |
| 0082                           | 77          | 0084 |                            | DB 77H      |                                       |
| 0083                           | 7C          | 0085 |                            | DB 7CH      |                                       |
| 0084                           | 59          | 0086 |                            | DB 59H      |                                       |
| 0085                           | 3E          | 0087 |                            | DB 3EH      |                                       |
| 0086                           | 79          | 0088 |                            | DB 79H      |                                       |
| 0087                           | 71          | 0089 |                            | DB 71H      |                                       |
| 0088                           | FD 21 72 00 | 0090 | ; AUSGABE AUF LED-ELEMENTE |             |                                       |
| 008C                           | DD 21 6A 00 | 0091 | LEDA:                      | LD IX, TOR  |                                       |
| 0090                           | 06 06       | 0092 |                            | LD IX, LP   |                                       |
| 0092                           | 21 78 00    | 0093 |                            | LD B, C     |                                       |
| 0095                           | FD 4E 00    | 0094 |                            | LD HL, STAB |                                       |
| 0098                           | DD 5E 00    | 0095 | ZVK:                       | LD C, (IX)  | ; ELEMENTADRESSE NACH C               |
| 009B                           | 16 00       | 0096 |                            | LD E, (IX)  | ; ZIFFER NACH E                       |
| 009D                           | 19          | 0097 |                            | LD D, 0     |                                       |
| 009E                           | 7E          | 0098 |                            | ADD HL, DE  |                                       |
| 009F                           | ED 79       | 0099 |                            | LD A, (HL)  | ; SEGMENTCODE NACH A                  |
| 00A1                           | DD 23       | 0100 |                            | OUT A       |                                       |
| 00A3                           | FD 23       | 0101 |                            | INC IX      |                                       |
| 00A5                           | 18 EE       | 0102 |                            | INC IX      |                                       |
| 00A7                           | C9          | 0103 |                            | DJNZ ZVK-#  |                                       |
| 00AB                           |             | 0104 |                            | RET         |                                       |
|                                |             | 0105 |                            | END         |                                       |
| PROGRAM CONTAINS 0000 ERROR(S) |             |      |                            |             |                                       |

## Quadratwurzel

Das folgende Programm soll zeigen, wie höhere Operationen auf einem Mikrorechner programmiert werden können. Wichtig dabei ist, daß man zunächst ein Lösungsverfahren findet, das möglichst an das Dualsystem angepaßt ist. Es gilt:

$$Z = X$$

$$Z = 0, \quad Z_1 Z_2 Z_3 \dots Z_N \quad X = 0, \quad X_1 X_2 X_3 \dots X_N$$

Verfahren:

$$P_0 = Z \quad X_0 = 0$$

$$P_K = 2 \left[ P_{K-1} - \operatorname{sgn}(P_{K-1}) \left( 2^{-K} + \sum_{j=1}^{K-1} X_j \cdot 2^{-j} \right) + 2^{-K-1} \right]. \quad (14)$$

$$X_K = \frac{1 + \operatorname{sgn}(P_K)}{2} \operatorname{sgn}(P) = \begin{cases} +1, & \text{falls } P \text{ positiv} \\ -1, & \text{falls } P \text{ negativ} \end{cases}$$

Das Verfahren ist ein Iterationsverfahren. Die Zahl Z, aus der die Wurzel gezogen wird, muß kleiner als 1 sein. Sie liegt als reine Dualzahl mit n Dualstellen  $Z_1, Z_2$  bis  $Z_n$  vor. Die Wurzel aus Z hat wieder n Dualstellen  $X_1, X_2$  bis  $X_N$ . Im Verfahren geht man von der Zahl  $P_0 = Z$  aus. Die Ziffer  $X_0$  (Stelle vor dem Komma) ist 0. Mit

$P_0$  und  $X_0$  wird nach den angegebenen Formeln  $P_1$  und  $X_1$  berechnet.  $X_1$  ist die 1. Stelle der Lösung hinter dem Komma. Mit  $P_1$  und  $X_1$  berechnet man  $P_2$  und  $X_2$  usw. Mit jedem Schritt gewinnt man eine Dualstelle der Wurzel.

Um das Verfahren «rechnergerecht» aufzubereiten, wird die Formel etwas umgestellt. Es sei angenommen, daß  $P_0$   $P_1$  bis  $P_{K-1}$  und  $X_0$   $X_1$  bis  $X_{K-1}$  bereits ermittelt worden sind und nun  $P_K$  und  $X_K$  ermittelt werden sollen.

Wenn man  $P_{K-1}$  als Zahl schreibt, gilt:

$$P_{K-1} = 0, \quad S_1 S_2 \dots S_{K-1} S_K S_{K+1} \dots S_n, \quad (15)$$

wobei  $S_1 S_2 \dots S_n$  die Dualstellen von  $P_{K-1}$  sind.

Weiterhin gilt:

$$\sum_{j=1}^{K-1} X_j \cdot 2^{-j} = 0, \quad X_1 X_2 \dots X_{K-1}. \quad (16)$$

Ist  $P_{K-1}$  positiv, so ist  $\text{sgn}(P_{K-1}) = 1$  und

$$\left. \begin{aligned} P_K &= 2 \left[ P_{K-1} - \sum_{j=1}^{K-1} X_j 2^{-j} - 2^{-K-1} \right], \\ P_K &= 2 \left[ P_{K-1} - \sum_{j=1}^{K-1} X_j 2^{-j} \right] - 2^{-K}. \end{aligned} \right\} \quad (17)$$

Ist  $P_{K-1}$  negativ, so ist  $\text{sgn}(P_{K-1}) = -1$  und

$$P_K = 2 \left[ P_{K-1} + \sum_{j=1}^{K-1} X_j 2^{-j} + 2^{-K} + 2^{-K-1} \right]. \quad (18)$$

Wird  $P_K$  nach Gl. (17) negativ, so wird  $P_{K+1}$  nicht nach Gl. (18) berechnet, sondern es gilt mit  $P_K$  negativ  $\text{sgn}(P_K) = -1$ ;  $X_K = 0$ :

$$P_{K+1} = 2 \left[ 2P_K + \sum_{j=1}^K X_j 2^{-j} + 2^{-K-1} + 2^{-K-2} \right], \text{ nach } (18)$$

$$P_K = 2 \left[ P_{K-1} - \sum_{j=1}^{K-1} X_j 2^{-j} - 2^{-K-1} \right]. \text{ Nach } (17)$$

$$P_{K+1} = 2 \left[ 2P_{K-1} - 2 \sum_{j=1}^{K-1} X_j 2^{-j} - 2^{-K} + \underbrace{\sum_{j=1}^{K-1} X_j 2^{-j}}_{\text{da } X_K = 0} + 2^{-K-1} + 2^{-K-2} \right],$$

$$P_{K+1} = 2 \left[ 2P_{K-1} - \sum_{j=1}^{K-1} X_j 2^{-j} - 2^{-K-2} \right],$$

$$P_{K+1} = 2 \left[ 2P_{K-1} - \sum_{j=1}^{K-1} X_j 2^{-j} \right] - 2^{-K-1}. \quad (19)$$

Mit Gl. (17) und Gl. (19) erhält man folgendes Verfahren zum Radizieren von Quadratwurzeln:

| 1. Voraussetzungen                                             | Dualwert                |
|----------------------------------------------------------------|-------------------------|
| a) Bereitstellen von $P_0 = Z$                                 | 0 $Z_1$ $Z_2 \dots Z_n$ |
| b) $X = 0, X_1 X_2 \dots X_n = 0$ setzen                       | 0   0   0 $\dots 0$     |
| c) Bereitstellen einer Konstanten $K$ zum Bit der Ziffer $X_K$ | 0   1   0   0 $\dots 0$ |

## 2. Rechenablauf

Ausgangspunkt  $P_{K-1}$ ; am Anfang  $P_0$

a) Bildung von  $2P_{K-1}$  für Gl. (19)

b) Bildung von

$$P_K = 2 \left[ P_{K-1} - \sum_{j=1}^{K-1} X_j 2^{-j} \right] - 2^{-K}$$

c) Ist  $P_K$  positiv, so wird  $X_K = 1$  gesetzt (Bildung von  $X + K \rightarrow X$ ), und statt  $P_{K-1}$  wird  $P_K$  gesetzt.

d) Ist  $P_K$  negativ, so wird  $X_K = 0$  gesetzt, und statt  $P_K$  wird für Gl. (19)  $2P_{K-1}$  gesetzt.

e)  $K$  wird um eine Stelle nach rechts verschoben.

f) Fortführung bei a) so lange, bis die Anzahl der Stellen genügt. Das folgende Beispiel zeigt das Programm für die Quadratwurzel aus einer 15stelligen Dualzahl  $|z| < 1$ .

## Beispiel 15

Quadratwurzel aus einer echtgebrochenen Zahl 15 Bit (Bild 6.9)

Eingangsparameter: Radikand in HL, Komma steht nach Bit 15

Ausgangsparameter: Wurzel in DE

Komma steht nach Bit 15



|                                      |      |        |     |          |               |
|--------------------------------------|------|--------|-----|----------|---------------|
| 0029                                 | 0025 | 81     | ADD | C        |               |
| 0030                                 | 0026 | CO     | RNZ |          | ; Abfrage K15 |
| 0031                                 | 0027 | 0E01   | LD  | C,1      |               |
| 0032                                 | 0029 | 18DB   | JR  | K4-#     |               |
| 0033                                 | 002B | 2A4200 | LD  | HL, (ZW) |               |
| 0034                                 | 002E | CB25   | SLA | L        |               |
| 0035                                 | 0030 | CB14   | RL  | H        |               |
| 0036                                 | 0032 | 18EA   | JR  | K12-#    |               |
| 0037                                 | 0034 | 7D     | LD  | A,L      |               |
| 0038                                 | 0035 | 90     | SUB | B        |               |
| 0039                                 | 0036 | 6F     | LD  | L,A      |               |
| 0040                                 | 0037 | 7C     | LD  | A,H      |               |
| 0041                                 | 0038 | DE00   | SBC | 0        |               |
| 0042                                 | 003A | 67     | LD  | H,A      |               |
| 0043                                 | 003B | 38EE   | JRC | K17-#    | ; Abfrage K20 |
| 0044                                 | 003D | 7B     | LD  | A,E      |               |
| 0045                                 | 003E | B0     | OR  | B        |               |
| 0046                                 | 003F | 5F     | LD  | E,A      |               |
| 0047                                 | 0040 | 18DC   | JR  | K12-#    |               |
| 0048                                 | 0042 | 0000   | DA  | 0        |               |
| 0049                                 |      |        | END |          |               |
| Keine Syntax-Fehler CRAS 4200-K 1520 |      |        |     |          |               |

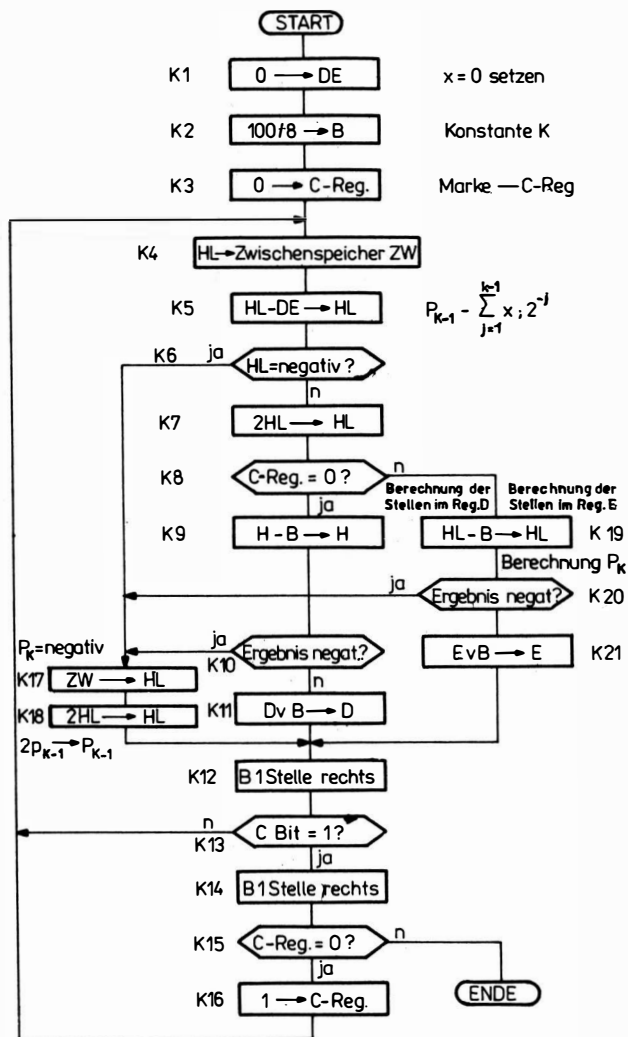


Bild 6.9 Flußdiagramm für das Quadratwurzelnziehen aus einer 15stelligen Dualzahl mit  $|z| < 1$

Beispiel 16

Quadratwurzel aus einer ganzen 4stelligen Dezimalzahl. Geht man von der arithmetischen Reihe

$$1 + 3 + 5 + \dots + (2n - 1) = n^2, \quad a = 2n - 1$$

aus, so kann man die Wurzel aus einer Zahl  $z$  mit folgendem Algorithmus ermitteln:

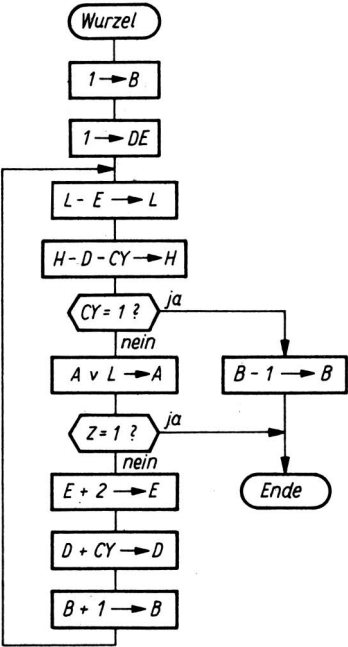
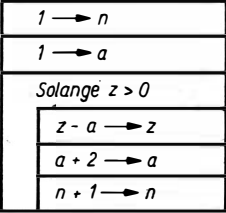


Bild 6.10  
Flußbild: Wurzel aus  
einer ganzen Dezimalzahl

Während man die Glieder der Reihe a vom Radikanden abzieht, zählt man n aufwärts, bis sich kein nächstes Glied a mehr abziehen läßt. n ist damit der ganze Teil der Wurzel.

Bringt man z nach Register HL, a nach Register DE und n nach Register B, so kann man daraus das vorstehende Flußbild bilden.

```

ABS. MACROASSEMBLER K1520 /1 ME05 1521 V4.2
LOC  OBJ.CODE  STMT  SOURCE PROGRAM
=====84/00/03===001=====
0001          FN  B0
0002
0003          ;QUADRATWURZEL AUS EINER GANZEN DEZIMALZAHL MIT 4 BCD-STELLEN
0004          ;RADIKAND IN HL
0005          ;WURZEL IN B
0006
0000 06 01 0007          LD  B,1
0002 11 01 00          LD  DE,1
0005 7D          0009 M1: LD  A,L
0006 93          0010 SUB  E
0007 27          0011 DAA
0008 6F          0012 LD  L,A
0009 9A          0013 SBC  D
000A 27          0014 DAA
000B 67          0015 LD  H,A
000C 30 14       0016 JRC  M3-#
000E E5          0017 OR   L
000F 20 16       0018 JRZ  M2-#
0011 7E          0019 LD  A,E
0012 C6 02       0020 ADD  2
0014 27          0021 DAA
0015 5F          0022 LD  E,A
0016 7A          0023 LD  A,D
0017 CE 00       0024 ADC  0
0019 27          0025 DAA
001A 57          0026 LD  D,A
001B 70          0027 LD  A,B
001C C6 01       0028 ADD  1
001E 27          0029 DAA
001F 47          0030 LD  B,A
0020 18 E3       0031 JR   M1-#
0022 78          0032 M3: LD  A,B
0023 D6 01       0033 SUB  1
0025 27          0034 DAA
0026 47          0035 LD  B,A
0027 C9          0036 M2: RET
0028            0037 END
PROGRAM CONTAINS 0000 ERROR(S)

```

### 6.3. Programme mit peripheren Bausteinen

#### Beispiel 17

Mit einem CTC-Baustein soll eine Impulsfolge nach Bild 6.11 für eine vorgegebene Frequenz erzeugt werden.

Eine Impulsfolge mit einem Tastverhältnis von 1:1 läßt sich mit



Bild 6.11

Impulsfolge mit einem Tastverhältnis von 1:1

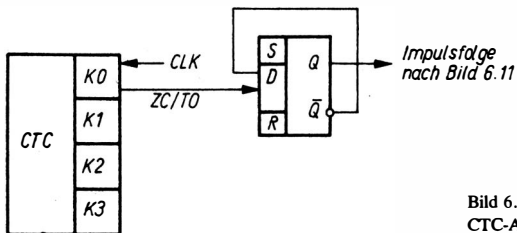


Bild 6.12  
CTC-Ausgang mit D-FF

dem CTC-Baustein nur dadurch realisieren, daß dem Ausgang ZC/TO ein D-Flip-Flop nachgeschaltet wird (Bild 6.12).

Das ist notwendig, weil der Ausgangsimpuls ZC/TO eine konstante Breite von etwa  $0,6\mu s$  hat.

Durch das D-Flip-Flop erfolgt eine Untersetzung 1 : 2.

Bezeichnet man die Zeitkonstante des CTC-Kanals mit Z und die CTC-Teilerkonstante mit K (16 oder 256), so ist die Gesamtuntersetzung 1 :  $(Z \cdot K \cdot 2)$ . Soll z. B. die Frequenz nach Bild 6.11 2,4 kHz betragen und ist die Taktfrequenz 2,4 MHz, muß

$$\frac{2,4 \text{ MHz}}{Z \cdot K \cdot 2} = 2,4 \text{ kHz} \quad \text{sein,} \quad (20)$$

$$Z \cdot K \cdot 2 = 1000$$

$$\text{mit } K = 16$$

$$Z = \frac{1000}{16 \cdot 2} = 31,25 \approx 31 \quad (21)$$

Das Einstellprogramm für den CTC-Schaltkreis lautet:

LD A, 7 ; Zeitgeber, Teiler 16, Reset

OUT CTC

LD A, 31 ; Zeitkonstante 31

OUT CTC

Ein allgemeines Unterprogramm, in dem die Zeitkonstante im Register B und die Adresse des CTC-Kanals im Register C stehen, heißt:

```
CTC:  LDA, 7      ; Zeitgeber, Teiler 16, Reset
      OUT A      ; Kanalsteuerwort
      OUT B      ; Zeitkonstante
      RET
```

Dieses Unterprogramm wird wie folgt aufgerufen:

LDB, 31

LDC, ADR ; CTC-Kanaladresse

CALL CTC

### Beispiel 18

#### Digitaluhr mit CTC-Baustein

Der Vorteil dieser Konfiguration ist es, daß das Uhrenprogramm den Rechner nur dann beansprucht, wenn eine Weiterzählung der Zeit (Sekunden) vorgesehen ist. Der Rechner kann damit gleichzeitig für weitere Aufgaben genutzt werden.

Bild 6.13 zeigt die Zusammenstellung der Baugruppen zum Aufbau einer Digitaluhr mit CTC-Baustein.

Das Programm besteht aus folgenden Teilabschnitten:

1. Initialisieren des CTC-Bausteins,
2. Uhrenprogramm,
3. Abspeichern der Zahlenwerte für Sekunden, Minuten und Stunden in den Datenpuffer für die Anzeige (LED-Puffer),
4. Aufruf des Programms für die LED-Anzeige.

Der CTC-Baustein zählt die Takte  $\Phi$  für die Zeiteinheit «1 s». Dazu setzt man Kanal 0 des Bausteins in die Betriebsart «Zeitgeber». Wenn man den Vorteiler nutzt, so können mit einem Kanal  $256 \times 256 = 65536$  Takte gezählt werden. Beträgt die Quarzfrequenz z. B.  $f = 2,4 \text{ MHz}$ , so besteht 1 s aus  $2,4 \times 10^6$  Takten. Ein Kanal reicht daher für die Zählung einer Sekunde nicht aus. Man zählt deshalb im Kanal 0 die Hundertstelsekunden und über Ka-

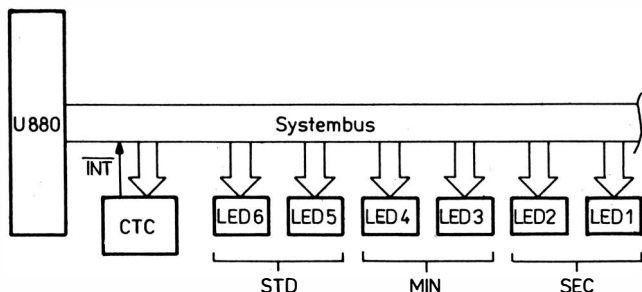


Bild 6.13 Zusammenschaltung der Bausteine U 880 und U 857 D zur Realisierung einer Digitaluhr

nal 1 die Sekunden. Der ZC/TO 0-Impuls wird dabei für die Zählung im Kanal 1 benutzt.

Wird der Vorteiler von Kanal 0 auf  $p = 256$  gestellt, so gilt nach der in Abschnitt 2.3.3. angegebenen Formel für die Zeitdifferenz der Nulldurchgänge:

$$t_i = t_c \cdot p \cdot TKO = \frac{p \cdot TKO}{f} \quad (22)$$

Mit  $t_i = 0,01 \text{ s}$ ,  $p = 256$ ,  $f = 2,4576 \text{ MHz} = 2,4576 \cdot 10^6 \text{ s}^{-1}$  gilt:

$$0,01 \text{ s} = \frac{256 \cdot TKO}{2,4576 \cdot 10^6 \cdot \text{s}^{-1}}.$$

Daraus folgt:

$$TKO = \frac{24576}{256} = 96.$$

Im Kanal 0 des CTC-Bausteins wird demzufolge die Zeitkonstante  $TKO = 96$  eingestellt.

Kanal 1 wird in die Betriebsart «Zähler» gesetzt. Die Zeitkonstante beträgt  $TK1 = 100$ . Damit durchläuft Kanal 1 jede Sekunde den Zählerstand «0». Mit diesem Nulldurchgang bildet man eine Interruptanforderung an den Prozessor. Für die Initialisierung des CTC-Bausteins werden für das Beispiel folgende Adressen vereinbart:

1. Die Auswahladresse des CTC-Bausteins sei 80 H.
2. Der Interruptvektor des CTC-Bausteins sei 90 H.
3. Die Taktfrequenz des Rechners sei 2,4576 MHz.

---

; Uhrenprogramm mit CTC-Baustein

PN UHRCTD

; Initialisierung CTC

|       |     |           |                                            |
|-------|-----|-----------|--------------------------------------------|
| UCTC: | LD  | A,0       | ; Höherwertiger Teil Interruptvektor       |
|       | LD  | I,A       |                                            |
|       | LD  | A,90H     | ; Niederwertiger Teil Interruptvektor      |
|       | OUT | 80H       |                                            |
|       | LD  | IX, ABED  | ; Adresse Bedienprogramm CTC               |
|       |     |           | Interrupt                                  |
|       | LD  | (90H), IX | ; für Kanal 1                              |
|       | LD  | A, 37H    | ; Kanalsteuerwort für Kanal 0              |
|       |     |           | ; Betriebsart, Zeitgeber, Interruptverbot, |
|       |     |           | ; Vorteiler 256, Zeitkonstante laden       |

```

OUT 80H
LD A,(TKO) ; Zeitkonstante für Kanal 0
OUT 80H
LD A,OC7H ; Kanalsteuerwort für Kanal 1
; Betriebsart Zähler, Interrupt
; erlaubt, Zeitkonstante laden

OUT 81H
LD A,(TK1) ; Zeitkonstante für Kanal 1
OUT 81H
IM 2 ; Interruptmode 2
EI ; Interruptfreigabe

; Anzeigeprogramm
ANZ: LD HL, LEDB ; Adresse LED-Puffer nach HL
LD A, (UHR) ; Sekunden nach LED-Puffer
CALLABSP
LD A, (UHR+1) ; Minuten nach LED-Puffer
CALLABSP
LD A, (UHR+2) ; Stunden nach LED-Puffer
CALLABSP
CALLED ; SEC, MIN und STD in
; Anzeigeregister

JRANZ - #

; Interruptbedienprogramm (Zeitzahlprogramm)
ABED: PUSH HL
PUSHAF

SEC: LD HL, UHR ; Sekundenzähleradresse nach HL
LD A, (HL)
ADD 1 ; SEC+1 → SEC
DAA
LD (HL), A
CMP60H ; SEC = 60?
JRZMIN - #
JREN - #

MIN: LD (HL), 0 ; 0 nach SEC
INCHL ; Minutenzähleradresse nach HL
LD A, (HL)
ADD 1 ; MIN + 1 → MIN
DAA
LD (HL), A
CMP60H ; MIN = 60?
JRZSTD - #
JREN - #

STD: LD (HL), 0 ; 0 nach MIN
INCHL ; STD-Zähler nach HL
LD A, (HL)
ADD 1
DAA ; STD + 1 → STD
LD (HL), A

```

```

        CMP24H                ; STD = 24?
        JRZ RES - #
        JREN - #
RES:    LD (HL), 0            ; 0 nach Std
EN:     POP AF
        POP HL
        EI                    ; Interruptfreigabe
        RET

; Abspeicherprogramm für 2 BCD-Ziffern
; aus Register A in einem LED-Puffer
; dessen Adresse in HL steht

ABSP:   PUSH AF
        AND OFH
        LD (HL), A           ; Untere 4 Bit nach LED-Puffer
        INCHL
        POP AF
        RRCA
        RRCA
        RRCA
        RRCA
        AND OFH
        LD (HL), A           ; Obere 4 Bit nach LED-Puffer
        INCHL
        RET

; Ausgabeprogramm der Ziffern vom LED-Puffer
; in die LED-Elemente

LED:    LD B, 6               ; Zähler für 6 Ziffern
        LD IX, LEDB           ; Adresse LED-Puffer nach IX
        LD IY, ANR            ; Adresse Anzeigeregistertabelle
                                ; nach IY
AUS:    LD C, (IY)             ; Adresse Anzeigeregister nach C
        LD HL, SEGM           ; Adresse der Segmenttabelle nach HL
        LD E, (IX)            ; Ziffer nach DE
        LDD, 0
        ADD HL, DE            ; In HL steht Adresse des Segment-
                                ; codes der Ziffer
        LD A, (HL)
        OUT A                 ; Ziffer ausgeben
        INC IX
        INC IY
        DJNZ AUS - #          ; nächste Ziffer
        RET

LEDB:   BER 6                 ; 6 Zellen für LED-Puffer
ANR:    DB OCH                ; Tabelle der Adressen der LED-Elemente
        DB ODH
        DB OEH
        DB OFH
        DB 1CH
        DB 1DH

```

|       |     |      |                                            |
|-------|-----|------|--------------------------------------------|
| SEGM: | DB  | 05FH | ; Segmenttabelle                           |
|       | DB  | 6H   |                                            |
|       | DB  | 3BH  |                                            |
|       | DB  | 2EH  |                                            |
|       | DB  | 66H  |                                            |
|       | DB  | 6DH  |                                            |
|       | DB  | 7DH  |                                            |
|       | DB  | 7H   |                                            |
|       | DB  | 7FH  |                                            |
|       | DB  | 6FH  |                                            |
| UHR:  | BER | 3    | ; Speicher für SEC-, MIN-, STD-<br>; Werte |
| TKO:  | DB  | 60H  | ; Zeitkonstante für CTC-Kanal 0            |
| TK1:  | DB  | 64H  | ; Zeitkonstante für CTC-Kanal 1            |
|       | END |      |                                            |

---

### *Programm zur Ausgabe eines Zeichens über PIO-Schaltkreis-Realisierung eines SIF-1000-Anschlußspiegels*

Der Interfacespiegel SIF 1000 zur Datenausgabe besteht aus folgenden Leitungen:

- 8 Leitungen zur Datenausgabe DATA 1 bis DATA 8
- 3 Kommandoleitungen KOMA 1 bis KOMA 3
- 3 Statusleitungen STATA 1 bis STATA 3
- Der RUF A-Leitung
- Der END A-Leitung

Zunächst werden Daten und Kommando ausgegeben. Sind Daten und Kommando auf den Leitungen stabil, wird dies durch RUFA den peripheren Geräten mitgeteilt. Nachdem die Elektronik des peripheren Gerätes das Datenzeichen und das Kommando abgenommen haben, gibt es den STATUS STATA und etwas später

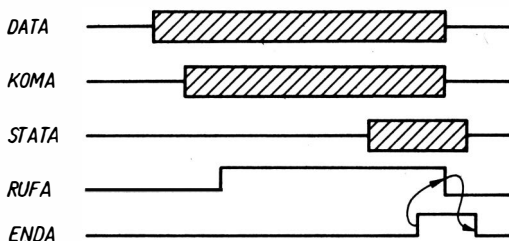


Bild 6.14 Taktdiagramm zur Datenausgabe über SIF 1000 – Spiegel

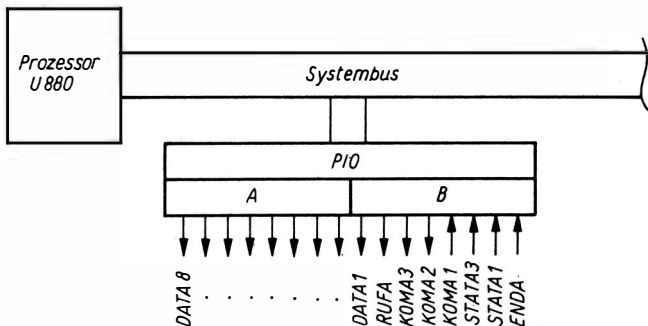


Bild 6.15 Realisierung der SIF-100-Signale mit PIO-Schaltkreis

das Signal ENDA. ENDA sagt dem Rechner, daß der STATUS vorhanden ist und Daten, Kommando und RUFA abgeschaltet werden können. Nachdem der Rechner den STATUS übernommen hat, schaltet er RUFA ab. Danach schaltet die Elektronik des peripheren Gerätes ENDA und STATA ab (Bild 6.14).

Zum Anschluß eines Ausgabegerätes über SIF-1000-Interface-Spiegel benötigen wir 16 Leitungen. Diese 16 Leitungen lassen sich mit einem PIO-Schaltkreis realisieren. Kanal A nehmen wir für die Daten und Kanal B für die restlichen Signale. Bild 6.15 zeigt das Anschlußbild.

Die programmtechnische Realisierung der Ausgabe eines Zeichens läßt sich entweder nach der Betriebsart «Polling» oder «Interrupt» realisieren.

Bei der Betriebsart Polling fragt das Programm nach RUFA ständig das Signal ENDA ab.

Bei der Betriebsart Interrupt erzeugt das Signal ENDA einen Interrupt. Der Rechner kann zwischen Ausgabe eines Zeichens und dem folgenden Interrupt durch ENDA eine 2. Aufgabe (sogenannte Hintergrundaufgabe) bearbeiten.

Der PIO-Schaltkreis muß vor der eigentlichen Ausgabe auf die Betriebsart eingestellt (initialisiert) werden.

### Beispiel 19

Ausgabe eines Zeichens in Pollingmode aus dem Register A mit SIF-1000-Interface über PIO nach Bild 6.15. Das Programm besteht aus 2 Teilen, dem Initialisierungsprogramm und dem Ausga-

beprogramm. Für die Ausgabe wird Kanal A und B in Bitmode initialisiert.

```

ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
LOC   OBJ. CODE  STMT      SOURCE PROGRAM
=====84/00/15==001=====
0001      FN      C2
0002      ;PIO-ADRESSEN
0003      ;DIE ADRESSEN SIND ENTSPRECHEND DER HARDWARE EINZUSETZEN
0004      SPIOA: EQU 0      ;STEUERWORT PIO KANAL A
0005      SPIOB: EQU 0      ;STEUERWORT PIO-KANAL B
0006      ;INITIALISIERUNGSPROGRAMM
0007      INIT: LD HL,STAB      ;ADRESSE STEUERWORTTABELLE
0008      LD C,SPIOA      ;ADRESSE STEUERWORT PIO-KANAL A
0009      LD B,3      ;ANZAHL DER STEUERWÖRTE
0010      OTIR
0011      LD C,SPIOB      ;ADRESSE STEUERWORT PIO-KANAL B
0012      LD B,3      ;ANZAHL DER STEUERWÖRTE
0013      OTIR
0014      RET
0015      ;INITIALISIERUNGSDATEN
0016      ;KANAL A
0017      STAB: DB 0FFH      ;BETRIEBSART BITMODE
0018      DB 0      ;BIT 0-7 AUSGABE
0019      DB 7H      ;INTERRUPTUERBOT
0020      ;KANAL B
0021      DB 0FFH      ;BETRIEBSART BITMODE
0022      DB 0FH      ;BIT 0-3 EINGABE;BIT 4-7 AUSGABE
0023      DB 7H      ;INTERRUPTUERBOT
0024      END
PROGRAM CONTAINS 0000 ERROR(S)

```

## Beispiel 20

```

ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
LOC   OBJ. CODE  STMT      SOURCE PROGRAM
=====84/00/15==001=====
0001      FN      C2
0002      ;PROGRAMM ZUR AUSGABE EINES ZEICHENS AUS DEM REGISTER A
0003      ;-PIO ADRESSE TOR A IN REGISTER D
0004      ;-PIO ADRESSE TOR B IN REGISTER E
0005      ;-SIF 1000 KOMMANDO STEHT IN REGISTER C
0006      ;-SIF 1000 STATUS KOMMT NACH REGISTER B
0007      ;AUSGABE ZEICHEN NACH KANAL A
0008      AUS: OUT D
0009      LD A,C
0010      OUT E      ;AUSGABE KOMMANDO NACH KANAL B
0011      OR 8CH
0012      OUT E      ;AUSGABE KOMMANDO UND RUFA
0013      IN E
0014      WS1: IN E
0015      BIT 0,A      ;TEST AUF ENDA
0016      JRZ WS1-#
0017      AND 0EH      ;AUSBLENDEN STATUS
0018      LD B,A      ;STATUS NACH REGISTER B
0019      LD A,0
0020      OUT E
0021      IN E      ;ABSCHALTEN RUF UND KOMMANDO
0022      BIT 0,A
0023      JRZ WS2-#      ;TEST AUF ABGESCHALTETES ENDA
0024      RET
0025      END
PROGRAM CONTAINS 0000 ERROR(S)

```

## Beispiel 21

```

ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
LOC   OBJ.CODE   STMT   SOURCE PROGRAM
=====04/06/15==001=====
0001          PN      C4
0002 ;AUSGABE EINES DATENSATZES AUS DEM ARBEITSSPEICHER ZU EINEM
0003 ;SIF-1000 GERAET UEBER PIO
0004 ;EINGANGSPARAMETER:ADRESSE AUSGABEPUFFER IN HL
0005 ;          PIO-ADRESSE TOR A IN D
0006 ;          PIO-ADRESSE TOR B IN E
0007 ;          SIF-1000 KOMMANDO STELLENRICHTIG IN C
0008 ;          STATUS WIRD NICHT AUSGEWERTET
0009 AUS: EQU 0 ;ADRESSE FUER ZEICHENAUSGABE
0010 ;DIE ADRESSE IST SPAETER EINZUSETZEN
0011 ASA: LD A,(HL)
0012      PUSH HL
0013      PUSH BC
0014      LD H,C
0015      CALL AUS ;AUSGABE EINES ZEICHENS MIT PROGRAMM
0016 ;AUS BEISPIEL 18
0017      POP BC
0018      POP HL
0019      LD A,(HL)
0020      CMP B
0021      JRZ ENDE-#
0022      INC HL
0023      JR ASA-#
0024 ENDE: RET
0025      END
0000 7E      0011
0001 E5      0012
0002 C5      0013
0003 61      0014
0004 CD 00 00 0015
0007 C1      0017
0008 E1      0018
0009 7E      0019
000A 88      0020
000B 28 03   0021
000D 23      0022
000E 18 F0   0023
0010 C9      0024
0011          0025
PROGRAM CONTAINS 0000 ERROR(S)

```

## **7. Mikrorechnersoftware**

Die Software für Mikrorechner reicht von einfachen Routinen für spezielle Funktionen bis zu kompletten Betriebssystemen für Programmentwicklung und Echtzeitverarbeitung. Je nach Gesichtspunkt läßt sich die Software wie folgt aufteilen:

a – nach Stellung im Rechner

Programmbausteine

Programmpakete

Dienstprogramme

– Datenorganisation

– Testprogramme

– Programmentwicklung

Betriebssysteme

Modellprogramme

b – nach dem Problem

Arithmetik

– Basisarithmetik

– Numerik

Statistik

Operation Research

Ein-/Ausgabe

– Bedienung DV-Geräte

– Bedienung Prozeßperipherie

Systemsoftware

Datenbanksysteme

Die Routinen (meist als einzelne Routinen einer Programmbibliothek) bilden die Basis für die Anwendung als Gerätesteuerechner. Hierbei kommt es darauf an, die Software für die Signalverarbeitung zu entwickeln und auf PROMs bereitzustellen. Für die einzelnen Funktionen werden dabei Routinen eingesetzt, die unter einem kleinen Steuersystem arbeiten. Wortlänge und Zahlendarstellung richten sich nach den Gerätedaten. Um die unterschiedlichen Zahlendarstellungen einander anzupassen, werden Konvertierungsroutinen benötigt. Arithmetische Funktionen kommen in dem Umfang vor, wie sie für die Steuerung des Gerätes benötigt werden; häufig schnelle Routinen mit kurzen Wortlängen, z. B.

Logarithmusfunktion 12 oder 16 Bit für die Umsetzung von linearen in logarithmische Skalen, trigonometrische und dazugehörige Umkehrfunktionen bei der Umwandlung von kartesischen in Polarkoordinaten.

Für die Realisierung gibt es verschiedene Verfahren. Dabei sind solche Verfahren besonders geeignet, die sich unmittelbar auf die Verarbeitung von Bit-Mustern stützen.

Die folgende Zusammenstellung zeigt die gebräuchlichsten Verfahren zur Lösung von arithmetischen Problemen auf Mikrorechnern. Grundrechenarten: (Addition, Subtraktion, Multiplikation, Division)

- Lösung durch Verschiebe-, Addier- und Subtrahieroperationen von Bit-Mustern

Zahlenkonvertierung

- Lösung durch einfache Multiplikation und Division, die auf Addition, Subtraktion und Verschiebung zurückgeführt werden.

Standardfunktion: (Logarithmus, Exponentialfunktion, trigonometrische- und Umkehrfunktion, hyperbolische Funktion)

Lösungsverfahren

- Approximation durch Polynome (Taylor, Tschebyscheff)

- Approximation durch Kettenbrücke

- Cordicalgorithmen

Cordicalgorithmen eignen sich besonders für Festkommaoperationen, da sie unmittelbar mit Bit-Mustern arbeiten.

Programmbausteine lassen sich je nach Anwendungsgebiet in Stufen unterteilen.

Unterteilung Programmbausteine für numerische Probleme:

1. Ebene: Basisprogramme

- Grundfunktion zur Zahlenverarbeitung

- elementare Multiplikation und Division

- Grundoperationen für N-Byte (Verschiebung, Addition, Subtraktion, Normalisierung)

2. Ebene: Arithmetik

- Programmbausteine für Fest- und Gleitkommazahlen im Dual- und BCD-Format

- Grundrechenarten

- Zahlenumwandlung

- Standardfunktionen

3. Ebene:

- allgemeine numerische Lösungsverfahren

## Unterteilung Programmbausteine für Peripheriesteuerung:

1. Ebene: Steuerrountinen
2. Ebene: Ein-/Ausgaberoutinen
3. Ebene: Komplexe Ein-/Ausgabefunktionen

Programmpakete fassen eine Gruppe von Operationen eines Gebietes zusammen. Sie eignen sich bei Einsätzen von Mikrorechnern für Meßwerterfassung, Prozeßsteuerung und Spezialrechnern, wie Bürocomputer, Personalcomputer, Textverarbeitungssystemen usw. Am häufigsten werden Programmpakete für arithmetische Funktionen benötigt. Über sie läuft die Auswertung von Meßdaten sowie die Zusammenstellung von Ergebnissen.

Innerhalb eines Arithmetikpaketes ist die Zahlendarstellung fest. Zahlen, die nicht in der betreffenden Darstellung vorliegen, müssen in die Darstellung des Pakets umgeformt werden. Arithmetikpakete realisieren i. a. folgende Funktionen für eine feste Zahlendarstellung:

### Konvertierung von Zahlen

- BCD – Dual
- Dual – BCD
- Text (ASCII) – DUAL
- DUAL – TEXT (ASCII)
- Festkomma – Gleitkomma
- Gleitkomma – Festkomma

### Grundrechenarten

- Addition
- Subtraktion
- Multiplikation
- Division

### Standardfunktionen

- |                               |                                      |
|-------------------------------|--------------------------------------|
| – Wurzel                      | – Hyperbolischer Sinus ( $\sinh$ )   |
| – Sinus ( $\sin$ )            | – Hyperbolischer Cosinus ( $\cosh$ ) |
| – Cosinus ( $\cos$ )          | – Hyperbolischer Tangens ( $\tanh$ ) |
| – Tangens ( $\tan$ )          | – natürlicher Logarithmus            |
| – Arcus-Sinus ( $\arcsin$ )   | – Exponentialfunktion                |
| – Arcus-Tangens ( $\arctan$ ) | – Potenzfunktion                     |
| – $\text{area sinh}$          | – ganzer Teil einer Zahl             |
| – $\text{areacosh}$           |                                      |
| – $\text{area tanh}$          |                                      |

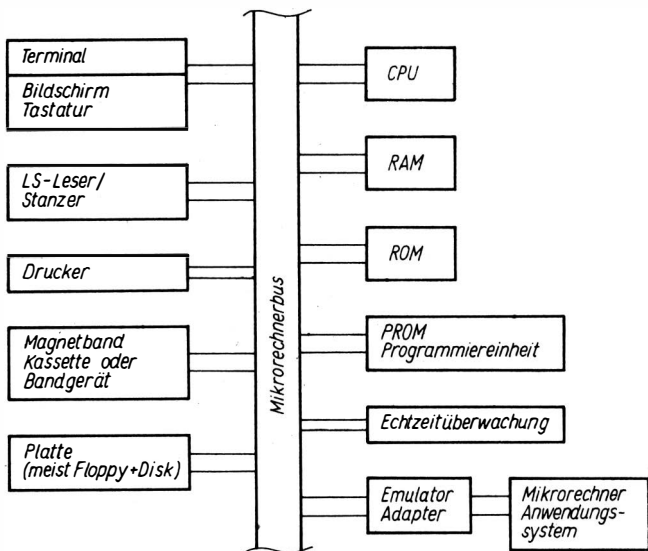


Bild 7.1 Struktur eines Mikrorechnerentwicklungssystems

Dienstprogramme zur Programmaufbereitung werden in Mikrorechnerentwicklungssystemen eingesetzt. Mikrorechnerentwicklungssysteme sind speziell für die universelle Unterstützung der Programmentwicklung vorgesehen.

Bild 7.1 zeigt die Struktur eines solchen Systems. Entwicklungssysteme sind für den universellen Dialogbetrieb ausgelegt. Sie besitzen die in Bild 7.1 angeführten Funktionsteile. Der Emulator-Adapter ermöglicht das Anschließen des zu entwickelnden Gerätes an das Entwicklungssystem. Damit können Programme in das Anwendersystem übertragen und im Anwendersystem unter Echtzeitbedingungen ablaufen. Der Ablauf kann durch die Echtzeitüberwachung kontrolliert und protokolliert werden. Der Dialog zur Programmentwicklung erfolgt im Wechselspiel zwischen Bediener und Rechner. Für diese Dialogarbeit gibt es verschiedene Ausbaustufen:

### Kommandosteuerung

Auf dem Bildschirm erscheint ein spezielles Kennzeichen (z. B. Punkt (.) oder Doppelkreuz (#)). Der Bediener gibt nun über die Tastatur ein Kommando.

Das Kommando besteht aus einem Operationsteil (meist abgekürzt), aus Parametern (durch Komma oder Lehrzeichen getrennt) und aus dem Abschlußzeichen (meist «Return»). Nach dem Geben des Abschlußzeichens führt das System das Kommando aus.

### Frage-Antwortsystem

Auf dem Bildschirm erscheint eine Frage, die durch Eingabe von Parametern beantwortet werden muß. Nach Eingabe der Parameter wird die entsprechende Funktion ausgelöst.

### Menütechnik

Auf dem Bildschirm erscheint eine Frage und, z. B. in Klammern gesetzt, die möglichen Antworten. Der Bediener muß jetzt nur eine der möglichen Antworten eintippen, um die entsprechende Funktion auszulösen. Meistens ist die Antwort einfach Ja (Y = YES) oder Nein (N = No).

Die Software zu einem Entwicklungssystem besteht im wesentlichen aus einem leistungsfähigen Betriebssystem und Dienstprogrammen wie

- |                     |                                       |
|---------------------|---------------------------------------|
| – Transferprogramme | – Echtzeitanalyseprogramme            |
| – Assembler         | – Programme zur Fileorganisation      |
| – Reassembler       | – Initialisierungsprogramme           |
| – Interpreter       | – Copier- und Konvertierungsprogramme |
| – Compiler          | – Bibliotheksorganisation             |
| – Simulatoren       | – Binder oder Taskbuildner            |
| – Emulatoren        | – Debugger (Fehlersuchprogramme)      |
|                     | – PROM-Programmerroutinen             |

(Dienstprogramme zur Systemüberprüfung)

- Testprogramme
- Fehlersuchroutinen

(Programme zur Datenaufbereitung)

- Textbehandlungsprogramme
- Datenbanksysteme
- Sortierprogramme

Betriebssysteme unterteilen sich je nach Anwendung in

- Programmentwicklungssysteme
- Echtzeitbetriebssysteme
- Timesharingsysteme
- Mehrprozessorbetriebssysteme
- Netzwerkbetriebssysteme
- Betriebssysteme für Spezialanwendungen  
(z. B. Feldrechner)

## 8. Schlußbetrachtung

Es wurden eine Reihe von Grundlagen der Mikrorechentechnik anhand der Bausteinreihe *U880* erläutert.

Die Mikrorechentechnik befindet sich zur Zeit noch in einer Phase starker Entwicklung. Durch die Möglichkeit, immer mehr Grundelemente auf einem Chip zu vereinigen, ist man bestrebt, sämtliche Funktionen eines Rechners durch integrierte Schaltkreise zu realisieren. Diese Entwicklung wird sich so lange fortsetzen, bis alle Funktionen durch integrierte Schaltkreise realisiert sind.

Zur Zeit kann man bereits Mikroprozessoren für 16 und 32 Bit Datenbreite realisieren. Andererseits läßt sich auf einem Chip ein ganzer Mikrocomputer mit Rechenwerk, Steuerwerk und Speicher unterbringen.

Hierzu gehören u. a. der Schaltkreis *Z8* und der Schaltkreis *8082*. Für die Realisierung der Speicherfunktion kann man heute dynamische RAM-Speicher mit 64 KBit und ROM-Schaltkreise mit 32 KBit herstellen. Dabei hat sich in den letzten Jahren die mögliche Kapazität pro Jahr nahezu vervierfacht.

Aus dem Sortiment der Mikrocomputer und Mikroprozessoren werden in der DDR die Schaltkreise Mikrocomputer *U881*, *U882* und Mikroprozessor *U8000* und *K1810 WM86* (UdSSR) mit folgenden peripheren und BUS-Schaltkreisen verwendet:

- 8 Bit Datenregister mit Tri-state-Ausgang ohne Invertierung
- 8 Bit Datenregister mit Tri-state-Ausgang und Invertierung
- Taktgenerator und Treiber für den Prozessor
- 8 Bit Bustreiber bidirektional, Tri-state, ohne Invertierung
- 8 Bit Bustreiber bidirektional, Tri-state, mit Invertierung
- Buscontroller für den Prozessor
- Programmierbare Interruptsteuereinheit
- Programmierbarer Ein-/Ausgabecontroller.

Durch die rasche Entwicklung der Halbleitertechnologie zu immer höheren Integrationsgraden entstehen auch Schaltkreise für komplizierte Rechnerfunktionen, z. B. Gleitkommaprozessoren, Schaltkreise für DA- und AD-Wandlung und Schaltkreise für mathematische Standardfunktionen wie Radizieren, Sinus, Cosinus, Tangens, Logarithmus und Potenzieren. Beispielsweise gibt es

zum Aufbau von Tischrechnern vollständige Arithmetikprozessoren. In vielen Tischrechnern werden sogenannte CAP-Chips (Cordic-Arithmetic-Prozessor-Chip) verwendet. Der Einsatz solcher Rechenschaltkreise erfordert ein grundsätzliches Umdenken in der Digitalelektronik. Funktionen werden nicht mehr verdrahtet, sondern programmiert. Es wird ein Grundsystem (eines Mikrorechners) aus Rechenschaltkreisen aufgebaut, und dieses System, das sehr universell einsetzbar ist, programmiert man dann auf den Einsatzfall. Der Aufbau des Mikrorechnersystems geschieht dabei nach durch die Schaltkreisfamilien vorgegebenen Regeln. Lediglich vom Umfang des Systems her ergeben sich bestimmte Unterschiede. Man kann Mikrorechnersysteme entsprechend der Größe des Einsatzfalles einteilen in:

- a) Einkartenrechner,
- b) Mikrorechnersysteme,
- c) Mehrprozessorsysteme,
- d) Hierarchische Rechnersysteme mit verteilter Intelligenz.

## **8.1. Einkartenrechner**

Die einfachste Variante eines Mikrorechners besteht in der Zusammenschaltung eines Mikroprozessors mit Speicherbausteinen und mit Ein- und Ausgabebausteinen zur Verbindung mit externen Geräten. Die dazugehörigen Schaltkreise lassen sich bei modernen mikroelektronischen Bausteinen unmittelbar zusammenschalten und auf einer Leiterkarte unterbringen. Man erhält auf diese Weise den Einkartenrechner, der als Steuerrechner für viele Anwendungsfälle ausreicht.

Bild 8.1 zeigt die prinzipielle Schaltung eines solchen Einkartenrechners mit dem Prozessor *U880*, dem PIO-Baustein und mit 2 K PROM und 1 K RAM.

Der Taktgenerator für diesen Rechner braucht bei Frequenzen unter 1 MHz nicht stabilisiert zu werden. Der angegebene Leiterkartenrechner läßt sich sehr vielseitig anwenden.

Zum Beispiel ist es möglich, ihn zur Steuerung einer elektrischen Schreibmaschine, deren Typenhebel durch Magnete ausgelöst werden, zu verwenden. Man kann auf diese Weise eine Schreibmaschine entwickeln, die den Text von einer Seite speichert. Durch eine geeignete Programmierung erreicht man, daß in dem gespei-

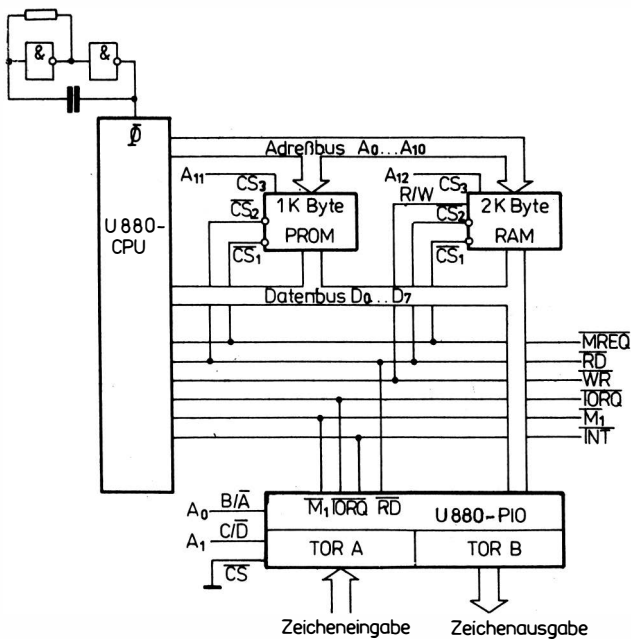


Bild 8.1 Struktur eines Einkartenrechners mit dem Prozessor *U 880* zur Schreibmaschinensteuerung

cherten Text beliebig korrigiert werden kann. Am Schluß einer Korrektur läßt sich der Inhalt so ausschreiben, wie er benötigt wird. Auf ganz ähnliche Weise kann man einen solchen Einkartenrechner zur Steuerung von Zeichengeräten, in Oszillographen sowie in Datenerfassungsgeräten einsetzen.

## 8.2. Mikrorechnersystem

Das Mikrorechnersystem ist ein abgestimmtes Sortiment von Leiterkarten mit unterschiedlichen Funktionen, die an einen zentralen Bus, den sogenannten Systembus, angeschlossen sind (Bild 8.2). Der Systembus unterteilt sich in

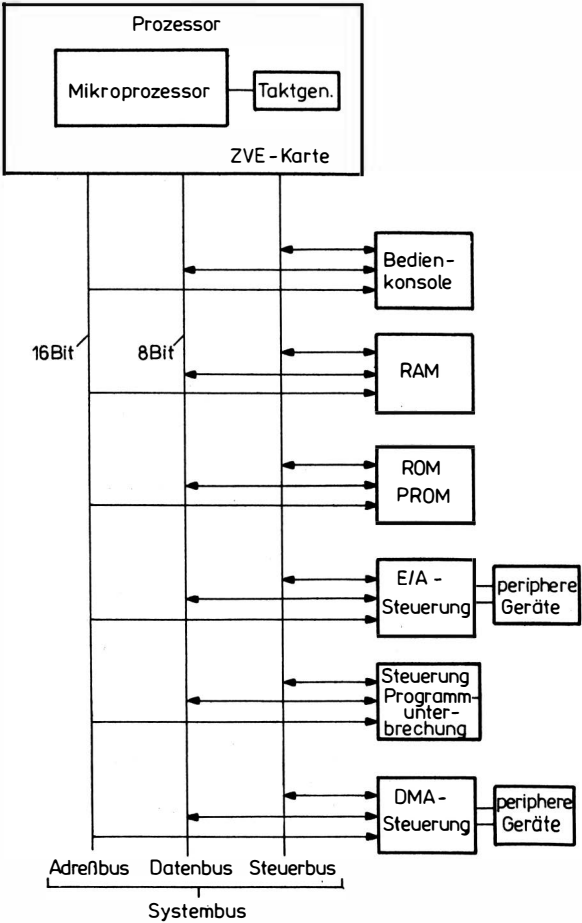


Bild 8.2 Aufbau eines Mikrorechnersystems

Adreßbus,  
Datenbus,  
Steuerbus.

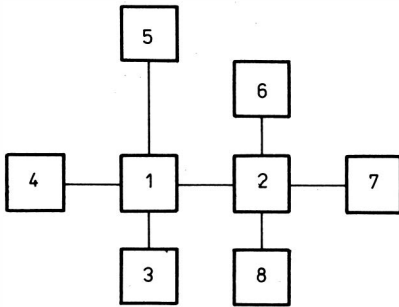
Der Adreßbus enthält eine Speicheradresse oder die Adresse eines Ein- und Ausgabetores. Über den Datenbus wird der Datenaustausch zwischen den einzelnen Moduln (Leiterkarten) vollzogen. Der Datenbus ist bidirectional (arbeitet in beiden Richtungen!). Der Steuerbus enthält eine Reihe von Steuersignalen zur Bedienung der adressierten Module sowie Meldesignale über den momentanen Zustand (z. B. bereit, nicht bereit, fehlerhaft) des adressierten Moduls.

### **8.3. Mehrprozessorsysteme**

Da im Mikroprozessor der vollständige Rechnerkern als Chip zur Verfügung gestellt wird, können bei relativ geringen Kosten in einem System mehrere Prozessoren verwendet werden. Die Verwendung von mehreren Prozessoren hat aber nur dann einen Sinn, wenn von der Aufgabenstellung her eine entsprechende Aufteilung in Teilaufgaben möglich ist. Die Aufteilung in Teilaufgaben kann dabei z. B. so aussehen, daß eine größere Aufgabe in Teilaufgaben zerlegt wird und jeder Mikroprozessor praktisch eine solche Aufgabe abarbeitet.

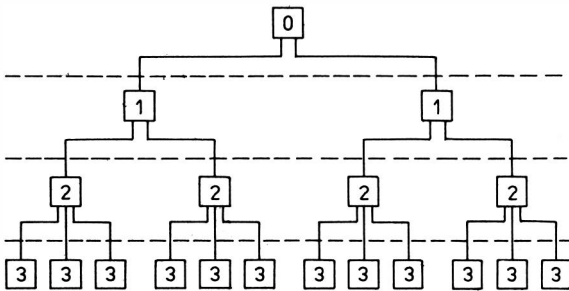
Man kommt durch diese Aufteilung zu sogenannten Hierarchischen Systemen. Zur Kommunikation der einzelnen Prozessoren miteinander (Austausch von Parametern, Aufgabenteilung und Fertigmeldung) müssen diese entsprechend verbunden werden. Dafür gibt es eine ganze Reihe von Zusammenschaltungsprinzipien.

Bild 8.3 zeigt einige Möglichkeiten der Zusammenschaltung von Prozessoren. Bei der sternförmigen Zusammenschaltung (Bild 8.3a) läuft der Informationsaustausch über den Sternpunkt. Jeder Prozessor ist am Sternpunkt angeschlossen. Er kann seine Information über diesen an andere Prozessoren weitergeben. In Bild 8.3b ist eine hierarchische Struktur dargestellt. In dieser gibt es zu einer Reihe von Prozessoren einen übergeordneten Prozessor, über den die entsprechenden Aufgaben verteilt und die Ergebnisse in Empfang genommen werden. Die hierarchische Struktur kann auch als Mehrebenenstruktur aufgebaut werden. Bei der Busstruk-



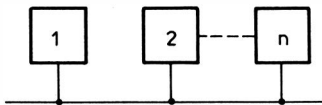
Sternstruktur

a)

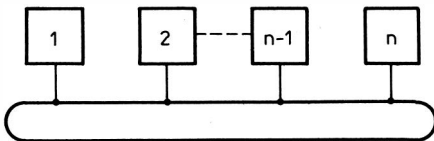


Hierarchiestruktur

b)

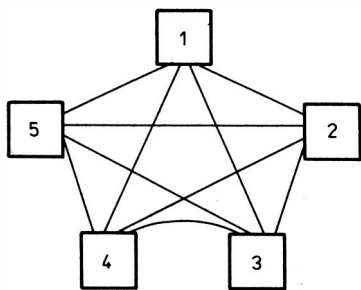


Busstruktur



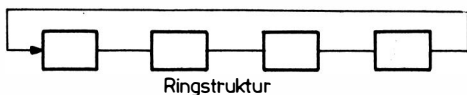
Ringbus

c)



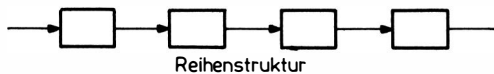
Netzstruktur

d)



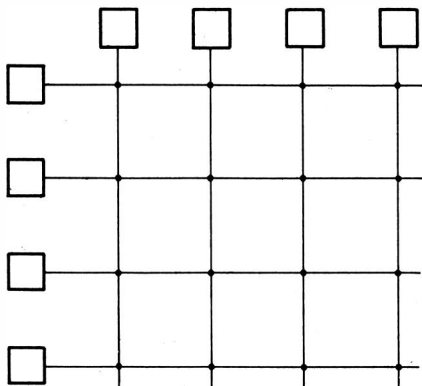
Ringstruktur

e)



Reihenstruktur

f)



Matrixstruktur

g)

Bild 8.3 Schaltungsvarianten für Mehrprozessorsysteme;

a – Sternstruktur, b – Hierarchiestruktur, c – Busstruktur, d – Netzstruktur, e – Ringstruktur, f – Reihenstruktur, g – Matrixstruktur

tur (Bild 8.3c) sind die einzelnen Prozessoren an einen zentralen BUS angeschlossen. Damit ein Prozessor mit einem 2. Prozessor in Verbindung treten kann, muß vom 1. Prozessor eine Busanforderung gestellt werden. Nachdem die Busanforderung angenommen wurde, wird der aufrufende Prozessor zum Master und die Gegenstelle zum Slaver erklärt. Der Bus ist nun für den Datenaustausch frei.

Das System läßt sich so aufbauen, daß alle Elemente gleichberechtigt sind oder daß für die Busanforderung ein Prioritätssystem existiert. Der Nachteil dieser Schaltung besteht darin, daß immer nur 2 Elemente gleichzeitig miteinander arbeiten können.

Das Bussystem ist z. B. in der SKR-Reihe (System von Kleinrechnern) verbreitet.

Bei der Netzstruktur (Bild 8.3d) ist jedes Prozessorelement mit jedem verbunden. Die Struktur gestattet einen flexiblen Einsatz, jedoch ist sie sehr aufwendig in den Verbindungsstrecken. In Bild 8.3e ist die Ringstruktur dargestellt. Bei dieser Struktur werden die Information und die Zieladresse über den Ringbus gegeben. Das Prozeßelement mit der gewünschten Adresse übernimmt die über den Bus laufenden Informationen. Dieses System ist für den Aufbau eines Netzes, bei dem die einzelnen Prozessorelemente örtlich weiter voneinander aufgestellt sein können, sehr zweckmäßig, z. B. die komplexe Steuerung von Maschinen in Werkhallen oder Industrieeinrichtungen. Das System wird im international bekannten CAMAC-System angewendet.

Die Reihenstruktur (Bild 8.3f) ähnelt der Ringstruktur, jedoch ist die Kette nicht geschlossen.

Aus Bild 8.3g ist die Matrixstruktur zu ersehen. Bei ihr läßt sich jedes Prozeßelement mit jedem anderen über die Matrixknotenpunkte verbinden. Die Knotenpunkte sind als elektronische Schalter ausgeführt, die von einem Steuerrechner des Netzwerks durchgeschaltet werden.

Eine weitere Anwendung der Mehrprozessorsysteme ist die Parallelverarbeitung. Dabei werden mehrere Prozessoren so in Kette zusammengefügt, daß sie alle den gleichen Anteil zur Realisierung eines Vorgangs leisten. Als Beispiel stelle man sich eine Stahlplatte vor, die an einer Seite ( $x$ -Richtung) ungleichmäßig erwärmt wird. Das System soll nun ermitteln, wie sich die Erwärmung in  $y$ -Richtung fortpflanzt. Zu diesem Zweck wird die  $x$ -Richtung in einzelne Teilabschnitte aufgeteilt ( $x_1$  bis  $x_n$ ). Jeden Teilabschnitt stellt man

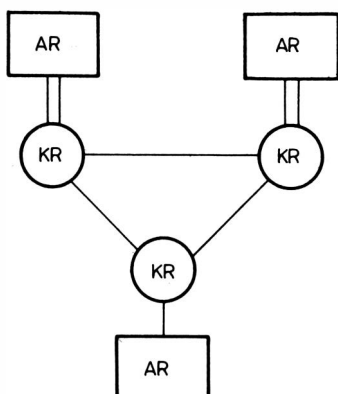
durch ein Prozessorelement dar. Das Programm der einzelnen Elemente basiert in jedem Prozessor auf den Regeln der Wärmeleitung, so daß die Verteilung in y-Richtung als Zeitkonstante bei der Programmbearbeitung dargestellt wird.

## 8.4. Hierarchische Rechnersysteme (Rechnernetze)

Bei den Rechnernetzen werden Mikrorechnersysteme miteinander gekoppelt. Die Kopplung dient dem Zweck der besseren Verteilung der Aufgaben zur gleichmäßigen Auslastung der einzelnen Rechner. Das wichtigste Element eines solchen Rechnernetzes ist das dazugehörige Betriebssystem, das die Aufgaben und die Geräte verwaltet und steuert. Auf diese Art und Weise können große Aufgaben mit umfangreichem Datenmaterial gelöst werden.

Bild 8.4 zeigt die Struktur eines solchen Rechnernetzes. Zum Netz gehören die Arbeitsrechnersysteme und ein sogenannter Kommunikationsrechner zur Steuerung der Verbindungen.

Der häufigste Anwendungsfall der Rechenschaltkreise wird der Einkartenrechner sein. Er dient als eingebauter Steuerrechner für einzelne Geräte. Das Mikrorechnersystem wird als Prozeßrechner bei größeren Steuerungsproblemen angewendet werden. Mehr-



KR = Kommunikationsrechner  
AR = Arbeitsrechner

Bild 8.4  
Struktur eines Rechnernetzes

prozessorsysteme werden seltener auftreten. Sie sind dort von Nutzen, wenn es um die Nachbildung von physikalischen Modellen geht. Hierarchische Rechnersysteme finden in größeren Systemen Anwendung, z. B. als Verbundnetz bei einer Flugplatzüberwachung.

Bei der Frage nach der möglichen Anwendung von Mikroprozessoren kann man heute schon antworten: Sie sind überall einsetzbar. Wichtig für ein erfolgreiches Anwenden ist die Auswahl des geeigneten Prozessors und des geeigneten Systems sowie der verfügbaren Programme. Die vorliegenden Hefte sollen dazu eine gewisse Hilfestellung leisten.

## 9. Literatur

- Kühn, E.; Schmied, H.:* Integrierte Schaltkreise. Berlin 1972.  
...: Mikroprozessoren, Elektronik-Sonderausgabe, München 1977.
- Gößler, R.:* Ein-/Ausgabe-Bausteine für Mikroprozessoren. In: Elektronik 1976.  
...: Betriebsdokumentation des Mikrorechnersystems K 1510. Dokumentation des VEB Kombinat Robotron, Dresden, 1977.  
...: Z 80 – Entwicklungssystem. Firmenunterlagen Mostek, Wien 1977.
- NN: Mikroprozessor-Schaltkreise 8080. Firmendokumentation Siemens, München 1977.
- Dawidczak, S.; Weise, K. D.:* Mikrorechnersystem robotron K 1520. In: msr 20 (1977), H. 12.
- NN: Mikroprozessoren, Mikrorechner. Literaturinformation des VEB Kombinat Robotron (ZfT), Dresden, 1977.
- Kanton, D.:* Mikroprozessorsystem. Reihe Automatisierungstechnik, Band 183, Berlin 1978.
- Meiling, W.:* Einige Tendenzen in der Entwicklung und Anwendung von Mikroprozessoren und Mikrorechnern. In: Nachrichtentechnik 27 (1977), H. 1.
- NN: Mikrorechnersystem K 1520 – Systembeschreibung. Firmendokumentation 1978, VEB Kombinat Robotron.
- Weller, W.:* Anwendung der Mikroelektronik in der Prozeßautomatisierung. Reihe Automatisierungstechnik, Band 187, Berlin 1981.
- Bader, B.:* Mikrorechnersystem K 1520. In: radio fernsehen elektronik 28 (1979), H. 10, S. 616–620.
- Jugel, A.:* Mikroprozessorsysteme. Berlin 1980.
- Claßen, L.:* Programmierung des Mikroprozessorsystems U 880 – K 1520. Reihe Automatisierungstechnik, Band 192, Berlin 1981.
- Schwarz, W.; Meyer, G.; Eckhard, D.:* Mikrorechner – Wirkungsweise, Programmierung, Applikation. Berlin 1980.
- TGL 35333: Statischer Lese-Schreib-Speicher U 202 D.
- TGL 34815: Unipolarer Festwertspeicherschaltkreis U 505 D.

*TGL 37787: Festwertspeicherschaltkreis U 555 C.*

*TGL 26276: Unipolarer Mikroprozessorschaltkreis U 880 D.*

*TGL 35837: Unipolarer Parallel-Eingabe-Ausgabe-Schaltkreis U 855 D.*

*TGL 37001: Unipolarer Serieller Eingabe-Ausgabe-Schaltkreis U 856 D.*

*TGL 37002: Unipolarer Zähler-/Zeitgeberschaltkreis U 857 D.*

## 10. Anhang

**Tabelle 10.1.** Darstellung der Zahlen 0 bis 32 in dezimaler, dualer, oktaler, hexadezimaler und BCD-Form

| dezimal | dual   | oktal | hexadezimal | BCD       |
|---------|--------|-------|-------------|-----------|
| 0       | 000000 | 0     | 0           | 0000 0000 |
| 1       | 000001 | 1     | 1           | 0000 0001 |
| 2       | 000010 | 2     | 2           | 0000 0010 |
| 3       | 000011 | 3     | 3           | 0000 0011 |
| 4       | 000100 | 4     | 4           | 0000 0100 |
| 5       | 000101 | 5     | 5           | 0000 0101 |
| 6       | 000110 | 6     | 6           | 0000 0110 |
| 7       | 000111 | 7     | 7           | 0000 0111 |
| 8       | 001000 | 10    | 8           | 0000 1000 |
| 9       | 001001 | 11    | 9           | 0000 1001 |
| 10      | 001010 | 12    | A           | 0001 0000 |
| 11      | 001011 | 13    | B           | 0001 0001 |
| 12      | 001100 | 14    | C           | 0001 0010 |
| 13      | 001101 | 15    | D           | 0001 0011 |
| 14      | 001110 | 16    | E           | 0001 0100 |
| 15      | 001111 | 17    | F           | 0001 0101 |
| 16      | 010000 | 20    | 10          | 0001 0110 |
| 17      | 010001 | 21    | 11          | 0001 0111 |
| 18      | 010010 | 22    | 12          | 0001 1000 |
| 19      | 010011 | 23    | 13          | 0001 1001 |
| 20      | 010100 | 24    | 14          | 0010 0000 |
| 21      | 010101 | 25    | 15          | 0010 0001 |
| 22      | 010110 | 26    | 16          | 0010 0010 |

| dezimal | dual   | oktal | hexadezimal | BCD       |
|---------|--------|-------|-------------|-----------|
| 23      | 010111 | 27    | 17          | 0010 0011 |
| 24      | 011000 | 30    | 18          | 0010 0100 |
| 25      | 011001 | 31    | 19          | 0010 0101 |
| 26      | 011010 | 32    | 1A          | 0010 0110 |
| 27      | 011011 | 33    | 1B          | 0010 0111 |
| 28      | 011100 | 34    | 1C          | 0010 1000 |
| 29      | 011101 | 35    | 1D          | 0010 1001 |
| 30      | 011110 | 36    | 1E          | 0011 0000 |
| 31      | 011111 | 37    | 1F          | 0011 0001 |
| 32      | 100000 | 40    | 20          | 0011 0010 |

**Tabelle 10.2.** Addition hexadezimaler Zahlen

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9  | A  | B  | C  | D  | E  | F  |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9  | A  | B  | C  | D  | E  | F  |
| 1 |   | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  | A  | B  | C  | D  | E  | F  | 10 |
| 2 |   |   | 4 | 5 | 6 | 7 | 8 | 9 | A  | B  | C  | D  | E  | F  | 10 | 11 |
| 3 |   |   |   | 6 | 7 | 8 | 9 | A | B  | C  | D  | E  | F  | 10 | 11 | 12 |
| 4 |   |   |   |   | 8 | 9 | A | B | C  | D  | E  | F  | 10 | 11 | 12 | 13 |
| 5 |   |   |   |   |   | A | B | C | D  | E  | F  | 10 | 11 | 12 | 13 | 14 |
| 6 |   |   |   |   |   |   | C | D | E  | F  | 10 | 11 | 12 | 13 | 14 | 15 |
| 7 |   |   |   |   |   |   |   | E | F  | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 8 |   |   |   |   |   |   |   |   | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 9 |   |   |   |   |   |   |   |   |    | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| A |   |   |   |   |   |   |   |   |    |    | 14 | 15 | 16 | 17 | 18 | 19 |
| B |   |   |   |   |   |   |   |   |    |    |    | 16 | 17 | 18 | 19 | 1A |
| C |   |   |   |   |   |   |   |   |    |    |    |    | 18 | 19 | 1A | 1B |
| D |   |   |   |   |   |   |   |   |    |    |    |    |    | 1A | 1B | 1C |
| E |   |   |   |   |   |   |   |   |    |    |    |    |    |    | 1C | 1D |
| F |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    | 1E |

**Tabelle 10.3. Multiplikation hexadezimaler Zahlen**

|   | 0 | 1 | 2 | 3 | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  |
|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 0 | 1 | 2 | 3 | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  |
| 2 |   | 2 | 4 | 6 | 8  | A  | C  | E  | 10 | 12 | 14 | 16 | 18 | 1A | 1C | 1E |
| 3 |   |   | 6 | 9 | C  | F  | 12 | 15 | 18 | 1B | 1E | 21 | 24 | 27 | 2A | 2D |
| 4 |   |   |   | C | 10 | 14 | 18 | 1C | 20 | 24 | 28 | 2C | 30 | 34 | 38 | 3C |
| 5 |   |   |   |   | 14 | 19 | 1E | 23 | 28 | 2D | 32 | 37 | 3C | 41 | 46 | 4B |
| 6 |   |   |   |   |    | 1E | 24 | 2A | 30 | 36 | 3C | 42 | 48 | 4E | 54 | 5A |
| 7 |   |   |   |   |    |    | 2A | 31 | 38 | 3F | 46 | 4D | 54 | 5B | 62 | 69 |
| 8 |   |   |   |   |    |    |    | 38 | 40 | 48 | 50 | 58 | 60 | 68 | 70 | 78 |
| 9 |   |   |   |   |    |    |    |    | 48 | 51 | 5A | 63 | 60 | 75 | 7E | 87 |
| A |   |   |   |   |    |    |    |    |    | 5A | 64 | 6E | 78 | 82 | 8C | 96 |
| B |   |   |   |   |    |    |    |    |    |    | 6E | 79 | 84 | 8F | 9A | A5 |
| C |   |   |   |   |    |    |    |    |    |    |    | 84 | 90 | 9C | AB | B4 |
| D |   |   |   |   |    |    |    |    |    |    |    |    | 9C | A9 | B6 | C3 |
| E |   |   |   |   |    |    |    |    |    |    |    |    |    | B6 | C4 | D2 |
| F |   |   |   |   |    |    |    |    |    |    |    |    |    |    | D2 | E1 |

**Tabelle 10.4. Umrechnung hexadezimal → dezimal**

|   | 16 <sup>0</sup> | 16 <sup>1</sup> | 16 <sup>2</sup> | 16 <sup>3</sup> | 16 <sup>4</sup> |
|---|-----------------|-----------------|-----------------|-----------------|-----------------|
| 0 | 0               | 0               | 0               | 0               | 0               |
| 1 | 1               | 16              | 256             | 4.096           | 65.536          |
| 2 | 2               | 32              | 512             | 8.192           | 131.072         |
| 3 | 3               | 48              | 768             | 12.288          | 196.608         |
| 4 | 4               | 64              | 1.024           | 16.384          | 262.144         |
| 5 | 5               | 80              | 1.280           | 20.480          | 327.680         |
| 6 | 6               | 96              | 1.536           | 24.576          | 393.216         |
| 7 | 7               | 112             | 1.792           | 28.672          | 458.752         |
| 8 | 8               | 128             | 2.048           | 32.768          | 524.288         |
| 9 | 9               | 144             | 2.304           | 36.864          | 589.824         |
| A | 10              | 160             | 2.560           | 40.960          | 655.360         |
| B | 11              | 176             | 2.816           | 45.056          | 720.896         |
| C | 12              | 192             | 3.072           | 49.152          | 786.432         |
| D | 13              | 208             | 3.328           | 53.248          | 851.968         |
| E | 14              | 224             | 3.584           | 57.344          | 917.504         |
| F | 15              | 240             | 3.840           | 61.440          | 983.040         |

**Tabelle 10.5.** Umrechnung dezimal → hexadezimal

|   | $10^0$ | $10^1$ | $10^2$ | $10^3$ | $10^4$ |
|---|--------|--------|--------|--------|--------|
| 0 | 0      | 0      | 0      | 0      | 0      |
| 1 | 1      | A      | 64     | 3E8    | 2.710  |
| 2 | 2      | 14     | C8     | 7D0    | 4.E20  |
| 3 | 3      | 1E     | 12C    | BB8    | 7.530  |
| 4 | 4      | 28     | 190    | FA0    | 9.C40  |
| 5 | 5      | 32     | 1E4    | 1.388  | C.350  |
| 6 | 6      | 3C     | 248    | 1.770  | E.A60  |
| 7 | 7      | 46     | 2AC    | 1.B58  | 11.170 |
| 8 | 8      | 50     | 320    | 1.F40  | 13.880 |
| 9 | 9      | 5A     | 384    | 2.238  | 15.F90 |

**Tabelle 10.6** Umrechnung hexadezimal  $\rightarrow$  oktal

| 0  | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | A   | B   | C   | D   | E   | F   |
|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0  | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  |
| 10 | 20  | 21  | 22  | 23  | 24  | 25  | 26  | 27  | 30  | 31  | 32  | 33  | 34  | 35  | 36  | 37  |
| 20 | 40  | 41  | 42  | 43  | 44  | 45  | 46  | 47  | 50  | 51  | 52  | 53  | 54  | 55  | 56  | 57  |
| 30 | 60  | 61  | 62  | 63  | 64  | 65  | 66  | 67  | 70  | 71  | 72  | 73  | 74  | 75  | 76  | 77  |
| 40 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 |
| 50 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 |
| 60 | 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 |
| 70 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 170 | 171 | 172 | 173 | 174 | 175 | 176 | 177 |
| 80 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 |
| 90 | 220 | 221 | 222 | 223 | 224 | 225 | 226 | 227 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 |
| A0 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 250 | 251 | 252 | 253 | 254 | 255 | 256 | 257 |
| B0 | 260 | 261 | 262 | 263 | 264 | 265 | 266 | 267 | 270 | 271 | 272 | 273 | 274 | 275 | 276 | 277 |
| C0 | 300 | 301 | 302 | 303 | 304 | 305 | 306 | 307 | 310 | 311 | 312 | 313 | 314 | 315 | 316 | 317 |
| D0 | 320 | 321 | 322 | 323 | 324 | 325 | 326 | 327 | 330 | 331 | 332 | 333 | 334 | 335 | 336 | 337 |
| E0 | 340 | 341 | 342 | 343 | 344 | 345 | 346 | 347 | 350 | 351 | 352 | 353 | 354 | 355 | 356 | 357 |
| F0 | 360 | 361 | 362 | 363 | 364 | 365 | 366 | 367 | 370 | 371 | 372 | 373 | 374 | 375 | 376 | 377 |

Tabelle 10.7 Potenzen der Basis 16

| 16 <sup>n</sup> |  |     | n | 16 <sup>n</sup> |    |         |       |       |                     |
|-----------------|--|-----|---|-----------------|----|---------|-------|-------|---------------------|
|                 |  |     |   | 1               | 0  | 0.10000 | 00000 | 00000 | x 10                |
|                 |  |     |   | 16              | 1  | 0.62500 | 00000 | 00000 | x 10 <sup>-1</sup>  |
|                 |  |     |   | 256             | 2  | 0.39062 | 50000 | 00000 | x 10 <sup>-2</sup>  |
|                 |  | 4   |   | 096             | 3  | 0.24414 | 06250 | 00000 | x 10 <sup>-3</sup>  |
|                 |  | 65  |   | 536             | 4  | 0.15258 | 78906 | 00000 | x 10 <sup>-4</sup>  |
|                 |  | 1   |   | 576             | 5  | 0.95367 | 43164 | 00000 | x 10 <sup>-6</sup>  |
|                 |  | 16  |   | 216             | 6  | 0.59604 | 64477 | 06250 | x 10 <sup>-7</sup>  |
|                 |  | 268 |   | 456             | 7  | 0.37252 | 90298 | 53906 | x 10 <sup>-8</sup>  |
|                 |  | 4   |   | 294             | 8  | 0.23283 | 06436 | 46191 | x 10 <sup>-9</sup>  |
|                 |  | 68  |   | 719             | 9  | 0.14551 | 91522 | 53869 | x 10 <sup>-10</sup> |
|                 |  | 1   |   | 511             | 10 | 0.90949 | 47017 | 83668 | x 10 <sup>-12</sup> |
|                 |  | 17  |   | 186             | 11 | 0.56843 | 41886 | 72928 | x 10 <sup>-13</sup> |
|                 |  | 281 |   | 976             | 12 | 0.35527 | 13678 | 08080 | x 10 <sup>-14</sup> |
|                 |  | 4   |   | 503             | 13 | 0.22204 | 46049 | 80050 | x 10 <sup>-15</sup> |
|                 |  | 72  |   | 057             | 14 | 0.13877 | 78780 | 25031 | x 10 <sup>-16</sup> |
|                 |  | 152 |   | 921             | 15 | 0.86736 | 17379 | 78144 | x 10 <sup>-18</sup> |
| 1               |  |     |   | 976             |    |         |       | 88403 | x                   |

**Tabelle 10.8** Potenzen der Basis 10 in hexadezimaler Darstellung

|      | $10^n$ | n  | $10^{-n}$ |       |      |      |   |            |  |
|------|--------|----|-----------|-------|------|------|---|------------|--|
|      | 1      | 0  | 1.0000    | (0000 | 0000 | 0000 |   |            |  |
|      | A      | 1  | 0.1999    | 9999  | 9999 | 999A |   |            |  |
|      | 64     | 2  | 0.28F5    | C28F  | 5C28 | F5C3 | x | $16^{-1}$  |  |
|      | 3E8    | 3  | 0.4189    | 374B  | C6A7 | EF9E | x | $16^{-2}$  |  |
|      | 2710   | 4  | 0.68DB    | 8BAC  | 710C | B296 | x | $16^{-3}$  |  |
| I    | 86A0   | 5  | 0.A7C5    | AC47  | 1B47 | 8423 | x | $16^{-4}$  |  |
| F    | 4240   | 6  | 0.10C6    | F7A0  | B5ED | 8D37 | x | $16^{-4}$  |  |
| 98   | 9680   | 7  | 0.1AD7    | F29A  | BCAF | 4858 | x | $16^{-5}$  |  |
| 5F5  | E100   | 8  | 0.2AF3    | 1DC4  | 6118 | 73BF | x | $16^{-6}$  |  |
| 3B9A | CA00   | 9  | 0.44B8    | 2FA0  | 9B5A | 52CC | x | $16^{-7}$  |  |
| 540B | E400   | 10 | 0.6DF3    | 7F67  | SEF6 | EADF | x | $16^{-8}$  |  |
| 4876 | E800   | 11 | 0.AFEB    | FFOB  | CB24 | AAFF | x | $16^{-9}$  |  |
| D4A5 | 1000   | 12 | 0.1197    | 9981  | 2DEA | 1119 | x | $16^{-9}$  |  |
| 4E72 | A000   | 13 | 0.1C25    | C268  | 4976 | 81C2 | x | $16^{-10}$ |  |
| 107A | 4000   | 14 | 0.2D09    | 370D  | 4257 | 3604 | x | $16^{-11}$ |  |
| A4C6 | 8000   | 15 | 0.480E    | BE7B  | 9D58 | 566D | x | $16^{-12}$ |  |
| 6FC1 | 0000   | 16 | 0.734A    | CA5F  | 6226 | F0AE | x | $16^{-13}$ |  |
| 5D8A | 0000   | 17 | 0.B877    | AA32  | 36A4 | B449 | x | $16^{-14}$ |  |
| A764 | 0000   | 18 | 0.1272    | 5DD1  | D243 | ABAI | x | $16^{-14}$ |  |
| 89E8 | 0000   | 19 | 0.1D83    | C94F  | B6D2 | AC35 | x | $16^{-15}$ |  |

**Tabelle 10.9. ASCII-Code/SIF-1000-Code**

| Oktaler Code | ASCII-Zeichen | SIF-1000 Zeichen | Bemerkungen                                                                     |
|--------------|---------------|------------------|---------------------------------------------------------------------------------|
| 000          | NUL           | NUL              | Null, Leer                                                                      |
| 001          | SOH           |                  | Anfang des Kopfes<br>(Start of heading)                                         |
| 002          | STX           |                  | Anfang des Textes<br>(Start of text)                                            |
| 003          | ETX           |                  | Ende des Textes (End of text)                                                   |
| 004          | EOT           |                  | Ende der Übertragung<br>(End of transmission)                                   |
| 005          | ENQ           |                  | Anfrage (Enquiry); Wer ist dort?                                                |
| 006          | ACK           |                  | Bestätigung (Acknowledge)                                                       |
| 007          | BEL           |                  | Akustisches Signal (Bell)                                                       |
| 010          | BS            |                  | Rücktaste, Rücksetzen um ein Zeichen<br>(Backspace)                             |
| 011          | HT            | HT               | Horizontaler Tabulator<br>(Horizontal tab)                                      |
| 012          | LF            | LF               | Zeilenvorschub (Line feed)                                                      |
| 013          | VT            |                  | Vertikaler Tabulator (Vertical tab)                                             |
| 014          | FF            |                  | Formatsteuerung (Form feed);<br>Vorschub auf die nächste Seite                  |
| 015          | CR            | CR               | Wagenrücklauf (Carriage return)                                                 |
| 016          | SO            |                  | Umschalten auf Großbuchstaben (Shift out); Umschalten auf rotes Farbband        |
| 017          | SI            |                  | Umschalten auf Kleinbuchstaben<br>(Shift in); Umschalten auf schwarzes Farbband |
| 020          | DLE           |                  | Umschaltung der Datenübertragung<br>(Data link escape)                          |
| 021          | DC1           | DC1              | Gerätesteuerzeichen 1                                                           |
| 022          | DC2           | DC2              | Gerätesteuerzeichen 2                                                           |
| 023          | DC3           | DC3              | Gerätesteuerzeichen 3                                                           |
| 024          | DC4           | DC4              | Gerätesteuerzeichen 4                                                           |
| 025          | NAK           |                  | Negative Bestätigung (Negativ acknowledge); Fehler                              |
| 026          | SYN           |                  | Synchronisation                                                                 |
| 027          | ETB           |                  | Ende des übertragenen Blockes<br>(End of transmission block)                    |
| 030          | CAN           |                  | auch LEM, Logisches Ende des Mediums<br>Annullierung (CANCEL)                   |
| 031          | EM            |                  | Ende des Mediums (End of Medium)<br>Ende des Datenträgers                       |
| 032          | SUB           |                  | Substitution                                                                    |
| 033          | ESC           |                  | Escape                                                                          |
| 034          | FS            |                  | File separator                                                                  |
| 035          | GS            |                  | Group separator                                                                 |

**Tabelle 10.9.** ASCII-Code/SIF-1000-Code (1. Fortsetzung)

| Oktaler Code | ASCII-Zeichen | SIF-1000 Zeichen | Bemerkungen                                                         |
|--------------|---------------|------------------|---------------------------------------------------------------------|
| 036          | RS            | NL               | Record separator; New line (NL)<br>Wagenrücklauf mit Zeilenvorschub |
| 037          | US            |                  | Unit separator                                                      |
| 040          | Sp            | Sp               | Leerzeichen (Space)                                                 |
| 041          | !             | !                | Ausrufezeichen (Exclamation point)                                  |
| 042          | "             | "                | Anführungszeichen                                                   |
| 043          |               |                  | Nummernzeichen                                                      |
| 044          | \$            |                  | Währungszeichen                                                     |
| 045          | %             | %                | Prozentzeichen                                                      |
| 046          | &             | &                | Ampersand                                                           |
| 047          | ,             | ,                | Hochkomma                                                           |
| 050          | (             | (                |                                                                     |
| 051          | )             | )                |                                                                     |
| 052          | *             | *                |                                                                     |
| 053          | +             | +                |                                                                     |
| 054          | ,             | ,                |                                                                     |
| 055          | -             | -                |                                                                     |
| 056          | .             | .                |                                                                     |
| 057          | /             | /                |                                                                     |
| 060          | 0             | 0                |                                                                     |
| 061          | 1             | 1                |                                                                     |
| 062          | 2             | 2                |                                                                     |
| 063          | 3             | 3                |                                                                     |
| 064          | 4             | 4                |                                                                     |
| 065          | 5             | 5                |                                                                     |
| 066          | 6             | 6                |                                                                     |
| 067          | 7             | 7                |                                                                     |
| 070          | 8             | 8                |                                                                     |
| 071          | 9             | 9                |                                                                     |
| 072          | :             | :                |                                                                     |
| 073          | ;             | ;                |                                                                     |
| 074          | <             | <                |                                                                     |
| 075          | =             | =                |                                                                     |
| 076          | >             | >                |                                                                     |
| 077          | ?             | ?                |                                                                     |
| 100          | a             | \$               |                                                                     |
| 101          | A             | A                |                                                                     |
| 102          | B             | B                |                                                                     |
| 103          | C             | C                |                                                                     |
| 104          | D             | D                |                                                                     |
| 105          | E             | E                |                                                                     |
| 106          | F             | F                |                                                                     |
| 107          | G             | G                |                                                                     |
| 110          | H             | H                |                                                                     |
| 111          | I             | I                |                                                                     |

**Tabelle 10.9.** ASCII-Code/SIF-1000-Code (2. Fortsetzung)

| Oktaler Code | ASCII-Zeichen | SIF-1000 Zeichen | Bemerkungen |
|--------------|---------------|------------------|-------------|
| 112          | J             | J                |             |
| 113          | K             | K                |             |
| 114          | L             | L                |             |
| 115          | M             | M                |             |
| 116          | N             | N                |             |
| 117          | O             | O                |             |
| 120          | P             | P                |             |
| 121          | Q             | Q                |             |
| 122          | R             | R                |             |
| 123          | S             | S                |             |
| 124          | T             | T                |             |
| 125          | U             | U                |             |
| 126          | V             | V                |             |
| 127          | W             | W                |             |
| 130          | X             | X                |             |
| 131          | Y             | Y                |             |
| 132          | Z             | Z                |             |
| 133          | [             | [                |             |
| 134          | \             | Ø                |             |
| 135          | ]             | ]                |             |
| 136          | ^             | ↑                |             |
| 137          | -             | -                |             |
| 140          | \             | \                |             |
| 141          | a             | a                |             |
| 142          | b             | b                |             |
| 143          | c             | c                |             |
| 144          | d             | d                |             |
| 145          | e             | e                |             |
| 146          | f             | f                |             |
| 147          | g             | g                |             |
| 150          | h             | h                |             |
| 151          | i             | i                |             |
| 152          | j             | j                |             |
| 153          | k             | k                |             |
| 154          | l             | l                |             |
| 155          | m             | m                |             |
| 156          | n             | n                |             |
| 157          | o             | o                |             |
| 160          | p             | p                |             |
| 161          | q             | q                |             |
| 162          | r             | r                |             |
| 163          | s             | s                |             |
| 164          | t             | t                |             |
| 165          | u             | u                |             |
| 166          | v             | v                |             |

**Tabelle 10.9.** ASCII-Code/SIF-1000-Code (3. Fortsetzung)

| Oktaler Code | ASCII-Zeichen | SIF-1000 Zeichen | Bemerkungen       |
|--------------|---------------|------------------|-------------------|
| 167          | w             | w                |                   |
| 170          | x             | x                |                   |
| 171          | y             | y                |                   |
| 172          | z             | z                |                   |
| 173          | {             | {                |                   |
| 174          |               |                  | Vertikaler Strich |
| 175          | }             | }                |                   |
| 176          | ~             | ¬                | Negation          |
| 177          | DEL           | DEL              | Delete            |