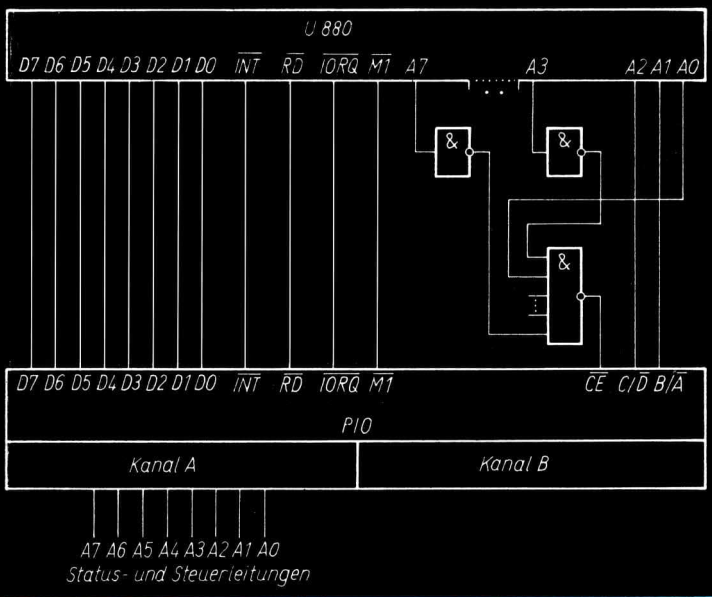
amateurreihe

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

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: 746693 2
00380

Inhaltsverzeichnis

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 (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 (Torungsimpuls)
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 Bildschirmeinheiten 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.

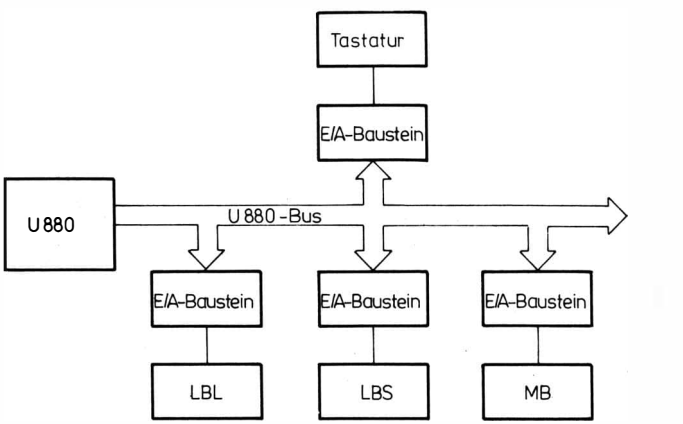


Bild 2.1 Anschluß von Bedienpult, Lochbandleser, Lochbandstanzer und Magnetbandgerät an den Prozessor *U 880*

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 BO 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.

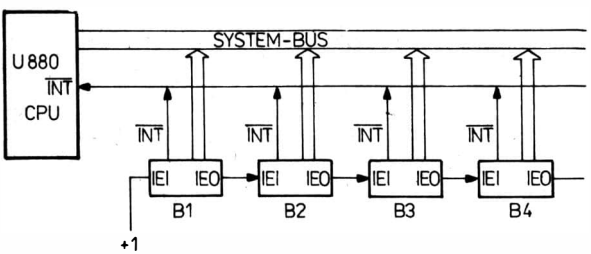


Bild 2.2 Zusammenschaltung von peripheren Bausteinen B1 bis B4 zu einer Interruptprioritätenkette

- 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-

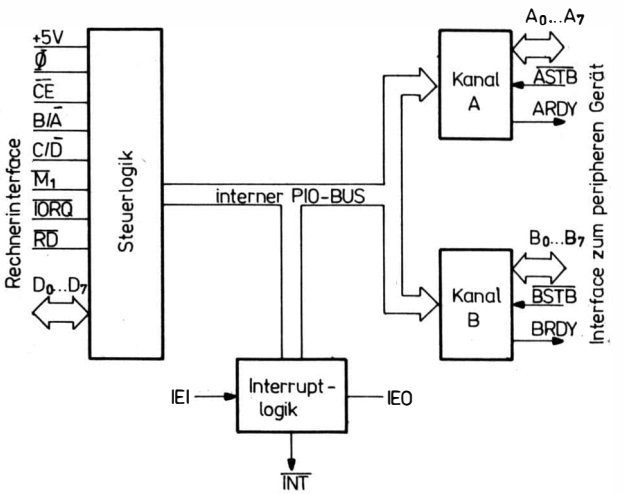


Bild 2.3 Grundstruktur des Bausteins *U 855 D*

teneingabe bzw. -ausgabe umzusetzen. Der Baustein verfügt über eine interne Steuerlogik, die mit Hilfe des Grundtaktes Φ 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.

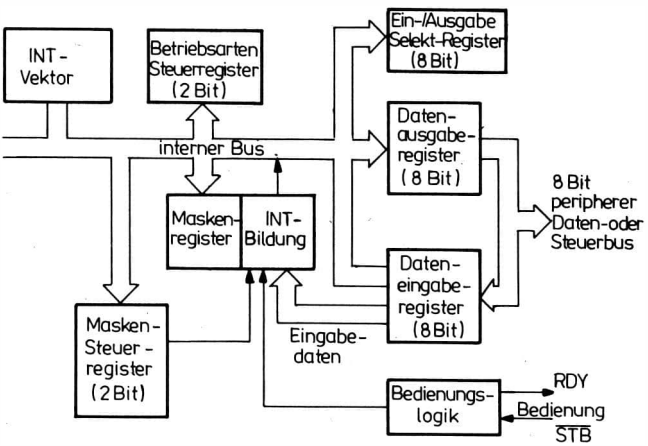


Bild 2.4 Logische Struktur eines Kanals des Bausteins *U 855 D*

2.2.2. Beschreibung der Anschlüsse des *U 855 D*

$D_0 \ldots D_7$ Datenbus zum Prozessor *U 880* (bidirectional, tristate).

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 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»).
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.
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»).

$A_0 \ldots A_7$ Datenbuskanal A. $A_0 \ldots A_7$ dient zur Ein- und Ausgabe der Information zwischen Kanal A und einem peripheren Gerät, A_0 ist die niederwertigste Stelle.

$\overline{\text{ASTB}}$ 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»).

$B_0 \ldots B_7$ Datenbuskanal B. $B_0 \ldots B_7$ dient zur Ein- und Ausgabe der Information zwischen Kanal B und einem peripheren Gerät. B_0 ist die niederwertigste Stelle.

$\overline{\text{BSTB}}$ 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 Löschsignal (RESET) gibt es nicht. Das Löschen der Register und der Flip-Flop des Bausteins geschieht entweder beim Anlegen der Betriebsspannung oder durch $\overline{\text{M1}}$, wenn nicht gleichzeitig eines der Signale $\overline{\text{RD}}$ bzw. $\overline{\text{IORQ}}$ 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 $\overline{\text{ASTB}}$, ARDY, $\overline{\text{BSTB}}$, BRDY werden inaktiv.

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

2.2.3. Arbeitsweise des *U 855 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_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
M	M	x	x	1	1	1	1

Die Stellen D_4 und D_5 sind ohne Bedeutung. Durch die Stellen D_6 und D_7 wird der Arbeitsmodus festgelegt.

	D_7	D_6	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_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	V_2	V_1	0

Das *Interruptsteuerwort* hat folgenden Aufbau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
M	x	x	x	0	0	1	1

Ist D_7 gleich 1, so wird die Interruptbildung erlaubt (Bildung des Signals INT). Ist D_7 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 $\overline{STB}$ ($\overline{ASTB}$, $\overline{BSTB}$) kommt. Mit dem Signal $\overline{STB}$ werden gewöhnlich die Daten vom Ausgaberegister in das periphere Gerät übernommen. Ist die Bildung einer Interruptanforderung erlaubt, dann wird durch $\overline{STB}$ das Signal $\overline{INT}$ erzeugt. Das Signal $\overline{INT}$ führt zu einer Programmunterbrechnung im Prozessor. Bei Annahme des Interrupt 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 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_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	V_2	V_1	0

Das *Interruptsteuerwort* hat folgenden Aufbau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
M	x	x	x	0	0	1	1

Ist D_7 gleich 1, so wird die Interruptbildung erlaubt (Bildung des Signals $\overline{INT}$). Ist D_7 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 $\overline{STB}$ ($\overline{ASTB}$ oder $\overline{BSTB}$) kommt. Mit der Vorderflanke des Signals $\overline{STB}$ können die Daten vom peripheren Gerät in das Eingaberegister übernommen werden.
Ist die Bildung einer Interruptanforderung erlaubt, so wird durch $\overline{STB}$ das Signal $\overline{INT}$ erzeugt. Das Signal $\overline{INT}$ führt zu einer Programmunterbrechnung 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 (ARDY, $\overline{ASTB}$) zur Ausgabe und die Signale von Kanal B (BRDY, $\overline{BSTB}$) zur Eingabe benutzt. Vor der Benutzung von Kanal A als bidirectionaler 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_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	V_2	V_1	0

Das *Interruptsteuerwort* hat wieder folgenden Aufbau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
M	x	x	x	0	0	1	1

Ist D_7 gleich 1, so wird die Interruptbildung erlaubt (Bildung des Signals $\overline{INT}$). Ist D_7 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 ARDY zur Kennzeichnung der Bereitschaft aktiv.
Eine Abnahme der Daten vom Ausgaberegister kann während der Zeit geschehen, in der $\overline{ASTB}$ und ARDY aktiv sind. Mit der Rückflanke von $\overline{ASTB}$ wird bei erlaubtem Interrupt das INT-Signal gebildet. Nach einem Eingabebefehl ist das Signal BRDY aktiv.
Mit dem Signal $\overline{BSTB}$ können neue Daten von einem peripheren Gerät in das Eingaberegister übernommen werden. Anschließend wird das Signal 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_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
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_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
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_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	V_2	V_1	0

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

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
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.

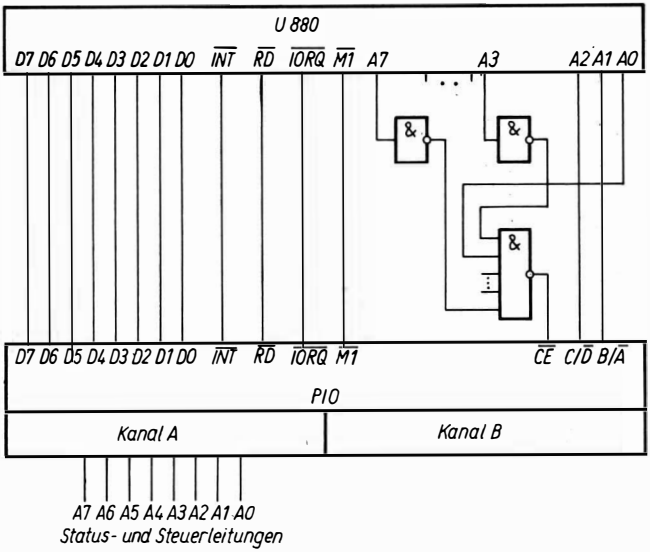


Bild 2.5 Zusammenschaltung eines PIO-Schaltkreises mit dem *U 880*

- 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-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 Zähl- 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 $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, $\overline{\text{M1}}$, Systemtakt Φ, $\overline{\text{RESET}}$, $\overline{\text{CE}}$, CS_1, CS_0. Zur Realisierung von Interruptketten verfügt der Baustein über die Signale IEI, IEO, $\overline{\text{INT}}$.

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.

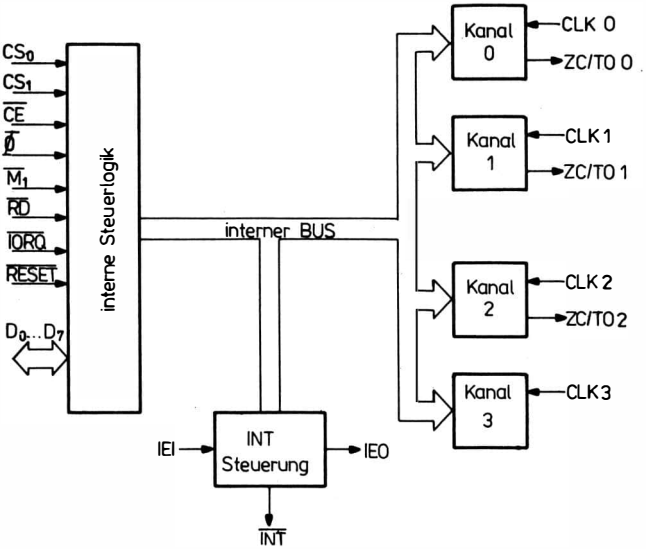


Bild 2.6 Grundstruktur des Bausteins *U 857 D*

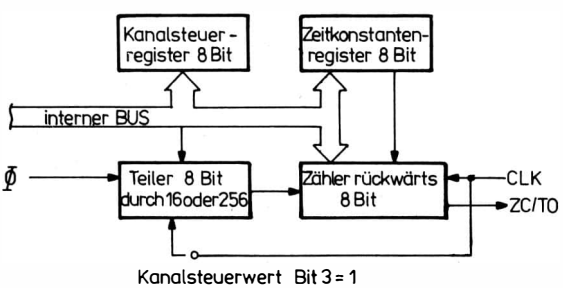


Bild 2.7 Logische Struktur eines Kanals des Bausteins *U 857 D*

2.3.2. Beschreibung der Anschlüsse des CTC-Bausteins

Im einzelnen haben die Anschlüsse folgende Bedeutung:

$D_0 \ldots D_7$ 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 Φ, 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 Φ 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

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

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

D_4
0 neg. Flanke, 1 pos. Flanke

D_3
Bestimmt Triggerzeitpunkt

} nur in der Betriebsart «Zeitgeber»

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 $\overline{\text{INT}}$-Ausgang aktiv, wenn die Interruptbildung erlaubt ist.

2.4. Serieller Ein-/Ausgabe-Baustein *U 856 D*

2.4.1. Struktur des *U 856 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.

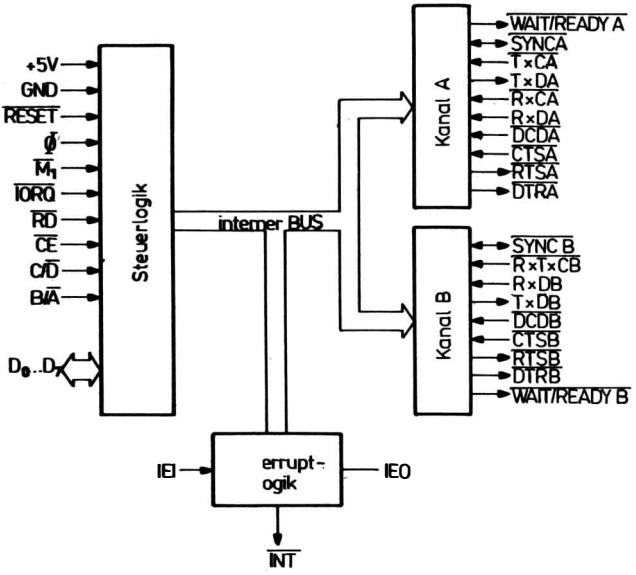


Bild 2.8 Grundstruktur des Bausteins *U 856 D*

2.4.2. Beschreibung der Anschlüsse des *U 856 D*

Signale des Rechnerinterface:

$D_0 \ldots D_7$	Datenbus zum Prozessor *U 880*.
B/$\overline{A}$ Eingangssignal	Kanalauswahl; «Low» ≙ Tor A; «High» ≙ Tor B.
C/$\overline{D}$ Eingangssignal	Auswahl Steuerwort/Datenwort: Dieses Signal legt fest, ob das vom Prozessor kommende Wort ein Steuerwort («High») oder ein Datenwort («Low») ist.
$\overline{CE}$ Eingangssignal	Chipauswahlsignal (aktiv ≙ «Low»).
$\overline{M1}$ Eingangssignal	Signal für M1-Zyklus: Dieses Signal entspricht dem Signal $\overline{M1}$ des Prozessors.
$\overline{RD}$ Eingangssignal	Lesesignal: Dieses Signal entspricht dem $\overline{RD}$-Signal des Prozessors (aktiv ≙ «Low»).
$\overline{IORQ}$ Eingangssignal	Eingabe-/Ausgabe-Anforderung vom Prozessor (aktiv = «Low«).
IEI Eingangssignal	Interruptfreigabesignal: Dieses Signal dient zur Bildung von Interruptprioritätsschaltungen.
IEO 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».
$\overline{INT}$ Ausgangssignal	Interruptanforderungssignal (aktiv ≙ «Low»).
Φ Eingangssignal	Systemtakt.

$\overline{\text{RESET}}$	Rücksetzen der Empfänger- und Senderlogik. Aus T × DA und T × DB entsteht das Grundsignal. Steuerregister sind gelöscht. Alle Interrupts sind gesperrt (aktiv ≙ «Low»).
Signale der Kanäle A und B:	
R × DA, R × DB	Empfangsdaten (aktiv ≙ «High»).
Eingangssignale	
T × DA, T × DB	Sendedaten (aktiv ≙ «High»).
Ausgangssignale	
$\overline{\text{R} \times \text{CA}}$, $\overline{\text{R} \times \text{CB}}$*)	Empfängertakt (aktiv ≙ «Low»).
Eingangssignale	
$\overline{\text{T} \times \text{CA}}$, $\overline{\text{T} \times \text{CB}}$*)	Sendertakt (aktiv ≙ «Low»).
Eingangssignale	
$\overline{\text{CTSA}}$, $\overline{\text{CTSB}}$	Steuersignal für Datensender. Im Arbeitsmodus AUTOENABLE werden bei $\overline{\text{CTS}}$ = «High» Daten nicht gesendet. In anderen Arbeitsmode können diese Anschlüsse als Eingänge anderweitig verwendet werden. Die beiden Leitungen haben *Schmitt*-Trigger-Eingänge (aktiv ≙ «Low»).
Eingangssignale	
$\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$	Steuersignal für Datenempfang. Das Signal $\overline{\text{DCD}}$ arbeitet analog dem Signal $\overline{\text{CTS}}$, jedoch wird durch $\overline{\text{DCD}}$ der Empfänger des zugehörigen Kanals gesperrt (aktiv ≙ «Low»).
Eingangssignale	
$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$	Programmierbares Quittungssignal. Sobald das RTS-Bit eines Kanals gesetzt ist, geht der zugehörige $\overline{\text{RTS}}$-Ausgang in den «Low»-Zustand. Wird das RTS-Bit im Asynchronmodus rückgesetzt, geht der zugehörige $\overline{\text{RTS}}$-Ausgang in den «High»-Zustand, sobald das Senderegister leer ist. Im Synchronmodus folgt der Anschluß dem RTS-Bit (aktiv ≙ «Low»).
Ausgangssignale	

$\overline{\text{DTRA}}$, $\overline{\text{DTRB}}$*) Ausgangssignale — Programmierbares Quittungssignal. Der Ausgang folgt dem Wert des DTR-Bit (aktiv ≙ «Low»).

$\overline{\text{SYNCA}}$, $\overline{\text{SYNCB}}$ — Synchronisationssignal (aktiv ≙ «Low»). Bei interner Synchronisation: «Ausgang». Das Signal ist in dem Takt aktiv, in dem ein Synchronisationszeichen erkannt wird.
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 *U 856 D*

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 (WR_0 bis WR_7),
drei Register für Statusinformationen (RR_0 bis 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.

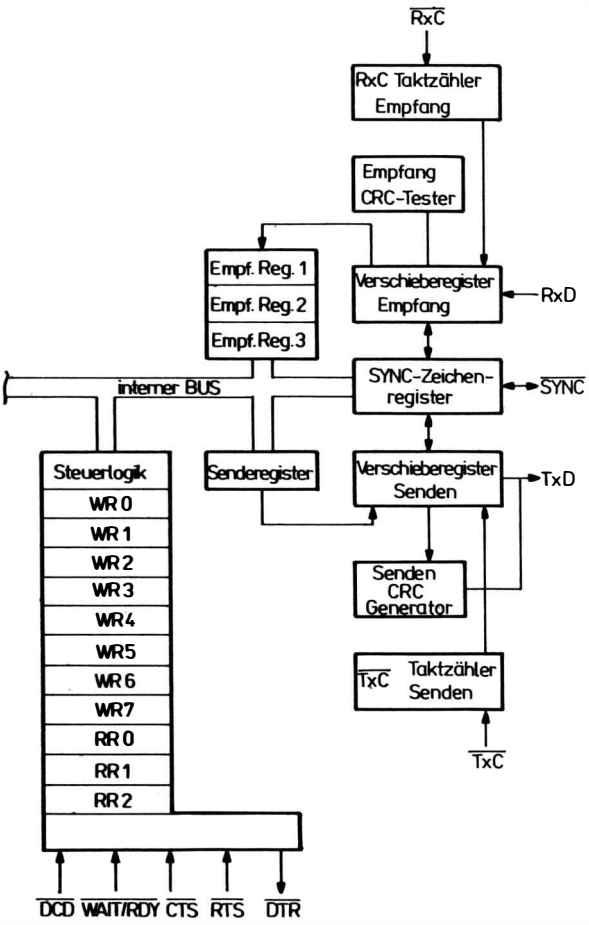


Bild 2.9 Logische Struktur eines Kanals des Bausteins *U 856 D*

Empfang von seriellen Daten:
Die Daten kommen über den Eingang R × 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/$\overline{D}$-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 ... D_n	Parity	STOP
		5, 6, 7, 8 Bit/Zeichen	Programmierbar ja/nein gerade/ ungerade	1, 1½ oder 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 $^1/_2$-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 (Nullinie 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_7. Bei Erkennen des SYNC-Zeichens wird der SYNC-PIN aktiv (bis zum Ende des Taktzyklus, in dem das SYNC-Zeichen erkannt wurde).

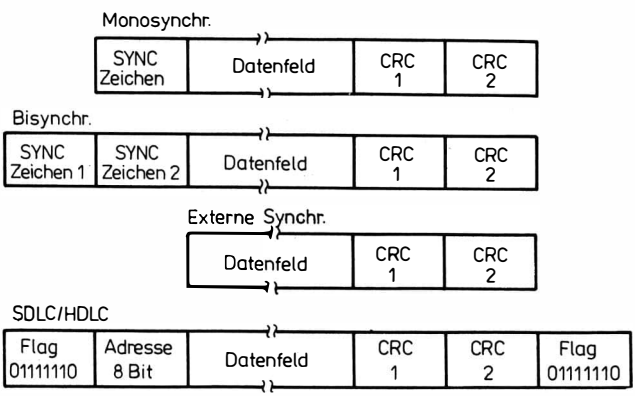


Bild 2.10 Datenformate im Synchronmodus des Bausteins *U 856 D*

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 $\overline{\text{SYNC}}$-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 $\overline{\text{SYNC}}$ ($\overline{\text{SYNC}}$-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 $\overline{\text{SYNC}}$-PIN von außen «Low» gesetzt ist.
- Abbruch des Empfangs erfolgt, wenn:
- RESET erscheint;
- Bit RECEIVER ENABLE rückgesetzt wird;
- im AUTOENABLE-Zustand $\overline{\text{DCD}}$ «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 Bit TRANSMIT BUFFER EMPTY nicht gesetzt	CRC-Wort wird gesendet
Bit CRC/SYNC gesetzt und Bit TRANSMIT BUFFER 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_0 bis WR_7 und der Statusregister RR_0 bis RR_2 (s. Bild 2.11)

WR_0:

$D_0 \ldots D_2$ Nr. des nächsten zu ladenden oder zu lesenden Registers.

$D_3 \ldots D_5$ RESET-Kommandos

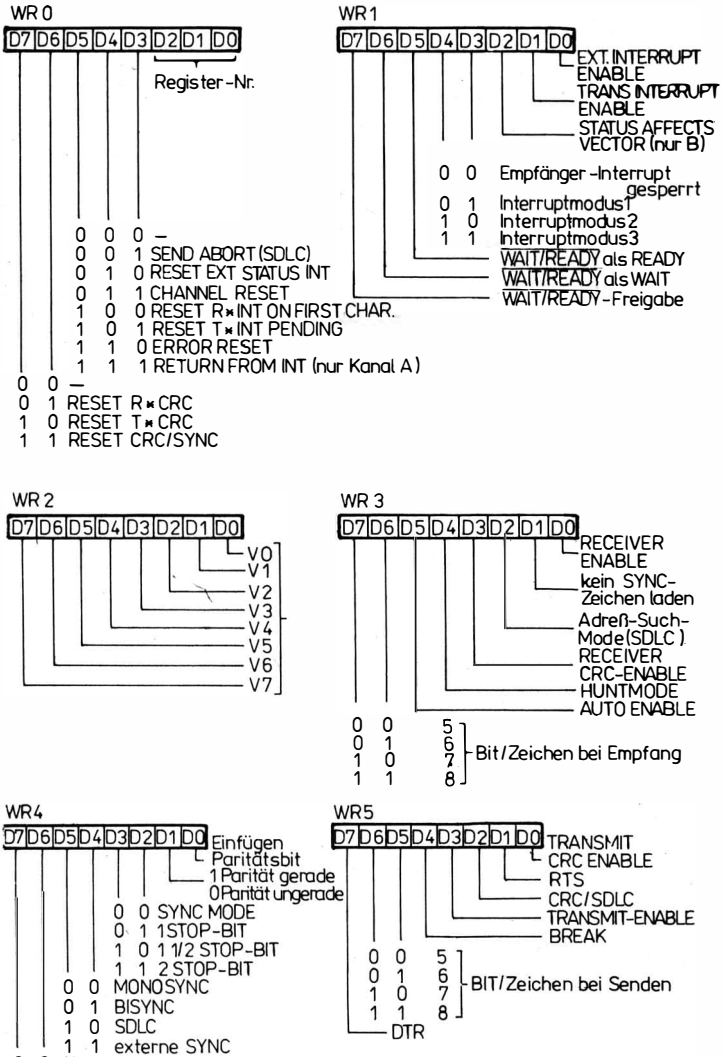
SEND ABORT (nur SDLC). Es wird eine Folge von 8 bis 13 «1» gesendet.

RESET EXT STATUS INT löscht die im RR_0 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_1.

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

$D_6 \ldots D_7$ setzt die CRC-Logik zurück.


WR 0
D7 D6 D5 D4 D3 D2 D1 D0
Register-Nr.
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 Rx INT ON FIRST CHAR.
1 0 1 RESET Tx INT PENDING
1 1 0 ERROR RESET
1 1 1 RETURN FROM INT (nur Kanal A)
0 0 —
0 1 RESET Rx CRC
1 0 RESET Tx CRC
1 1 RESET CRC/SYNC
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 Interruptmodus1
1 0 Interruptmodus2
1 1 Interruptmodus3
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 MONO SYNC
0 1 BISYNC
1 0 SDLC
1 1 externe SYNC
0 0 X1
0 1 X16
1 0 X32
1 1 X64
WR5
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

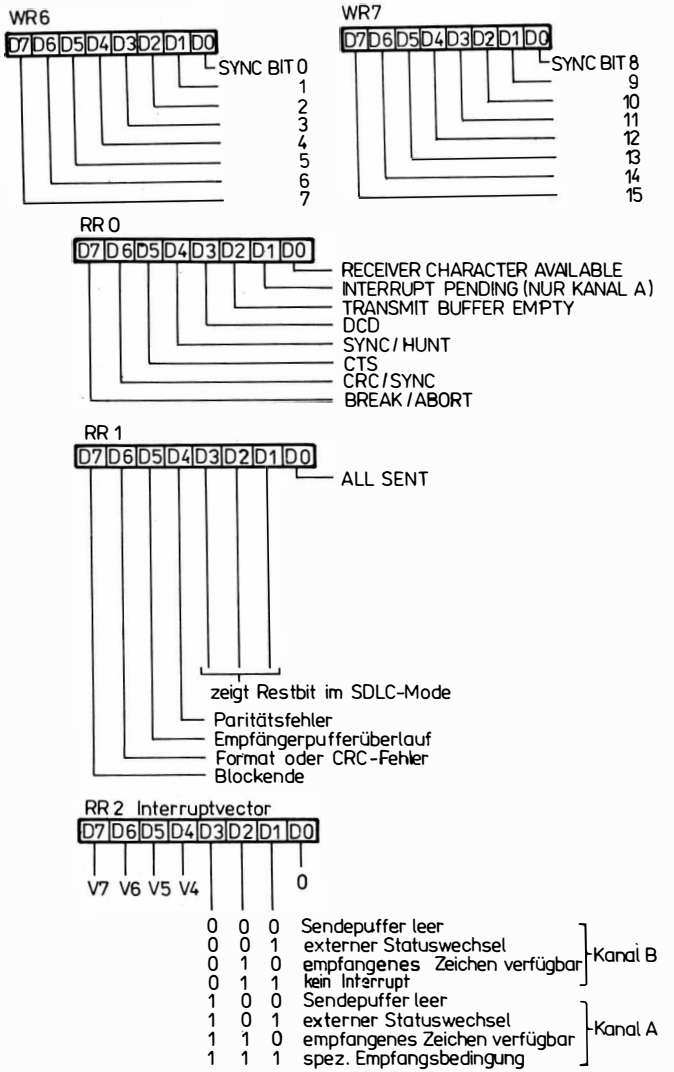


Bild 2.11 Bedeutung der Bitstellen der Steuerregister WR_0 bis WR_7 und der Statusregister RR_0 bis RR_2 im Baustein *U 856 D*

WR_1:

D_0 EXT INTERRUPT ENABLE. Gestattet Interrupt bei Wechsel von $\overline{DCD}$, $\overline{CTS}$, $\overline{SYNC}$,-bei BREAK, bei Sendebeginn von CRC oder SYNC-Zeichen.

D_1 TRANS INTERRUPT ENABLE. Erlaubt INT, wenn der Sendepuffer leer ist.

D_2 STATUS AFFECTS VECTOR (nur Kanal B). Ist dieser Modus ausgewählt, so wird der Interruptvektor in WR_2 wie folgt modifiziert:

V_3	V_2	V_1		
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*)	

$D_3 \ldots D_4$ Durch D_3 und D_4 werden spezielle Interruptbildungsmöglichkeiten für Empfänger eingestellt.

D_5 $D_5 = 1$ Der $\overline{WAIT}$-/$\overline{READY}$-Ausgang wird aktiv, wenn der Empfänger leer ist.

$D_5 = 0$ Der $\overline{WAIT}$-/$\overline{READY}$-Ausgang wird aktiv, wenn der Sendepuffer voll ist.

D_6 $D_6 = 0$ $\overline{WAIT}$-/$\overline{READY}$-Ausgang arbeitet als WAIT (Ausgang).

$D_6 = 1$ $\overline{WAIT}$-/$\overline{READY}$-Ausgang arbeitet als READY (Ausgang).

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

READY-Funktion: Das SIO ist zur Arbeit bereit.

D_7 Freigabe des WAIT-/READY-Ausgangs.

WR_2: Dieses Register enthält den Interruptvektor. Er existiert nur im Kanal B. Bei STATUS AFFECTS VECTOR werden die Bit-Stellen V_1 bis V_3 entsprechend der Interrupterzeugung eingetragen.

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

WR_3:

D_0 Erlaubt Empfangsfunktion.

D_1 Bei $D_1 = 1$ werden die SYNC-Zeichen nicht in die Empfangspufferregister geladen. CRC erfolgt normal.

D_2 $D_2 = 1$ verhindert im SDLC-Modus Interrupt, wenn keine Übereinstimmung zwischen empfangener und programmierter Adresse oder wenn keine Globaladresse (11111111) vorliegt.

D_3 Mit $D_3 = 1$ beginnt die CRC-Konrolle beim Start des letzten Zeichens vom Empfangsregister zum Puffer.

D_4 HUNTMODE. Die Übertragung wird abgebrochen, bis ein neues SYNC-Zeichen kommt.

D_5 AUTOENABLE. Bei $D_5 = 1$ arbeiten die DCD- und CTS-Eingänge als Empfangs- und Sendesteuerung.
Bei $D_5 = 0$ wirken DCD und CTS nur auf ihre Bit im Statusregister.

D_6/D_7 Bestimmt empfangene Bit/Zeichen.

WR_4:

D_0 $D_0 = 1$, SIO arbeitet mit Paritäts-Bit,

D_1 $D_1 = 0$, Parität gerade,
$D_1 = 1$, Parität ungerade.

D_2/D_3 Bestimmt die Anzahl der Stop-Bit bei Senden.

D_4/D_5 Bestimmt die Art des Synchronmodus.

D_6/D_7 Bestimmung des Multiplikationsfaktors zwischen Takt und Übertragungsrate.

WR_5:

D_0 TRANSMIT CRC-ENABLE. CRC-Freigabe bei Senden.

D_1 $D_1 = 1$, $\overline{RTS}$ geht auf 0.
$D_1 = 0$, $\overline{RTS}$ geht auf 1, wenn der Sender leer ist.

D_2 CRC/SDLC
$D_2 = 0$, CRC-Kontrolle mit SDLC-Polynom
$x^{16} + x^{12} + x^5 + 1$.
$D_2 = 1$, CRC-Kontrolle mit Polynom
$x^{16} + x^{15} + x^2 + 1$.

D_3 TRANSMIT ENABLE.
Sendeerlaubnis. Bei Rücksetzen wird das letzte Zeichen noch voll ausgegeben.

D_4 BREAK. Bewirkt Aussenden von «0» auf T × D.

D_5/D_6 Bestimmt auszugebende Bit/Zeichen.
Bei weniger als 5 Bit/Zeichen gilt:

1	1	1	1	0	0	0	D	sendet 1 Bit
1	1	1	0	0	0	D	D	sendet 2 Bit
1	1	0	0	0	D	D	D	sendet 3 Bit
1	0	0	0	D	D	D	D	sendet 4 Bit
0	0	0	D	D	D	D	D	sendet 5 Bit

D_7 DTR
$D_7 = 1$ bewirkt $\overline{DTR} = 0$ (aktiv).

WR_6 SYNC-Zeichen bei Monosynchronisation, die ersten 8 Bit bei Bisynchronisation.

WR_7 2. Zeichen bei Bisynchronisation oder das Synchronzeichen bei Monosynchronisation. Bei SDLC muß $WR_7 = 01111110$ sein.

RR_0:

D_0 RECEIVE CHARACTER AVAILABLE; gesetzt, wenn mindestens 1 Zeichen im Puffer.

D_1 INTERRUPT PENDING; wird bei jeder INT-Anmeldung gesetzt (nur Kanal A).

D_2 TRANSMIT BUFFER EMPTY; gesetzt, wenn der Sendepuffer leer ist (außer bei CRC).

D_3 DCD; zeigt Zustand $\overline{DCD}$.

D_4 SYNC/HUNT
$D_4 = 0$, Zeichensynchronisation erreicht,
$D_4 = 1$, keine Synchronisation.

D_5 CTS; zeigt Zustand $\overline{CTS}$.

D_6 BREAK/ABORT; wird gesetzt, wenn mit dem Senden von CRC-Worten begonnen wird, nachdem der Sendepuffer leer geworden ist.

D_7 CRC/SYNC
$D_7 = 1$ bei Erkennen von BREAK (bei ASYNC).
Im SDLC-Format bei Erkennen einer Abbruchfolge (7 oder mehr Einsen).

RR_1:

D_0 ALL SENT wird gesetzt, wenn alle Zeichen im Asynchronmode gesendet sind.

$D_1/D_2/D_3$ Falls bei Empfang im SDLC-Modus die Anzahl der Bit/Zeichen nicht aufgehen, zeigt $D_1/D_2/D_3$ die restlichen Bit an.

D_4	Paritätsfehler,
D_5	Empfängerüberlauf,
D_6	Formatfehler im Asynchronmode, CRC-Fehler im Synchronmode.
D_7	Blockende bei SDLC mit gültigem CRC erkannt.
RR_2:	Enthält Interruptvektor. Die Bit sind entsprechend 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.

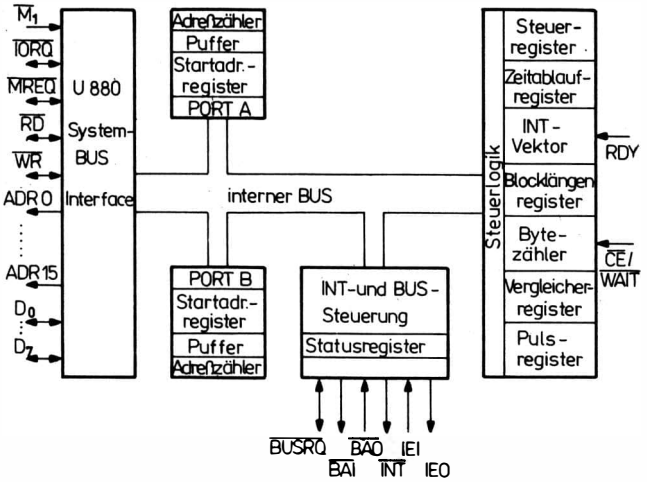


Bild 2.12 Grundstruktur des Bausteins *U 880* – DMA

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 gelesenem 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 $\overline{INT}$ ein Impuls ausgegeben. Die CPU reagiert nicht, da durch $\overline{BUSRQ}$ der DMA aktiviert ist.
Statusregister
Dieses Register enthält die gesammelte Statusinformation.

2.5.2. Beschreibung der Anschlüsse des DMA-Bausteins

$A_0 \ldots A_{15}$	Adreßbus (Tri-state-Ausgang, aktiv «High»).
$D_0 \ldots D_7$	Datenbus (Tri-state, bidirectional, aktiv «High»).
$\overline{M1}$	Signal für M1-Zyklus (Eingabesignal, aktiv «Low»).
$\overline{IORQ}$	Eingabe-/Ausgabe-Anforderung (Ein- bzw. Ausgabesignal aktiv «Low»).
$\overline{MREQ}$	Speicheranforderung (Ein- bzw. Ausgang, aktiv «Low»).
$\overline{RD}$	Lesen (Ein- bzw. Ausgang, aktiv «Low»).
$\overline{WR}$	Schreiben (Ein- bzw. Ausgang, aktiv «Low»).
$\overline{CE/WAIT}$	(Eingang, aktiv «Low»); dient als Bausteinfreigabesignal. Es kann, wenn $\overline{BAI}$ = «Low» ist, auch als «WAIT» verwendet werden. Die Auswahl erfolgt durch Programm.
$\overline{BUSRQ}$	Busanforderung (Ein-/Ausgang, aktiv «Low»); Signal zur Anforderung einer DMA-Funktion.
$\overline{BAI}$	Busanforderungsbestätigung (Eingang, aktiv «Low»); zeigt an, daß die Buskontrolle an den DMA-Baustein übergeben wurde.
$\overline{BAO}$	Weitergabe $\overline{BAI}$ (Ausgang, aktiv «Low»); gibt $\overline{BAI}$-Signal an den nächsten Baustein weiter.
$\overline{INT}$	Interruptanforderung (Ausgang, aktiv «Low»).
IEI	Interruptfreigabe (Eingang, aktiv «High»).
IEO	Weitergabe Interruptfreigabe (Ausgang, aktiv «High»).
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 Schreibtor und ein Speicherschreibzyklus. Bei «Suchen» kommt es zum Vergleich mit dem Byte im Vergleicherregister, 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 = «0» zu einer Busfreigabe kommt.

Byteweise Übertragung
Die Busfreigabe ist unabhängig von RDY nach jedem Lesezyklus bei «Suchen» und nach jedem Schreibzyklus bei «Übertragen». Nachdem $\overline{\text{BUSRQ}}$ inaktiv ist, kann ein erneutes $\overline{\text{BUSRQ}}$ erst kommen, wenn vorher $\overline{\text{BAI}}$ 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	≠0	≠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 → A; 1A → B
D_1 D_0	0 0 – ; 0 1 Transfer; 1 0 Suche; 1 1 Suche + Transfer

Steuerbyte 1B

D_7 0
D_6 Zeitablaufbyte folgt
D_5 Toradresse ist fest
D_4 0 Toradresse wird decrementiert
1 Toradresse wird incrementiert
D_3 0 } B = Speicher; 0 } A = Speicher; 1 } B = E/A;
D_2 0 } 1 } 1 } A = E/A
D_1 0
D_0 0

Zeitablaufbyte

D_7 0 = $\overline{WR}$-Ende 1/2 Zyklus früher
D_6 0 = $\overline{RD}$-Ende 1/2 Zyklus früher
D_5 frei
D_4 frei
D_3 0 = $\overline{MREQ}$-Ende 1/2 Zyklus früher
D_2 0 = $\overline{IORQ}$-Ende 1/2 Zyklus früher
D_1 0 } Zykluslänge 0 } Zykluslänge 1 } Zykluslänge
D_0 0 } = 4; 1 } = 3; 0 } = 2

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

Steuerbyte 2A

D_7 1
D_6 DMA-Freigabe
D_5 Interruptfreigabe
D_4 Vergleichsbyte folgt
D_3 Maskenbyte folgt
D_2 Stop bei Vergleich
D_1 0
D_0 0

Steuerbyte 2B

D_7 1
D_6 } Be- 0 } byte- 0 } block- 1 } burst 1 } –
D_5 } triebsart; 0 } weise; 1 } weise; 0 } 1 }
D_4 Interruptsteuerbyte folgt

D_3 Adresse B, «High»-Byte folgt
D_2 Adresse B, «Low»-Byte folgt
D_1 0
D_0 1

Interruptsteuerbyte
D_7 frei
D_6 Interrupt vor BUSRQ
D_5 Status verändert Interruptvektor
D_4 Interruptvektor folgt
D_3 Pulszählerbyte folgt
D_2 Pulse werden erzeugt
D_1 Interrupt bei Blockende
D_0 Interrupt bei Gleichheit

Interruptvektor
D_7 V_7
D_6 V_6
D_5 V_5
D_4 V_4
D_3 V_3

D_2 0	bei 0	bei 1	bei 1	bei
	Interrupt	Interrupt	Interrupt	Interrupt
	auf 1	auf 0	auf 1	auf
D_1 0	RDY;	Gleich-	Block-	Gleichheit
		heit;	ende;	oder
				Blockende

D_0 0

Steuerregister 2C
D_7 1
D_6 frei
D_5 0 Stop bei Blockende; 1 wiederholen bei Blockende
D_4 0 $\overline{CE}$ nur; 1 $\overline{CE}/\overline{WAIT}$
D_3 0 RDY aktiv «Low»; 1 RDY aktiv «High»
D_2 frei
D_1 frei
D_0 0

Steuerregister 2D

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

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_7 frei
D_6 frei
D_5 Blockende-Bit, 0 = Blockende
D_4 Vergleichs-Bit, 0 = Vergleich
D_3 0 INT-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_2 frei
D_1 0 RDY = aktiv
D_0 0 DMA-Zyklus nicht erschienen
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_7
D_6 Adresse B HWT
D_5 Adresse B NWT
D_4 Adresse A HWT
D_3 Adresse A NWT
D_2 Bytezähler HWT
D_1 Bytezähler NWT
D_0 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 *U 880*, 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-

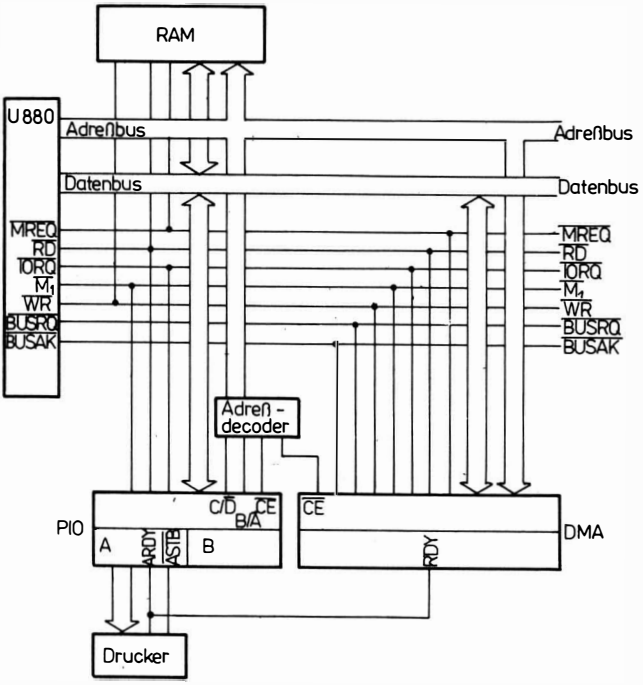


Bild 2.13 Anschluß eines DMA-Bausteins an den Systembus *U 880*

itialisiert. 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	Bausteinauswahl		Adresse 30H

Zur Initialisierung der Übertragung vom Speicher zum Drucker müssen folgende Bit-Muster von der CPU *U880* ausgegeben werden:

	Befehl	Wort auf Datenbus
Ausgabe Steuerwort zur Festlegung der Betriebsart «Ausgabe» im PIO Kanal A	OUT 40 H	00001111
Interruptsperre für PIO Kanal A	OUT 40 H	00000011
Steuerbyte 1A für DMA Übertragung B → A Da Kanal B mit einer festen Adresse versehen wird, muß er erst als Eingabekanal definiert werden.	OUT 30 H	00001000
Steuerbyte 1B für DMA Kanal B Kanal B zur Peripheriesteuerung mit fester Adresse	OUT 30 H	00101000
Steuerbyte 2B Byteweise Übertragung Festlegen Adresse für Kanal B	OUT 30 H	10000101
Adresse PIO Datenkanal A zum DMA Kanal B	OUT 30 H	PIO-Adresse
Steuerbyte 2D Laden Toradresse B	OUT 30 H	11001111
Steuerbyte 1A für DMA 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 IB für DMA-Kanal A Kanal steuert Speicher, Adresse wird incrementiert, kein Zeitsteuerbyte	OUT 30 H	00010100

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

2.6. Serieller Ein-/Ausgabeschaltkreis *8251*

2.6.1. Funktionen des *8251*

Ein neben dem SIO häufig verwendeter Baus
von paralleler in serielle Information und umge
kreis *8251* (USART = Universal Synchron A
Transmitter). Er hat nicht so viele Möglichkeite
stein, ist aber für viele Anwendungsfälle ausre
den Anschluß peripherer Geräte über serielle
dem *8251* lassen sich folgende Betriebsarten re

Asynchron
- 5 bis 8 Zeichen
- Taktuntersetzung ×1, ×16 oder ×64
- Erzeugung BREAK
- 1, 1½ oder 2 Stopbits

Synchron
- 5 bis 8 Zeichen
- 1 oder 2 Synchronisationszeichen
- externe Synchronisation

(jedoch kein SDLC und keine CRC-Prüfung)

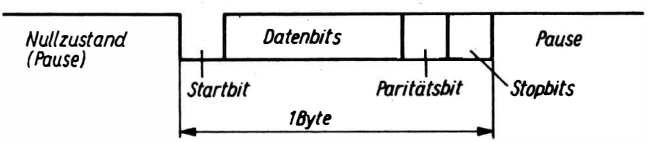
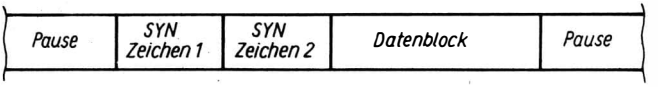


Bild 2.14 Asynchrones Datenprotokoll



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

Bild 2.15 Synchrones Datenprotokoll

2.6.2. Anschlußbelegung

Beschreibung der Anschlüsse

D_0–D_7	Datenbus (Bidirektional).
C/$\overline{D}$	Unterscheidung Steuerwort – Datenwort 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×RDY	Sender ist zur Aufnahme eines Bytes bereit, wird beim Laden eines Zeichens zurückgesetzt (aktiv «High»).
T×E	kein Zeichen zum Senden im Baustein (Sender leer), wird beim Empfang eines Zeichens vom Prozessor zurückgesetzt (aktiv «High»).
R×RDY	Baustein hat ein Zeichen empfangen, wird beim Lesen des Zeichens zurückgesetzt (aktiv «High»).
$\overline{DTR}$	(Data Terminal Ready). $\overline{DTR}$ ist Ausgangssignal und läßt sich durch Programm einstellen (aktiv «Low»).

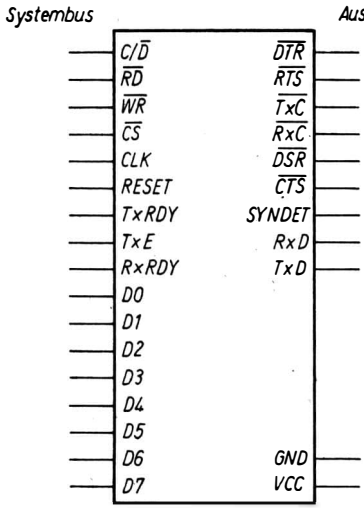


Bild 2.16
Anschlußbelegung des *8251*

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

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

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

T×C Sendetakt.

R×C 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*

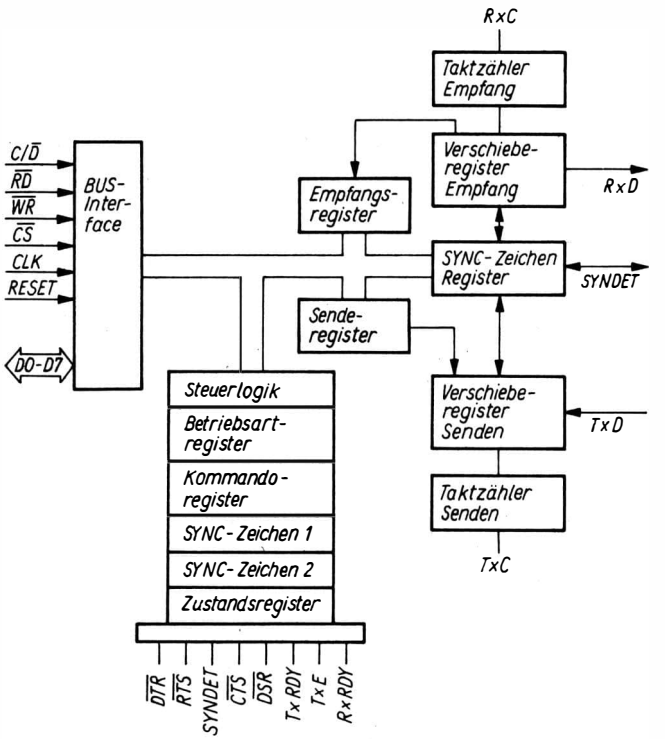


Bild 2.17 Logische Struktur des *8251*

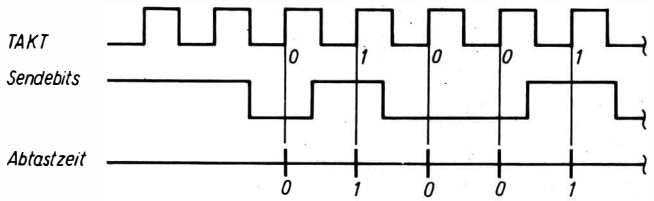


Bild 2.18 Senden und Empfangen von Bits im *8251*

2.6.4. Arbeitsweise

Das Senden eines Bits erfolgt mit der H/L-Flanke von T×D. Mit der L/H-Flanke von R×C werden Empfangsleitungen getestet.
Für Kurzschlußbetrieb (Sender mit Empfänger verbunden und T×C mit R×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 ×16 oder ×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×C oder R×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 SYNDET eingeschaltet. Bei externer Synchronisation erfolgt die Synchronisation mit SYNDET. Ein Eingangsimpuls an SYNDET 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.

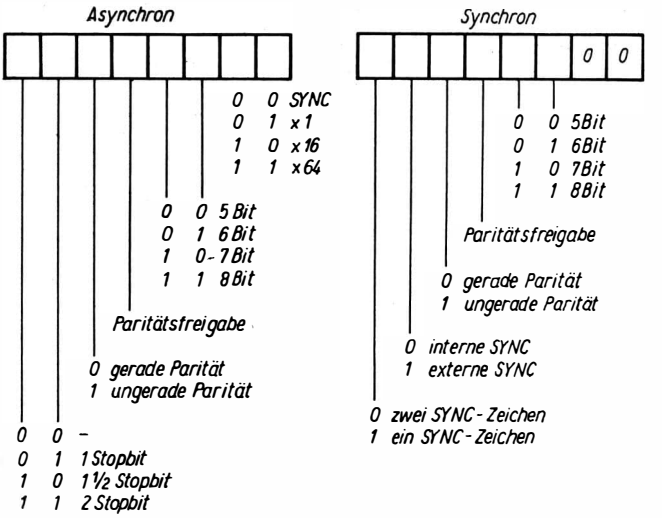


Bild 2.19 Einstellung der Betriebsarten des *8251*

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

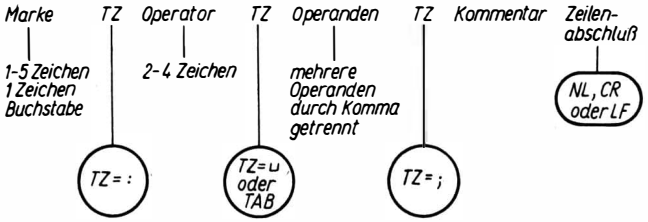


Bild 3.1 Aufbau einer Quellzeile

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 (Beispiele)
1. Zahlen:	Dezimal	14
	Oktal	140 oder 14Q
	Hexadezimal	(die erste Ziffer darf kein Buchstabe A–F sein) 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	A, B, C, usw.
	Doppelregister	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 P1. MA

Programmname — Zieladresse im Programm P1

- 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.

EJEC Der Drucker für die Programmliste macht einen Formularvorschub.

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) → (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)     ⎬   Makrokörper
       DAA          ⎪
       LD (Z1), A   ⎭
       ENDM;             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[1)]
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[1)]
Rechenoperationen	
OP A, r	OP r
OP A, n	OP n
CP s	CMP s
Sprünge	
JP ADR	JMP ADR
JP NZ, ADR	JPNZ ADR
JP Z, ADR	JPZ ADR
usw.	usw.
CALL NZ, ADR	CANZ ADR
CALL Z, ADR	CAZ ADR
usw.	usw.
RET NZ	RNZ
RET Z	RZ
usw.	usw.
Relative Sprünge	
JR C, MARKE	JRC MARKE − #
JR C, $ + 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 nn	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

[1)] 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.:

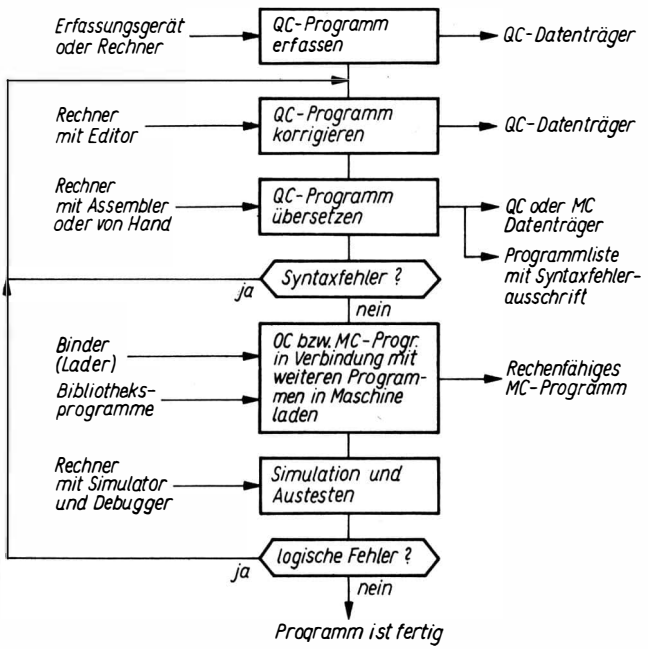


Bild 4.1 Übersicht der Schritte für die Programmaufbereitung und der dazu notwendigen Geräte und Systemprogramme

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

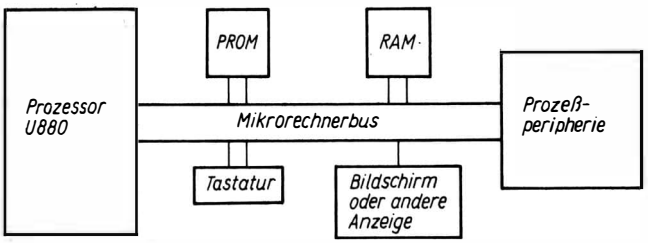


Bild 5.1 Basiskonfiguration eines Mikrorechners

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

Im folgenden Teil sollen einige Beispiele zum Aufbau der Minimalkonfiguration 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 Minimalkonfigurationen kann man dabei einige Signale, die nicht benötigt werden, weglassen (z. B. $\overline{\text{RFSH}}$, wenn ohne dynamische Speicher gearbeitet wird, $\overline{\text{BUSRQ}}$ und $\overline{\text{BUSAK}}$, 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 $\overline{\text{WAIT}}$-Leitung realisieren. (Bild 5.2) Die Schaltung TG soll als Eintaktgenerator arbeiten. Am Ausgang TG

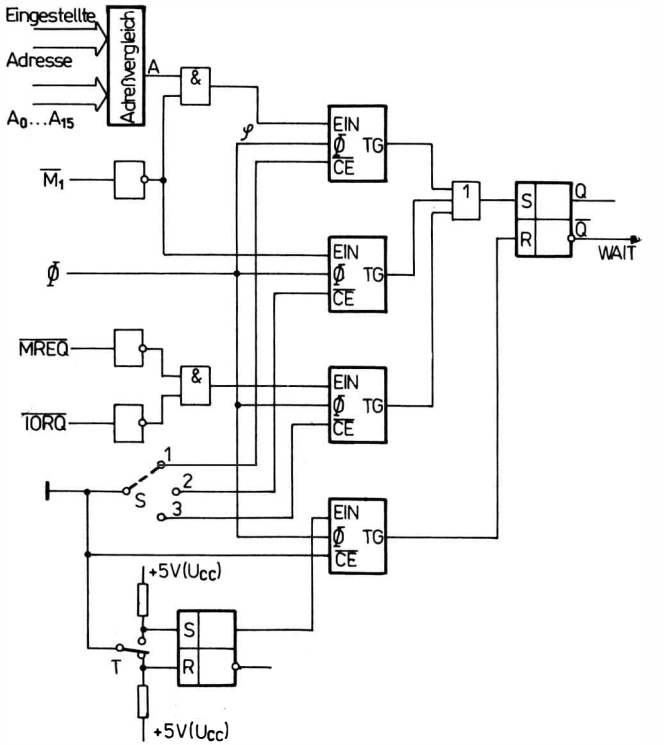


Bild 5.2 Erzeugung des WAIT-Signals zur Realisierung einer taktweisen und befehlsweisen Abarbeitung

wird, wenn $\overline{CE}$ Tiefpegel hat, der erste Takt Φ, 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

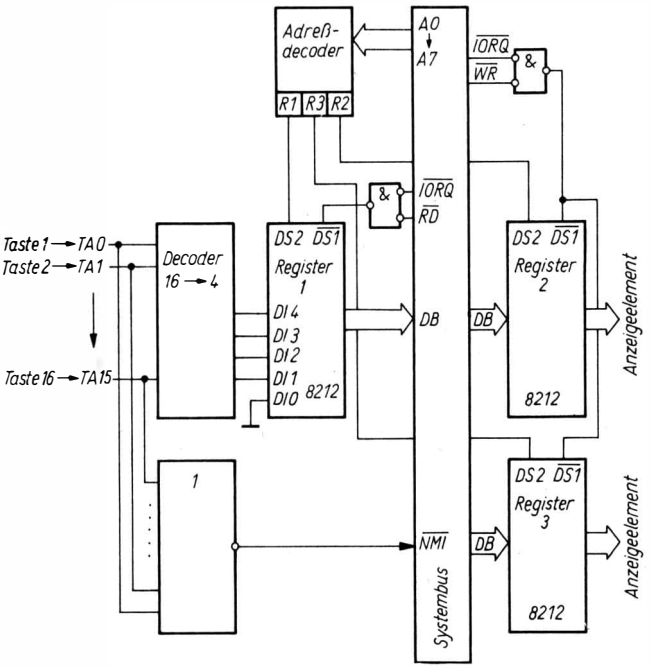


Bild 5.3 Logische Prinzipschaltbild zur Realisierung von Tastenfunktionen am Mikrorechner

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 $\overline{\text{NMI}}$, 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
LD D, 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:OUT 21H     ; 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

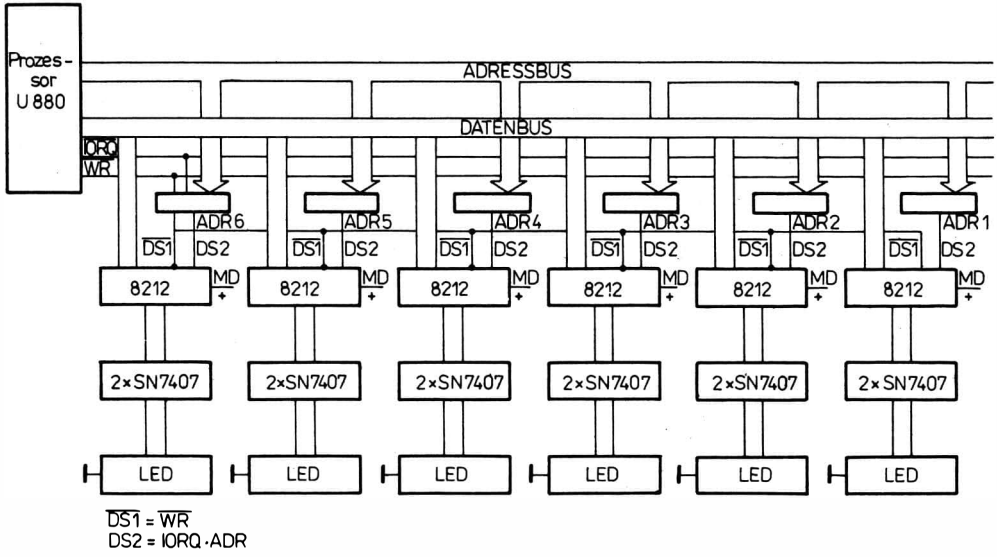


Bild 5.4 Anschluß von 6 LED-Anzeigen an den Systembus eines Mikrorechners

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

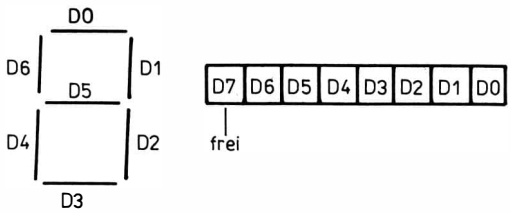
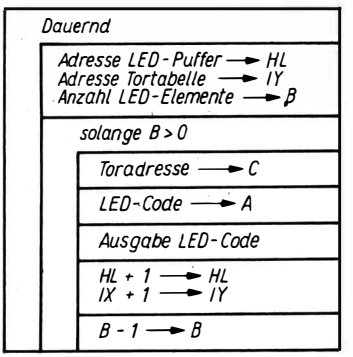


Bild 5.5 Zuordnung der Bit-Stellen eines Bytes zu den Segmenten einer LED-Anzeige

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. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
   LOC    OBJ.CODE   STMT     SOURCE PROGRAM
=========================================================================84/08/15===001==========
                  0001          PN   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
   0000  FD E5    0007  LEDA:  PUSH IY
   0002  E5       0008         PUSH HL
   0003  C5       0009         PUSH BC
   0004  FD 4E 00  0010  ZYK:  LD   C,(IY)
   0007  7E        0011         LD   A,(HL)
   0008  ED 49     0012         OUT  C
   000A  23        0013         INC  HL
   000B  FD 23     0014         INC  IY
   000D  10 F5     0015         DJNZ ZYK-#
   000F  C1        0016         POP  BC
   0010  E1        0017         POP  HL
   0011  FD E1     0018         POP  IY
   0013  18 EB     0019         JR   LEDA-#
   0015            0020         END
                 PROGRAM CONTAINS  0000 ERROR(S)
```

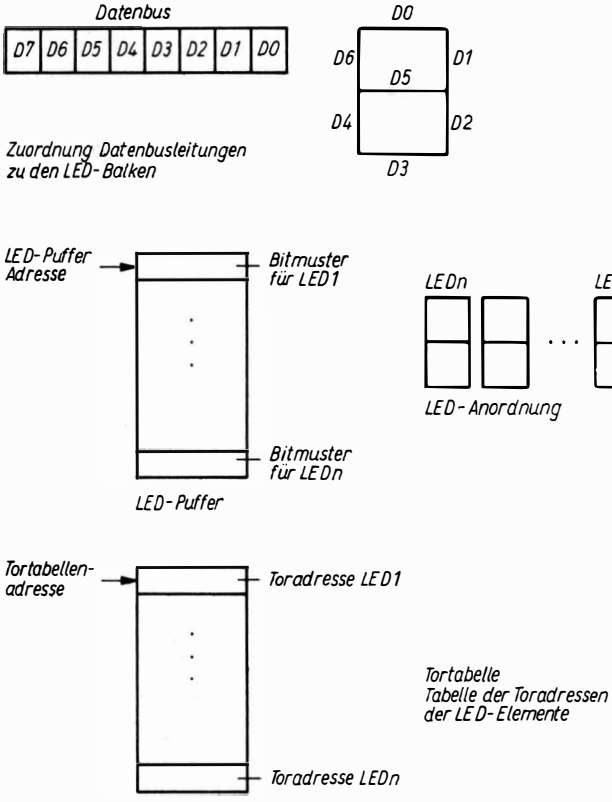


Bild 5.6 Schema zur Programmierung einer LED-Anzeige

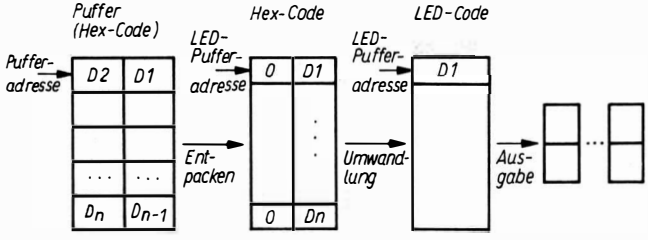
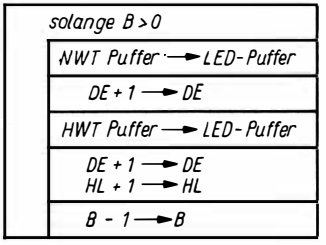


Bild 5.7 Schritte zur hexadezimalen Anzeige eines Puffers auf LED-Elementen

Um den Inhalt eines Speicherpuffers auf LED-Elementen hexadezimal 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



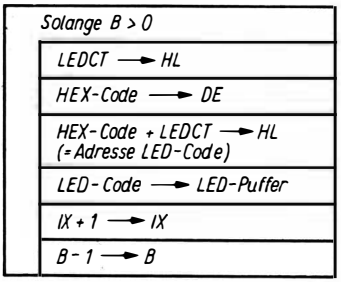
b – Assembler-Programm

```
; 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
        INC HL
        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 MEOS 1521 V4.2
  LOC    OBJ.CODE   STMT     SOURCE PROGRAM
=================================================================04/08/15===001==========
                  0001          PN   C6
                  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
  0000  21 00 00  0007  UHL:  LD   HL,LEDCT       ;TABELLE LED CODE
  0003  FD 5E 00  0008        LD   E,(IY)
  0006  16 00     0009        LD   D,0
  0008  19        0010        ADD  HL,DE
  0009  7E        0011        LD   A,(HL)
  000A  DD 77 00  0012        LD   (IX),A
  000D  FD 23     0013        INC  IY
  000F  10 EF     0014        DJNZ UHL-#
  0011  C9        0015        RET
                  0016  LEDCT: EQU  0             ;ADRESSE FUER LED-CODE TABELLE
                  0017  ;DIE ADRESSE IST ANZUGEBEN
  0012            0018        END
                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.

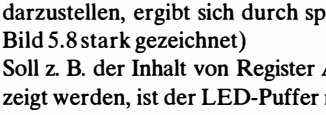
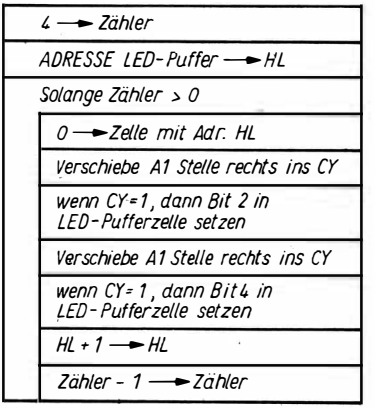


Bild 5.8
Auswahl von Balken einer 4stelligen LED-Anzeige zur Darstellung einer Dualzahl

Lösung:
a – Struktogramm



b – Assembler-Programm

```
                    ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
   LOC     OBJ. CODE    STMT      SOURCE PROGRAM
==========================================================================84/08/15==001==========
                     0001           FN   C7
                     0002
                     0003  ;LED-AUFBEREITUNG
                     0004  ;BITMUSTER VON REGISTER A WIRD AUF 4LED-ELEMENTEN ANGEZEIGT
                     0005  ;BITMUSTER IN REGISTER A
                     0006  ;ADRESSE LED-PUFFER IN HL
   0000  06 04       0007  DUALA: LD   B,4
   0002  36 00       0008  ZYK:   LD   (HL),0
   0004  1F          0009        RRA
   0005  30 02       0010        JRNC M1-#
   0007  CB D6       0011        SET  2,(HL)
   0009  1F          0012  M1:   RRA
   000A  30 02       0013        JRNC M2-#
   000C  CB E6       0014        SET  4,(HL)
   000E  23          0015  M2:   INC  HL
   000F  10 F1       0016        DJNZ ZYK-#
   0011  C9          0017        RET
   0012              0018        END
                    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

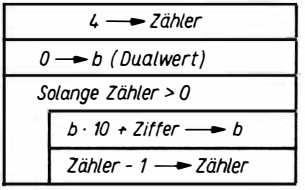
Die Darstellung der Zahlen im Mikrorechner wird durch den Anwendungsfall bestimmt. Sie wird durch das betreffende Programm definiert. Dabei treten oft rein duale Darstellungen in Verbindung mit BCD-Darstellungen oder Textdarstellungen (ASCII-Code) auf. Zur Realisierung einheitlicher Verarbeitung sind 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

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

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            PN   A1
                  0002  ;UMWANDLUNG BCD-DUAL
                  0003  ;UMWANDLUNG EINER INTEGER BCD-ZAHL (4 STELLEN UNSIGNIERT)
                  0004  ;IN EINE DUALZAHL (16 BIT)
                  0005
0000  E5          0006  DBI2:   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
                  0020  ;ALGORITHMUS
0014  21 00 00     0021          LD   HL,0               ;B=0
0017  06 04       0022          LD   B,4                ;ZAEHLER =4
0019  5D          0023  ZYK:    LD   E,L
001A  54          0024          LD   D,H
001B  29          0025          ADD  HL,HL              ;MULTIPLIKATION MIT 10
001C  29          0026          ADD  HL,HL
001D  19          0027          ADD  HL,DE
001E  29          0028          ADD  HL,HL
001F  D1          0029          POP  DE                 ;ZIFFER HOLEN
0020  5A          0030          LD   E,D
0021  16 00       0031          LD   D,0
0023  19          0032          ADD  HL,DE              ;ZIFFER ADDIEREN
0024  10 F3       0033          DJNZ ZYK-#
0026  D1          0034          POP  DE
0027  C9          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.

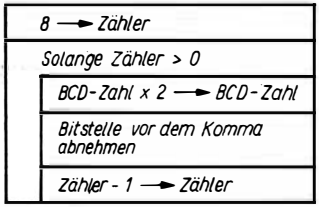
Eingangsparameter: BCD-Zahl in Register A

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:



Durch (BCD-Zahl · 2 = BCD-Zahl + 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.

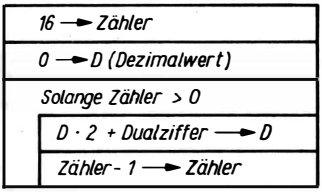
```
                    ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
  LOC    OBJ. CODE   STMT     SOURCE PROGRAM
=============================================================84/07/26===001==========
                    0001            PN   A2
                    0002  ;UMWANDLUNG EINER BCD-ZAHL /2/<1 (2 STELLEN UNSIGNIERT)
                    0003  ;IN EINE DUALZAHL (8 BIT)
                    0004
0000  06 08         0005  DBI1:  LD   B,8            ;ZAELER=8
0002  87            0006  MA1:   ADD  A              ;WERT*2
0003  27            0007        DAA
0004  CB 11         0008        RL   C
0006  10 FA         0009        DJNZ MA1-#          ;RUECKSPRUNG SOLANGE ZAELER>0
0008  79            0010        LD   A,C            ;RESULTAT NACH A
0009  C9            0011        RET
000A                0012        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 $b_{15}\, b_{14}\, b_{13} \ldots b_1\, b_0$ in der Form

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

so erkennt man folgenden Algorithmus:

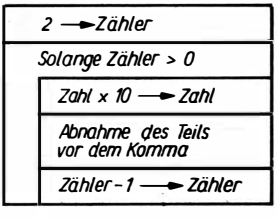


Die Dualziffern müssen mit der höchstwertigen beginnen. Durch Dualwert + Dualwert (HL + HL) kommen die Dualziffern der Reihe nach in das CY-Flag, das nur noch zu 2D addiert zu werden braucht.

```
                ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
  LOC    OBJ.CODE   STMT     SOURCE PROGRAM
=================================================================04/07/26===001=========
                  0001          FN   A3
                  0002  ;DUAL-BCD
                  0003  ;UMWANDLUNG EINER DUALZAHL (INTEGER 16BIT UNSIGNIERT)
                  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         ADC  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-$
  0011  EB        0022         EX   DE,HL
  0012  79        0023         LD   A,C
  0013  C9        0024         RET
  0014            0025         END
                FROGRAM 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          PN   A4
                    0002  ;UMWANDLUNG EINER DUALZAHL ECHT GEBROCHEN 8 BIT
                    0003  ;IN EINE DEZIMALZAHL 2 STELLEN
                    0004  ;DEZIMALZAHL UND DUALZAHL IN A
                    0005
0000  57           0006  UDBE1:  LD   D,A              ;DUALZAHL IN D
0001  67           0007          LD   H,A              ;DUALZAHL IN H
0002  2E 00        0008          LD   L,0
0004  1E 00        0009          LD   E,0
0006  AF           0010          XOR  A
0007  06 02        0011          LD   B,2              ;ZAEHLER=2
0009  29           0012  ZYK:    ADD  HL,HL            ;MULTIPLIKATION MIT 10
000A  8F           0013          ADC  A
000B  29           0014          ADD  HL,HL
000C  8F           0015          ADC  A
000D  19           0016          ADD  HL,DE
000E  CE 00        0017          ADC  0
0010  29           0018          ADD  HL,HL
0011  8F           0019          ADC  A
0012  05           0020          DEC  B
0013  C8           0021          RZ
0014  17           0022          RLA                   ;1 ZIFFER IN VORDERE 4 STELLEN VON A
0015  18 F2        0023          JR   ZYK-$
0017               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          PN   A5
                    0002  ;UMWANDLUNG DUAL NACH BCD
                    0003  ;16 BIT DUALZAHL UNSIGNIERT ERGIBT 4-STELLIGE BCD-ZAHL(ECHTGEBROCHEN)
                    0004  ;DUALZAHL IN HL
                    0005  ;BCD-ZAHL IN HL
                    0006
0000  06 04        0007  BDE2:   LD   B,4              ;FUER 4 BCD-STELLEN
0002  AF           0008          XOR  A
0003  4F           0009          LD   C,A              ;ZWISCHENSPEICHER VON 2BCD-STELLEN
0004  5D           0010  ZYK:    LD   E,L              ;MULTIPLIKATION MIT 10
0005  54           0011          LD   D,H
0006  29           0012          ADD  HL,HL
0007  17           0013          RLA                   ;REGISTER A UND C UEBERNEHMEN
                    0014                                ;DIE DEZIMALSTELLEN
0008  CB 11        0015          RL   C
000A  19           0016          ADD  HL,DE
000B  17           0017          RLA
000C  CB 11        0018          RL   C
000E  19           0019          ADD  HL,DE
000F  CE 00        0020          ADC  0
0011  29           0021          ADD  HL,HL
0012  17           0022          RLA
0013  CB 11        0023          RL   C
0015  05           0024          DEC  B
0016  28 06        0025          JRZ  MO-$
0018  CB 27        0026          SLA  A                ;DIE SCHON ERRECHNETEN STELLEN WERDEN
                    0027                                ;IN DIE RICHTIGE STELLUNG GEBRACHT
001A  CB 11        0028          RL   C
001C  18 E6        0029          JR   ZYK-$
001E  6F           0030  MO:     LD   L,A              ;ERGEBNIS NACH HL
001F  61           0031          LD   H,C
0020  C9           0032          RET
0021               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:


```
                    ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
  LOC    OBJ.CODE   STMT     SOURCE PROGRAM
=============================================================================04/07/31===001==========
                    0001            PN   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    ;**PROGRAMMABSCHNITT 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  CB 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    ;**PROGRAMMABSCHNITT 2**
                    0029    ;UMWANDLUNG BCD-ASCII
  0015  DD E1       0030            POP  IX              ;PUFFERADRESSE IN IX
  0017  C6 30       0031            ADD  30H
  0019  DD 77 00    0032            LD   (IX),A          ;1 STELLE-PUFFER
  001C  06 04       0033    Z0:     LD   B,4             ;FUER 4BCD-STELLEN
  001E  DD 23       0034    Z1:     INC  IX
  0020  AF          0035            XOR  A
  0021  0E 04       0036            LD   C,4             ;VERSCHIEBUNG 4STELLEN
  0023  29          0037            ADD  HL,HL           ;AHL
  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 *U 880* 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

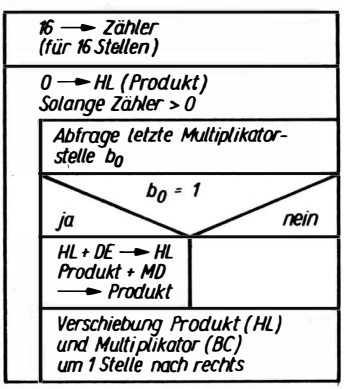
$$MR = b_{15} \cdot 2^{15} + b_{14} \cdot 2^{14} + \ldots b_1 \cdot b^1 + b_0 \,, \tag{4}$$

so erkennen wir aus

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

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-

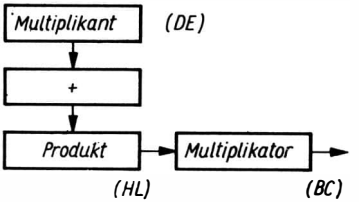


Bild 6.1
Funktionsprinzip für die Multiplikation

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.

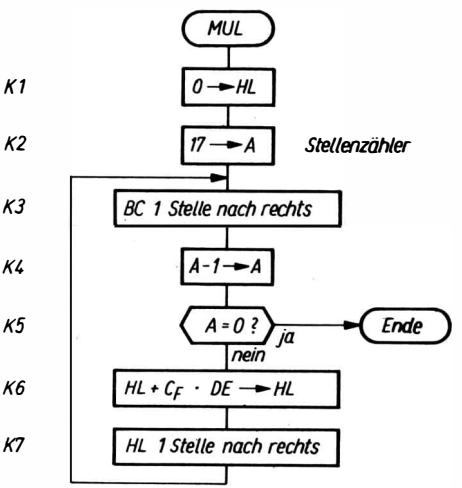


Bild 6.2 Flußdiagramm für die Multiplikation von 2 16stelligen Dualzahlen

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
==========================================================04/07/27===001==========
                    0001          PN   B1
                    0002    ;BASISPROGRAMM FUER MULTIPLIKATION
                    0003    ;16 X 16 BIT - MULTIPLIKATION, UNSIGNIERT,ERGEBNIS 32 BIT
                    0004    ;MULTIPLIKANT IN DE
                    0005    ;MULTIPLIKATOR IN BC
                    0006
0000  21 00 00      0007    MUL2:  LD   HL,0
0003  3E 11         0008           LD   A,17              ;ZAEHLER
0005  CB 18         0009    K3:    RR   B
0007  CB 19         0010           RR   C
0009  3D            0011           DEC  A
000A  C8            0012           RZ
000B  30 01         0013           JRNC K7-$
000D  19            0014           ADD  HL,DE
000E  CB 1C         0015    K7:    RR   H
0010  CB 1D         0016           RR   L
0012  18 F1         0017           JR   K3-$
0014                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

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

$$\text{oder} \quad \frac{\mathrm{DV}}{\mathrm{DR}} = \mathrm{Q} + \frac{\mathrm{R}}{\mathrm{DR}}\,. \tag{6}$$

Bringt man die Gleichung auf die Form

$$\mathrm{R} = \mathrm{DV} - \mathrm{Q} \cdot \mathrm{DR} \tag{7}$$

und schreibt den Quotienten Q als Dualzahl

$$\mathrm{Q} = \mathrm{Q}_{15} \cdot 2^{15} + \mathrm{Q}_{14} \cdot 2^{14} + \ldots \mathrm{Q}_1 \cdot 2^1 + \mathrm{Q}_0\,, \tag{8}$$

so kann man unser Verfahren aus der Gleichung

$$\mathrm{R} = \mathrm{DV} - 2^{15}\mathrm{Q}_{15}\mathrm{DR} - 2^{14}\mathrm{Q}_{14}\mathrm{DR} - \ldots 2^1\mathrm{Q}_1\mathrm{DR} - \mathrm{Q}_0\mathrm{DR} \tag{9}$$

ableiten.

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

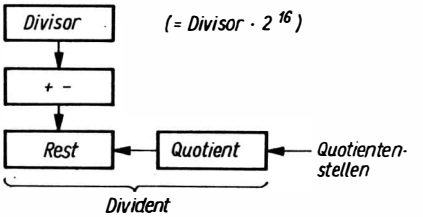


Bild 6.3
Funktionsprinzip
für die Division

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 Dividendenregister der Rest.
Da im Divisorregister der Divisor · 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: Dividend in HL BC
Divisor in DE
Ausgangsparameter: Quotient in BC
Bild 6.4 zeigt das Divisionsschema, wenn wir dem Dividenden, dem Divisor und dem Quotienten bestimmte Register zuweisen.
Als zusätzliches Register zum Auffangen der vordersten Dividendenstelle 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

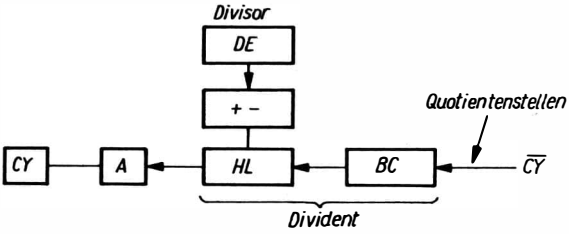


Bild 6.4 Blockschema für Division 32 Bit: 16 Bit → 16 Bit ohne Vorzeichen mit *U 880*

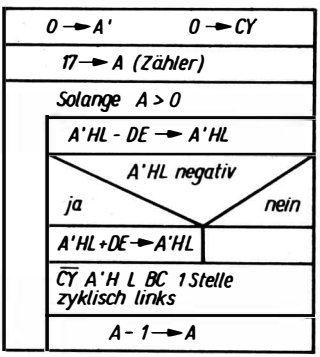


Bild 6.5
Struktogramm für die Division

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

```
                ABS. MAKROASSEMBLER 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
0000  AF         0006  DIV:    XOR  A
0001  08         0007          EXAF                    ;A=0
0002  3E 11      0008          LD   A,17               ;ZAEHLER=17
0004  08         0009  K3:     EXAF
0005  ED 52      0010          SBC  HL,DE              ;SUBTRAKTION
                  0011                                  ;A HL- DE-A HL
0007  DE 00      0012          SBC  0
0009  30 03      0013          JRNC K6-$
000B  19         0014          ADD  HL,DE              ;ADDITION WENN ERGEBNIS NEGATIV
000C  CE 00      0015          ADC  0
000E  3F         0016  K6:     CCF                     ;CY' = CY
000F  CB 11      0017          RL   C                  ;VERSCHIEBUNG
0011  CB 10      0018          RL   B
0013  CB 15      0019          RL   L
0015  CB 14      0020          RL   H
0017  17         0021          RLA
0018  08         0022          EXAF
0019  3D         0023          DEC  A                  ;A-1-A
001A  20 E8      0024          JRNZ K3-$
001C  C9         0025          RET
001D             0026          END
                PROGRAM CONTAINS   0000 ERROR(S)
```

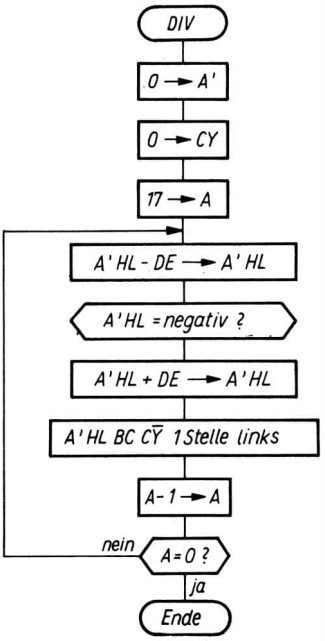


Bild 6.6
Flußdiagramm zur Division

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 Flußbilddarstellung 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-Dividenden 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  B7          0008         OR   A             ;CY=0
0003  ED 52       0009  ZVK:   SBC  HL,DE
0005  30 01       0010         JRNC KRUE-#        ;BEI CY=0 KEINE RUECKSTELLUNG
0007  19          0011         ADD  HL,DE         ;RESTRUECKSTELLUNG
0008  3F          0012  KRUE:  CCF
0009  CB 11       0013         RL   C             ;VERSCHIEBUNG DIVIDENT
000B  CB 10       0014         RL   B
000D  29          0015         ADD  HL,HL
000E  3D          0016         DEC  A
000F  20 F2       0017         JRNZ ZVK-#
0011  C9          0018         RET
0012              0019         END
                PROGRAM CONTAINS  0000 ERROR(S)
```

Da der Divisor gegenüber dem Dividenden 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 Dividend um 2 Stellen nach links verschoben werden, da sich der Divisor in DE nicht weiter nach rechts schieben läßt.

$$Q = Q_{14} \cdot 2^{14} + \ldots Q_0 \,. \tag{10}$$

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 MEOS 1521 V4.2
   LOC    OBJ.CODE   STMT     SOURCE PROGRAM
=============================================================================84/07/31===001==========
                    0001              PN   B4
                    0002    ;MULTIPLIKATION MIT VORZEICHEN (ZWEIERKOMPLEMENT)
                    0003    ;DE X BC NACH HLBC
                    0004
   0000  7A         0005    MUL2U:  LD   A,D            ;BERECHNUNG ERGEBNISVORZEICHEN
   0001  A8         0006            XOR  B
   0002  E6 80      0007            AND  80H
   0004  F5         0008            PUSH AF
   0005  CB 7A      0009            BIT  7,D            ;BETRAG MULTIPLIKANT
   0007  28 06      0010            JRZ  M3-*
   0009  21 00 00   0011            LD   HL,0
   000C  ED 52      0012            SBC  HL,DE
   000E  EB         0013            EX   DE,HL
   000F  CB 78      0014    M3:     BIT  7,B            ;BETRAG MULTIPLIKATOR
   0011  28 08      0015            JRZ  M4-*
   0013  21 00 00   0016            LD   HL,0
   0016  AF         0017            XOR  A
   0017  ED 42      0018            SBC  HL,BC
   0019  44         0019            LD   B,H
   001A  4D         0020            LD   C,L
   001B  CD 32 00   0021    M4:     CALL MUL2           ;PRODUKT DER BETRAEGE
   001E  F1         0022            POP  AF             ;KOMPLEMENT PRODUKT
   001F  CB 7F      0023            BIT  7,A
   0021  28 0E      0024            JRZ  M5-*
   0023  91         0025            SUB  C
   0024  4F         0026            LD   C,A
   0025  3E 00      0027            LD   A,0
   0027  98         0028            SBC  B
   0028  47         0029            LD   B,A
   0029  3E 00      0030            LD   A,0
   002B  9D         0031            SBC  L
   002C  6F         0032            LD   L,A
   002D  3E 00      0033            LD   A,0
   002F  9C         0034            SBC  H
   0030  67         0035            LD   H,A
   0031  C9         0036    M5:     RET
                    0037    ;MULTIPLIKATIONSROUTINE
   0032  21 00 00   0038    MUL2:   LD   HL,0
   0035  3E 11      0039            LD   A,17
   0037  CB 18      0040    K3:     RR   B
   0039  CB 19      0041            RR   C
   003B  3D         0042            DEC  A
   003C  C8         0043            RZ
   003D  30 01      0044            JRNC K7-*
   003F  19         0045            ADD  HL,DE
   0040  CB 1C      0046    K7:     RR   H
   0042  CB 1D      0047            RR   L
   0044  18 F1      0048            JR   K3-*
   0046             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)

```
                ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
  LOC    OBJ.CODE   STMT    SOURCE PROGRAM
=================================================================04/07/31===001==========
                   0001            PN   85
                   0002    ;DIVISIONSROUTINE MIT VORZEICHEN (ZWEIERKOMPLEMENT)
                   0003    ;32 BIT:16 BIT ERGIBT 16 BIT
                   0004    ;HLBC:DE NACH BC
                   0005
0000  7C           0006    DIV2U:  LD   A,H            ;BERECHNUNG ERGEBNISVORZEICHEN
0001  AA           0007            XOR  D
0002  E6 80        0008            AND  80H
0004  F5           0009            PUSH AF
0005  CB 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  CB 7C        0017    M8:     BIT  7,H            ;BETRAG DIVIDEND
0013  28 10        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 BETRAEGE
0028  F1           0032            POP  AF             ;BETRAG QUOTIENT
0029  CB 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            RET                 ;DIVISIONSROUTINE
0036  3E 11        0042    DIV2:   LD   A,17
0038  B7           0043            OR   A
0039  ED 52        0044    ZYK:    SBC  HL,DE
003B  30 01        0045            JRNC KRUE-#
003D  19           0046            ADD  HL,DE
003E  3F           0047    KRUE:   CCF
003F  CB 11        0048            RL   C
0041  CB 10        0049            RL   B
0043  29           0050            ADD  HL,HL
0044  3D           0051            DEC  A
0045  20 F2        0052            JRNZ ZYK-#
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 μs), 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

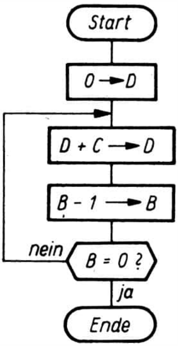


Bild 6.7
Flußbild zur Multiplikation von 2 BCD-Zahlen mit 2 Stellen (C · B → D)

```
                    ASS. MACROASSEMBLER K1520 V1 MEOS 1521 V4.2
    LOC     OBJ.CODE    STMT       SOURCE PROGRAM
=====================================================. ===================84/0/31===001==========
                        0001           PN    B6
                        0002   ;MULTIPLIKATION VON 2 BCD-ZAHLEN MIT 2 STELLEN
                        0003   ;MULTIPLIKANT IN REGISTER C
                        0004   ;MULTIPLIKATOR IN REGISTER B
                        0005   ;PRODUKT IN REGISTER D (MAXIMAL 2 STELLEN)
                        0006
    0000    16 00       0007           LD    D,0              ;PRODUKT=0 SETZEN
    0002    7A          0008   M1:     LD    A,D              ;ADDITION MULTIPLIKANT
    0003    81          0009           ADD   C
    0004    27          0010           DAA
    0005    57          0011           LD    D,A
    0006    78          0012           LD    A,B              ;ZAEHLUNG MULTIPLIKATOR
    0007    D6 01       0013           SUB   1
    0009    27          0014           DAA
    000A    47          0015           LD    B,A
    000B    20 F5       0016           JRNZ  M1-#
    000D    C9          0017           RET
    000E                0018           END
                    PROGRAM CONTAINS  0000 ERRORS.
```

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

```
                        ABS. MACROASSEMBLER K1520 V1 MEOS 1521 V4.2
   LOC    OBJ.CODE   STMT     SOURCE PROGRAM
=========================================================================84/07/31===001==========
                      0001          PN   87
                      0002
                      0003  ;GLEITKOMMAMULTIPLIKATION:2 BYTE MANTISSE,1BYTE EXPONENT
                      0004  ;ZAHLENDARSTELLUNG:MANTISSE NORMALISIERT,BETRAG + VORZEICHEN
                      0005  ;                 (15 + 1BIT)
                      0006  ;EXPONENT: ZWEIERKOMPLEMENT (7 + 1 BIT)
                      0007  ;EINGANGSPARAMETER:MANTISSE MULTIPLIKANT IN DE
                      0008  ;                          EXPONENT MULTIPLIKANT IN H
                      0009  ;                          MANTISSE MULTIPLIKATOR IN BC
                      0010  ;                          EXPONENT MULTIPLIKATOR IN L
                      0011  ;AUSGANGSPARAMETER:MANTISSE PRODUKT IN HL
                      0012  ;                          EXPONENT PRODUKT IN A
                      0013  ;BEI UEBERLAUF IST DAS P/U-FLAG GESETZT,SONST RUECKGESETZT
                      0014  ;BERECHNUNG VORZEICHEN
0000  7A              0015  GMUL:  LD   A,D
0001  A8              0016         XOR  B
0002  E6 80           0017         AND  80H              ;VORZEICHEN ERGEBNISMANTISSE AUF BIT
0004  CB B8           0018         RES  7,B              ;VORZEICHEN MULTIPLIKATOR LOESCHEN
0006  CB BA           0019         RES  7,D              ;VORZEICHEN MULTIPLIKANT LOESCHEN
0008  E5              0020         PUSH HL
0009  F5              0021         PUSH AF
                      0022  ;MULTIPLIKATION MANTISSEN
000A  21 00 00        0023  MUL:   LD   HL,0
000D  3E 11           0024         LD   A,11H
000F  CB 19           0025  K1:    RR   C
0011  CB 1A           0026         RR   D
0013  2D              0027         DEC  L
0014  20 09           0028         JRZ  NRM-*
0016  30 01           0029         JRNC K2-*
0018  19              0030         ADD  HL,DE
0019  CB 1C           0031  K2:    RR   H
001B  CB 1D           0032         RR   L
001D  1E F0           0033         JR   K3-*
                      0034  ;NORMALISIERUNG UND STELLENIDENTITAET HERSTELLEN
001F  3E 01           0035  NRM:   LD   A,1
0021  CB 10           0036  SHI:   RL   B
0023  CB 15           0037         RL   L
0025  CB 14           0038         RL   H
0027  0D              0039         DEC  C
0028  CB 74           0040         BIT  6,H
002A  28 F5           0041         JRZ  SHI-*
                      0042  ;RUNDUNG
002C  CB 10           0043  RND:   RL   B
002E  11 00 00        0044         LD   DE,0
0031  ED 5A           0045         ADC  HL,DE
0033  CB 7C           0046         BIT  7,H
0035  28 03           0047         JRZ  UL-*
                      0048  ;NORMALISIERUNG NACH DER RUNDUNG
0037  CB 1C           0049         RR   H
0039  0C              0050         INC  C
                      0051  ;VORZEICHEN EINSETZEN
003A  F1              0052  UL:    POP  AF
003B  B4              0053         OR   H
003C  67              0054         LD   H,A
                      0055  ;BERECHNUNG DES EXPONENTEN
003D  7D              0056  EXP:   POP  DE
003E  7B              0057         LD   A,E
003F  82              0058         ADD  A,D
0040  EA 44 00        0059         JPPE END              ;UEBERLAUF
0043  81              0060         ADD  A,C
0044  C9              0061  END:   RET
0045                  0062         END
                    PROGRAM CONTAINS  0000 ERROR(S)
```

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 6 5 4 3 2 1

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.

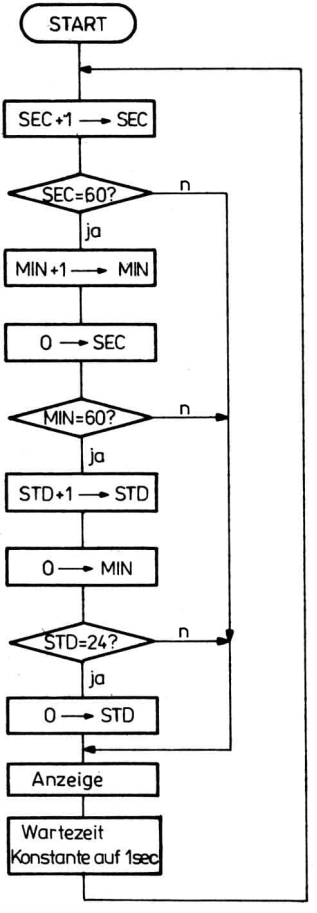


Bild 6.8
Flußdiagramm zur Errechnung von Sekunden, Minuten und Stunden eines Uhrenprogramms

Für den 1. Schritt läßt sich das in Bild 6.8 gezeigte Ablaufdiagramm aufstellen. Im 2. Schritt werden die Zahlenwerte für Sekunden, Minuten und Stunden der Reihe nach aus den zugehörigen Registern geholt. Nach Trennung der unteren und oberen 4 Bit (BCD-Ziffern) speichert man sie in den Datenpuffer für die LED-Anzeige. Anschließend erfolgt die Umwandlung in den LED-Code und die Ausgabe auf die LED-Elemente.

```
                        ;PROGRAMM ZUM ABSPEICHERN VON REGISTER A IN DEN LED-PUFFER
                        ;IM LED-PUFFER STEHEN DIE BEIDEN HEX-ZIFFERN
                        ;IN 2 ZAHLEN RECHTSBUENDIG
                        ;IN HL STEHT DIE LED-PUFFERADRESSE -HL WIRD UM 2 ERHOEHT
0035  47                ABSF:   LD    B,A
0036  E6 0F                      AND   0FH
0038  77                        LD    (HL),A          ;UNTERE 4 BIT -LED-PUFFER
0039  23                      INC   HL
003A  0F                        RRCA
003B  0F                        RRCA
003C  0F                        RRCA
003D  0F                        RRCA
003E  E6 0F                     AND   0FH
0040  77                        LD    (HL),A          ;OBERE 4 BIT -LED-PUFFER
0041  23                      INC   HL
0042  C9                       RET
                        ;ANZEIGEPROGRAMM
                        ;SPEICHERT DATEN FUER SEK,MIN UND STD IN DEN LED-PUFFER
0043  21 6A 00          ANZE:   LD    HL,LP           ;LED-PUFFERADRESSE
0046  3A 67 00                  LD    A,(U)
0049  CD 35 00                  CALL  ABSF            ;SEK -LED-PUFFER
004C  3A 68 00                  LD    A,(U+1)
004F  CD 35 00                  CALL  ABSF            ;MIN -LED-PUFFER
0052  3A 69 00                  LD    A,(U+2)
0055  CD 35 00                  CALL  ABSF            ;STD -LED-PUFFER
0058  ED 4B 70 00       ZS:     LD    BC,(ZK)         ;ZAEHLKONSTANTE FUER 1 SEC
005C  C5                        PUSH  BC              ;ZEITSCHLEIFE FUER 1 SEC
005D  CD 88 00                  CALL  LEDA            ;LED-ANZEIGE
0060  C1                        POP   BC
0061  0B                       DEC   BC              ;ZAEHLUNG FUER 1 SEC
0062  78                       LD    A,B
0063  B1                       OR    C
0064  20 F6                     JRNZ  ZS-#
0066  C9                       RET
0067  00                U:    DB    0               ;SEK-SPEICHERZELLE
0068  00                      DB    0               ;MIN-SPEICHERZELLE
0069  00                      DB    0               ;STD-SPEICHERZELLE
006A                    LP:   BER   6               ;6 ZELLEN FUER LED-PUFFER
```

```
  LOC    OBJ.CODE    STMT     SOURCE PROGRAM                    C103
=================================================================84/08/15===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   3B          0076        DB    3BH
 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
                    0090  ;AUSGABE AUF LED-ELEMENTE
 0088   FD 21 72 00 0091  LEDA:  LD    IY,TOR
 008C   DD 21 6A 00 0092        LD    IX,LP
 0090   06 06       0093        LD    B,6
 0092   21 78 00    0094        LD    HL,STAB
 0095   FD 4E 00    0095  ZYK:   LD    C,(IY)        ;ELEMENTADRESSE NACH C
 0098   DD 5E 00    0096        LD    E,(IX)        ;ZIFFER NACH E
 009B   16 00       0097        LD    D,0
 009D   19          0098        ADD   HL,DE
 009E   7E          0099        LD    A,(HL)        ;SEGMENTCODE NACH A
 009F   ED 79       0100        OUT   A
 00A1   DD 23       0101        INC   IX
 00A3   FD 23       0102        INC   IY
 00A5   10 EE       0103        DJNZ  ZYK-#
 00A7   C9          0104        RET
 00A8               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:

$Z = X$

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

Verfahren:

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

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

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

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:

$$P_{K-1} = 0, \quad S_1S_2 \ldots S_{K-1}S_KS_{K+1} \ldots S_n\,, \tag{15}$$

wobei $S_1S_2 \ldots S_n$ die Dualstellen von P_{K-1} sind.
Weiterhin gilt:

$$\sum_{j=1}^{K-1} X_j \cdot 2^{-1} = 0\,, \quad X_1X_2 \ldots X_{K-1}\,. \tag{16}$$

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

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

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

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

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

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

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

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

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

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

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

1. Voraussetzungen — Dualwert

a) Bereitstellen von $P_0 = Z$ — $0\ Z_1\ Z_2 \ldots Z_n$

b) $X = 0, X_1 X_2 \ldots X_n = 0$ setzen — $0\ 0\ 0 \ldots 0$

c) Bereitstellen einer Konstanten K zum Bit der Ziffer X_K — $0\ 1\ 0\ 0 \ldots 0$

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

```
0001                                    ; Wurzelprogramm
0002
0003                                    ; Radikand    0,D1...D15   In HL
0004                                    ; Wurzel      0,W1...W15   In DE
0005
0006                                    PN      W2
0007  0000  010040    WU2R:             LD      BC,4000H
0008  0003  110000                      LD      DE,0
0009  0006  224200    K4:               LD      (ZW),HL
0010  0009  AF                        XOR     A
0011  000A  ED52                      SBC     HL,DE
0012  000C  381D                      JRC     K17-#                   ;Abfrage K6
0013  000E  CB15                      RL      L
0014  0010  CB14                      RL      H
0015  0012  AF                        XOR     A
0016  0013  A9                        XOR     C
0017  0014  201E                      JRNZ    K19-#                   ;Abfrage
0018  0016  7C                        LD      A,H
0019  0017  90                        SUB     B
0020  0018  67                        LD      H,A
0021  0019  3810                      JRC     K17-#                   ;Abfrage K10
0022  001B  7A                        LD      AD
0023  001C  B0                        OR      B
0024  001D  57                        LD      D,A
0025  001E  CB18    K12:              RR      B
0026  0020  30E4                      JRNC    K4-#                    ;Abfrage K13
0027  0022  CB18                      RR      B
0028  0024  AF                        XOR     A
```

```
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 K17:  LD    HL,(ZW)
0034  002E  CB25        SLA   L
0035  0030  CB14        RL    H
0036  0032  18EA        JR    K12-*
0037  0034  7D    K19;  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  ZW:    DA    0
0049                    END
Keine Syntax-Fehler CRAS 4200-K1520
```

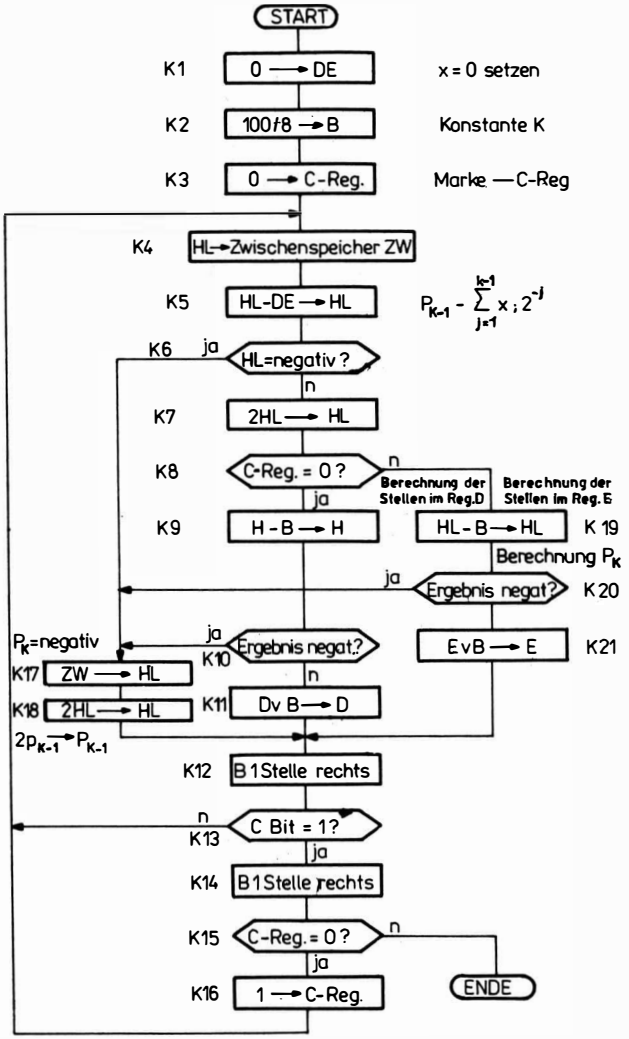


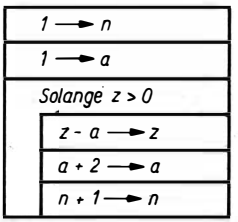
Bild 6.9 Flußdiagramm für das Quadratwurzelziehen aus einer 15stelligen Dualzahl mit $|z| < 1$

Beispiel 16

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

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

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



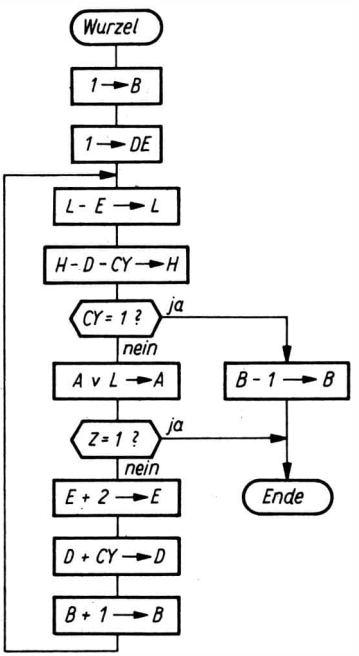


Bild 6.10
Flußbild: Wurzel aus einer ganzen Dezimalzahl

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 MEOS 1521 V4.2
  LOC     OBJ.CODE   STMT     SOURCE PROGRAM
=========================================================================04/08/83==001==========
                    0001          PN   B8
                    0002
                    0003  ;QUADRATWURZEL AUS EINER GANZEN DEZIMALZAHL MIT 4 BCD-STELLEN
                    0004  ;RADIKANT IN HL
                    0005  ;WURZEL IN B
                    0006
  0000  06 01       0007          LD   B,1
  0002  11 01 00    0008          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  38 14       0016          JRC  M3-#
  000E  B5          0017          OR   L
  000F  28 16       0018          JRZ  M2-#
  0011  7B          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  78          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

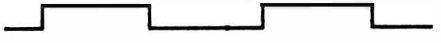


Bild 6.11
Impulsfolge mit einem Tastverhältnis von 1:1

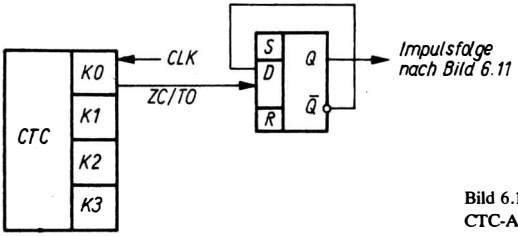


Bild 6.12
CTC-Ausgang mit D–FF

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 μs 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

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

$Z \cdot K \cdot 2 = 1000$
mit $K = 16$

$$Z = \frac{1000}{16.2} = 31{,}25 \approx 31\ . \tag{21}$$

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

Dieses Unterprogramm wird wie folgt aufgerufen:

```
LD B, 31
LD C, 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 Φ 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. $f = 2{,}4$ 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-

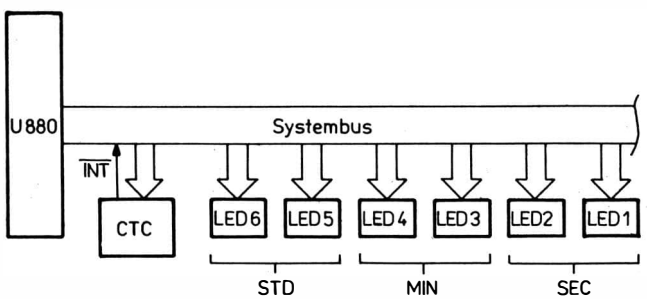


Bild 6.13 Zusammenschaltung der Bausteine *U 880* und *U 857 D* zur Realisierung einer Digitaluhr

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:

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

Mit $t_i = 0{,}01$ 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,O          ;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  81 H
        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
        CALLLED             ; SEC, MIN und STD in
                            ; Anzeigeregister
        JRANZ - *

; Interruptbedienprogramm (Zeitzählprogramm)
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 - *
        JR EN - *
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 - *
        JR EN - *
STD:    LD (HL), 0          ; 0 nach MIN
        INC HL              ; STD-Zähler nach HL
        LD A, (HL)
        ADD 1
        DAA                 ; STD + 1 → STD
        LD (HL), A
```

```
        CMP24H              ; STD = 24?
        JRZ RES - *
        JR EN - *
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
        INC HL
        POP AF
        RRCA
        RRCA
        RRCA
        RRCA
        AND OFH
        LD (HL), A          ; Obere 4 Bit nach LED-Puffer
        INC HL
        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
        LD D, 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-
                            ;Werte
TKO:    DB      60H         ;Zeitkonstante für CTC-Kanal0
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

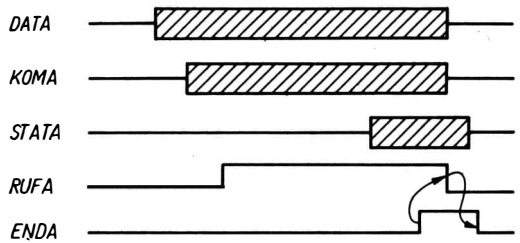


Bild 6.14 Taktdiagramm zur Datenausgabe über SIF 1000 – Spiegel

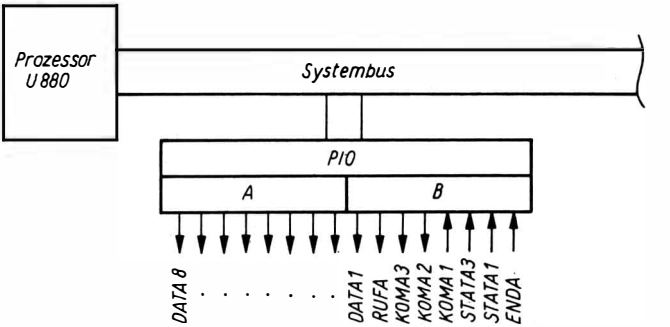


Bild 6.15 Realisierung der SIF-100-Signale mit PIO-Schaltkreis

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
==========================================================================04/00/15===001==========
                   0001          PN   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     ;INITIALISIERUNGSPROGAMM
0000  21  10 00    0007     INIT:  LD   HL,STAB         ;ADRESSE STEUERWORTTABELLE
0003  0E  00       0008            LD   C,SPIOA         ;ADRESSE STEUERWORT PIO-KANAL A
0005  06  03       0009            LD   B,3             ;ANZAHL DER STEUERWORTE
0007  ED  B3       0010            OTIR
0009  0E  00       0011            LD   C,SPIOB         ;ADRESSE STEUERWORT PIO-KANAL B
000B  06  03       0012            LD   B,3             ;ANZAHL DER STEUERWORTE
000D  ED  B3       0013            OTIR
000F  C9           0014            RET
                   0015     ;INITIALISIERUNGSDATEN
                   0016     ;KANAL A
0010  FF           0017     STAB:  DB   0FFH            ;BETRIEBSART BITMODE
0011  00           0018            DB   0              ;BIT 0-7 AUSGABE
0012  07           0019            DB   7H             ;INTERRUPTVERBOT
                   0020     ;KANAL B
0013  FF           0021            DB   0FFH           ;BETRIEBSART BITMODE
0014  0F           0022            DB   0FH            ;BIT 0-3 EINGABE,BIT 4-7 AUSGABE
0015  07           0023            DB   7H             ;INTERRUPTVERBOT
0016               0024            END
                 PROGRAM CONTAINS  0000 ERROR(S)
```

Beispiel 20

```
                    ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
 LOC    OBJ.CODE   STMT     SOURCE PROGRAM
==========================================================================04/00/15===001==========
                   0001          PN   C3
                   0002
                   0003     ;PROGRAMM ZUR AUSGABE EINES ZEICHENS AUS DEM REGISTER A
                   0004     ;-PIO ADRESSE TOR A IN REGISTER D
                   0005     ;-PIO ADRESSE TOR B IN REGISTER E
                   0006     ;-SIF 1000 KOMMANDO STEHT IN REGISTER C
                   0007     ;-SIF 1000 STATUS KOMMT NACH REGISTER B
                   0008     ;AUSGABE ZEICHEN NACH KANAL A
0000  ED 51        0009     AUS:   OUT  D
0002  79           0010            LD   A,C
0003  ED 59        0011            OUT  E               ;AUSGABE KOMMANDO NACH KANAL B
0005  F6 8C        0012            OR   8CH
0007  ED 59        0013            OUT  E               ;AUSGABE KOMMANDO UND RUFA
0009  ED 58        0014     WS1:   IN   E
000B  CB 47        0015            BIT  0,A             ;TEST AUF ENDA
000D  28 FA        0016            JRZ  WS1-#
000F  E6 0E        0017            AND  0EH             ;AUSBLENDEN STATUS
0011  47           0018            LD   B,A             ;STATUS NACH REGISTER B
0012  3E 00        0019            LD   A,0
0014  ED 59        0020            OUT  E               ;ABSCHALTEN RUF UND KOMMANDO
0016  ED 58        0021     WS2:   IN   E
0018  CB 47        0022            BIT  0,A
001A  28 FA        0023            JRZ  WS2-#           ;TEST AUF ABGESCHALTETES ENDA
001C  C9           0024            RET
001D               0025            END
                 PROGRAM CONTAINS  0000 ERROR(S)
```

Beispiel 21

```
                ABS. MACROASSEMBLER K1520 /1 MEOS 1521 V4.2
   LOC    OBJ.CODE   STMT     SOURCE PROGRAM
=========================================================================04/08/15===001==========
                    0001          PW    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
   0000  7E         0011  ASA:   LD   A,(HL)
   0001  E5         0012         PUSH HL
   0002  C5         0013         PUSH BC
   0003  61         0014         LD   H,C
   0004  CD 00 00   0015         CALL AUS              ;AUSGABE EINES ZEICHENS MIT PROGRAMM
                    0016                               ;AUS BEISPIEL 18
   0007  C1         0017         POP  BC
   0008  E1         0018         POP  HL
   0009  7E         0019         LD   A,(HL)
   000A  B8         0020         CMP  B
   000B  28 03      0021         JRZ  ENDE-#
   000D  23         0022         INC  HL
   000E  18 F0      0023         JR   ASA-#
   0010  C9         0024  ENDE:  RET
   0011              0025         END
                   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 Gerätesteuerrechner. 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: Steuerroutinen
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	Grundrechenarten
– BCD – Dual	– Addition
– Dual – BCD	– Subtraktion
– Text (ASCII) – DUAL	– Multiplikation
– DUAL – TEXT (ASCII)	– Division
– Festkomma – Gleitkomma	
– Gleitkomma – Festkomma	

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
– area sinh	– ganzer Teil einer Zahl
– areacosh	
– area tanh	

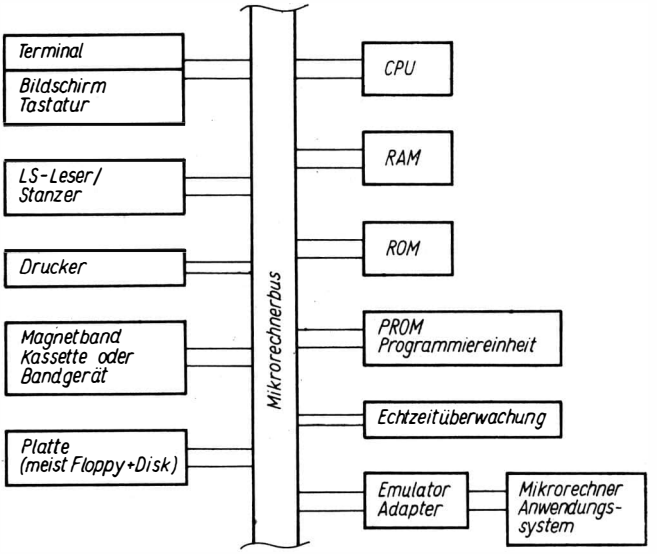


Bild 7.1 Struktur eines Mikrorechnerentwicklungssystems

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

(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 dazugehörenden 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-

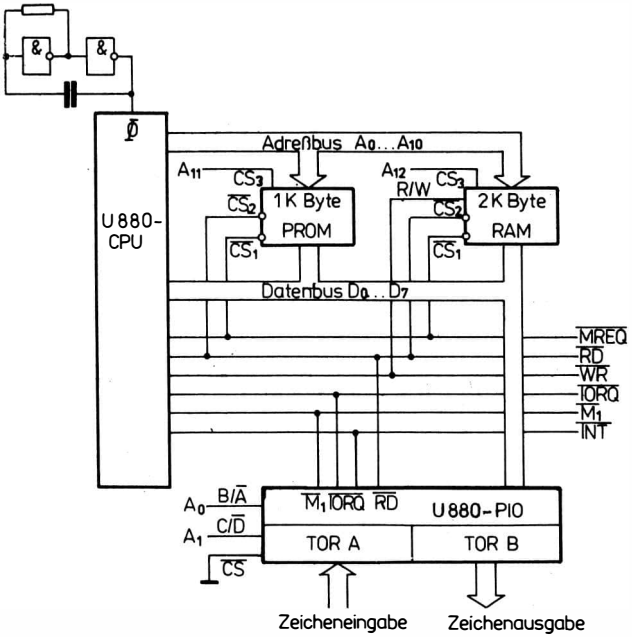


Bild 8.1 Struktur eines Einkartenrechners mit dem Prozessor *U 880* zur Schreibmaschinensteuerung

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

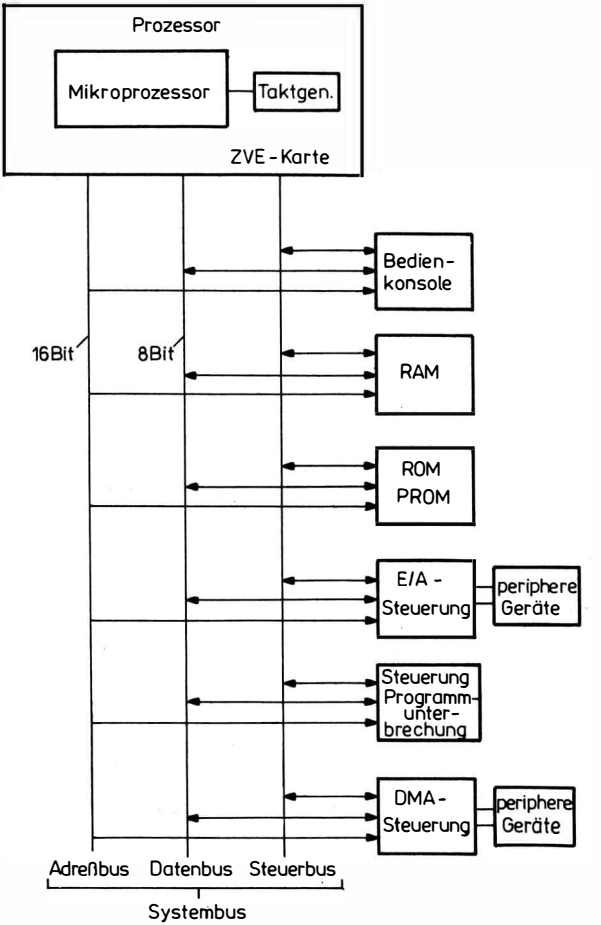
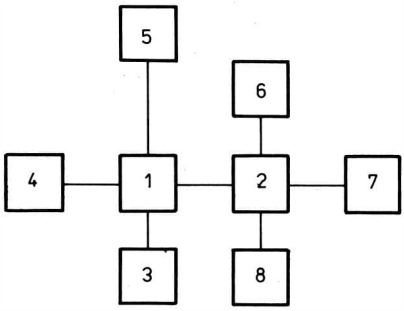


Bild 8.2 Aufbaueines Mikrorechnersystems

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.

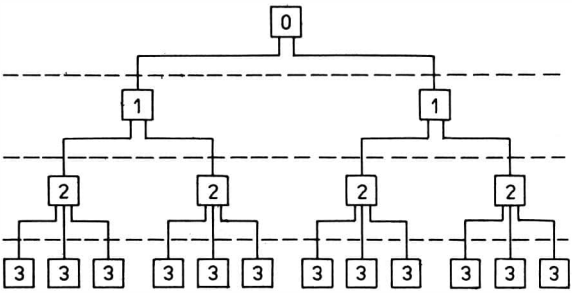
8.3. Mehrprozessorsysteme

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 Strukur 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-



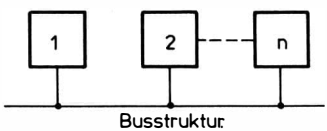
Sternstruktur

a)

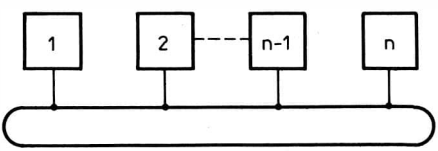


Hierarchiestruktur

b)

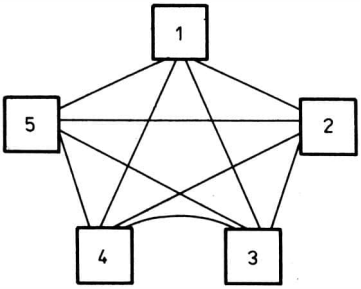


Busstruktur



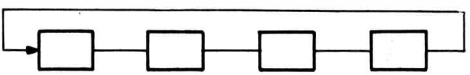
Ringbus

c)



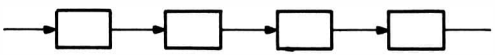
Netzstruktur

d)



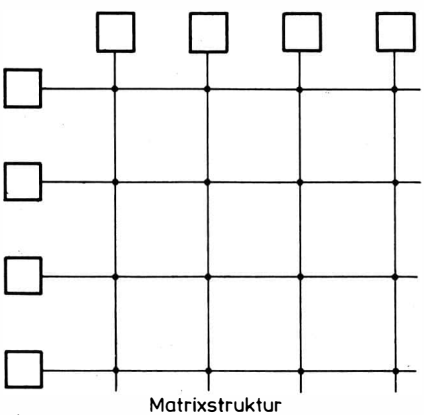
Ringstruktur

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.

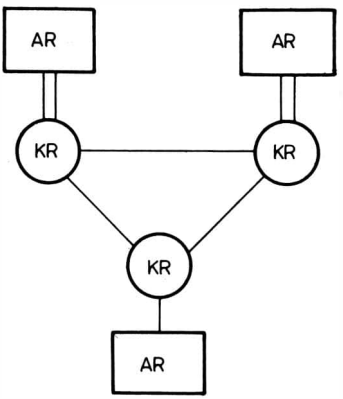
Aus Bild 8.3g ist die Matrixstruktur zu ersehen. Bei ihr läßt sich jedes Prozeßelement mit jedem anderen über die Matrixknotenpunkte verbinden. Die Knotenpunkte sind als elektronische Schalter ausgeführt, die von einem Steuerrechner des Netzwerks durchgeschaltet werden.

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

	16^0	16^1	16^2	16^3	16^4
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 → 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^n	n	16^n					
						1	0	0.10000	00000	00000	00000	x	10
						16	1	0.62500	00000	00000	00000	x	10^{-1}
						256	2	0.39062	50000	00000	00000	x	10^{-2}
					4	096	3	0.24414	06250	00000	00000	x	10^{-3}
					65	536	4	0.15258	78906	25000	00000	x	10^{-4}
				1	048	576	5	0.95367	43164	06250	00000	x	10^{-6}
				16	777	216	6	0.59604	64477	53906	25000	x	10^{-7}
				268	435	456	7	0.37252	90298	46191	40625	x	10^{-8}
			4	294	967	296	8	0.23283	06436	53869	62891	x	10^{-9}
			68	719	476	736	9	0.14551	91522	83668	51807	x	10^{-10}
		1	099	511	627	776	10	0.90949	47017	72928	23792	x	10^{-12}
		17	592	186	044	416	11	0.56843	41886	08080	14870	x	10^{-13}
		281	474	976	710	656	12	0.35527	13678	80050	09294	x	10^{-14}
	4	503	599	627	370	496	13	0.22204	46049	25031	30808	x	10^{-15}
	72	057	594	037	927	936	14	0.13877	78780	78144	56755	x	10^{-16}
1	152	921	504	606	846	976	15	0.86736	17379	88403	54721	x	10^{-18}

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}
		1	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}
	2	540B	E400	10	0.6DF3	7F67	SEF6	EADF	x	16^{-8}
	17	4876	E800	11	0.AFEB	FFOB	CB24	AAFF	x	16^{-9}
	E8	D4A5	1000	12	0.1197	9981	2DEA	1119	x	16^{-9}
	918	4E72	A000	13	0.1C25	C268	4976	81C2	x	16^{-10}
	5AF3	107A	4000	14	0.2D09	370D	4257	3604	x	16^{-11}
3	8D7E	A4C6	8000	15	0.480E	BE7B	9D58	566D	x	16^{-12}
23	8652	6FC1	0000	16	0.734A	CA5F	6226	F0AE	x	16^{-13}
163	4578	5D8A	0000	17	0.B877	AA32	36A4	B449	x	16^{-14}
DEO	B6B3	A764	0000	18	0.1272	5DD1	D243	ABA1	x	16^{-14}
8AC7	2304	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 (Start of heading)
002	STX		Anfang des Textes (Start of text)
003	ETX		Ende des Textes (End of text)
004	EOT		Ende der Übertragung (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 (Backspace)
011	HT	HT	Horizontaler Tabulator (Horizontal tab)
012	LF	LF	Zeilenvorschub (Line feed)
013	VT		Vertikaler Tabulator (Vertical tab)
014	FF		Formatsteuerung (Form feed); 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 (Shift in); Umschalten auf schwarzes Farbband
020	DLE		Umschaltung der Datenübertragung (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 (End of transmission block) auch LEM, Logisches Ende des Mediums
030	CAN		Annullierung (CANCEL)
031	EM		Ende des Mediums (End of Medium) Ende des Datenträgers
032	SUB		Substitution
033	ESC		Escape
034	FS		File seperator
035	GS		Group seperator

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

Oktaler Code	ASCII-Zeichen	SIF-1000 Zeichen	Bemerkungen
036	RS	NL	Recorf seperator; New line (NL) Wagenrücklauf mit Zeilenvorschub
037	US		Unit seperator
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