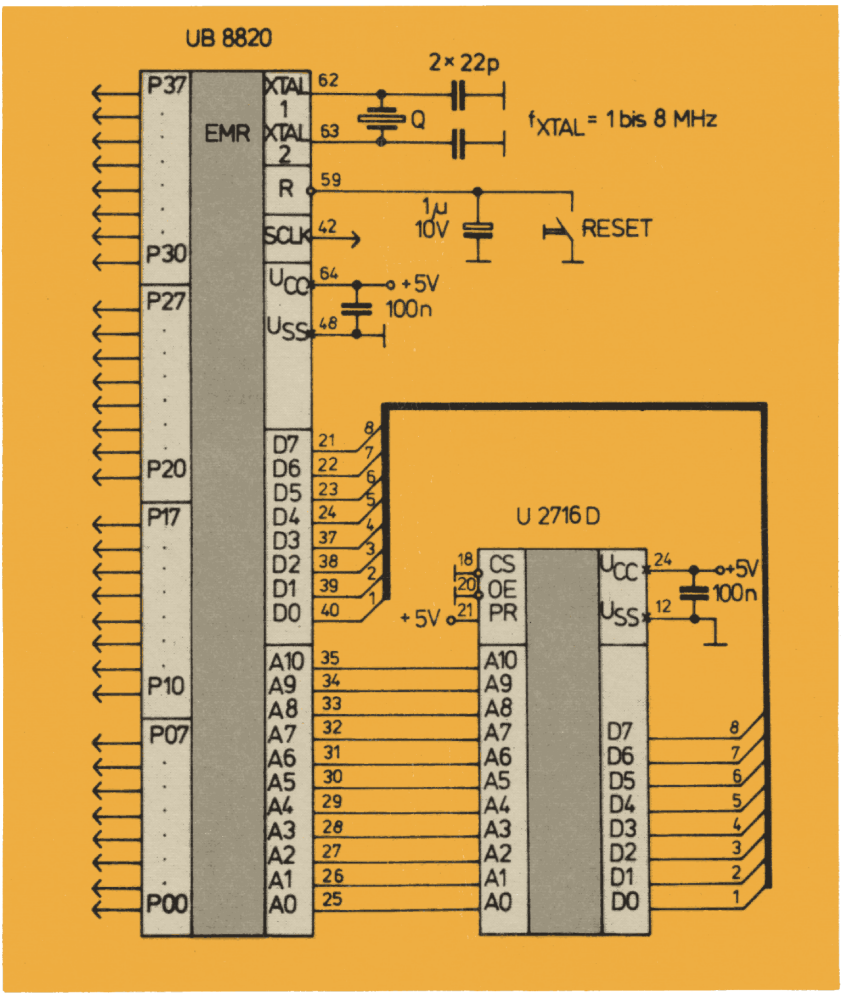
Einchip-Mikrorechner

Frank Bormann/
Michael Rentzsch



Frank Bormann
Dr. Michael Rentzsch

Einchip-Mikrorechner

Schaltungs- und Programmierpraxis

Militärverlag
der Deutschen Demokratischen Republik

Bormann, F.; Rentzsch, M.:
Einchip-Mikrorechner. Schaltungs- und Programmierpraxis. –
Berlin: Militärverlag der DDR (VEB), 1989. –
128 S.: 45 Bilder, 25 Programme

ISBN 3-327-00776-4

1. Auflage
© Militärverlag der Deutschen Demokratischen Republik
(VEB) – Berlin, 1989
Lizenz-Nr. 5
Lichtsatz: INTERDRUCK Graphischer Großbetrieb Leipzig – III/18/97
Druck und buchbinderische Weiterverarbeitung:
Druckerei des Ministeriums für Nationale Verteidigung
(VEB) - Berlin - 32341-9
Lektor: Steffen Würtenberger
Zeichnungen: Marina Jacob
Typografie: Anne-Katrin Engelstädter
Umschlaggestaltung: W. Ritter
Redaktionsschluß: 15. Januar 1989
LSV 3539
Bestellnummer: 7472064
00650

Inhaltsverzeichnis

1.	**Einordnung der Einchipmikrorechner**	5
1.1.	Abgrenzung von anderen Bauelementen der Mikrorechentechnik	5
1.2.	Übersicht zu den Einchipmikrorechnern *U 881/882*	6
2.	**Struktureller Aufbau der Einchipmikrorechner *U 881/U 882***	9
2.1.	Wesentliche Parameter und Komponenten	9
2.2.	Beschreibung einzelner Funktionskomplexe	10
3.	**Anschlußbelegung und -beschreibung**	18
4.	**Zeitverhalten**	22
5.	**Status- und Steuerregister**	27
5.1.	Funktionsbeschreibung	27
5.2.	Hinweise zur Nutzung der Interrupt-Steuerregister	35
6.	**Der Befehlssatz**	37
6.1.	Befehlsgruppen	37
6.2.	Adressierungsarten	37
6.3.	Befehlsbeschreibung	39
6.4.	Zusammenfassende Befehlsangabe	47
6.5.	Arbeitstabelle für das 1. Byte des OP-Kodes	50
7.	**Schaltungstechnik**	51
7.1.	Allgemeine Hinweise	51
7.2.	*U 882* ohne externen Speicher	52
7.3.	*U 882* mit 0,5 KByte RAM	53
7.4.	Schaltung mit *U 883* und 2 KByte externem Programmspeicher	54
7.5.	*U 883* mit 2 KByte Programmspeicher und 1 KByte Schreib-Lese-Speicher	55
7.6.	Besonderheiten bei der Erzeugung des RESET-Signals	56
7.7.	Erzeugung des Systemtakts	58
8.	**Ausbaumöglichkeiten der seriellen Schnittstelle des *U 881/U 882***	60
8.1.	Struktur	60
8.2.	Standardisierte Schnittstellen	61
8.2.1.	IFSS-Schnittstelle	62
8.2.2.	V.24-Schnittstelle	65
8.2.3.	Serielle Kopplung zwischen 2 EMR	66

9. Ausgewählte Programm- und Anwendungsbeispiele . . 74
9.1. Arithmetikprogramme 74
9.2. Konvertierungsprogramme 78
9.3. Anzeigeprogramme für Ziffernanzeigebaugruppen 79
9.3.1. Direkter Anschluß von 7-Segment-Dekodern 79
9.3.2. Anzeige mit *V40511* 80
9.3.3. Zeitmultiplexe Anzeige 81
9.4. Programme in Verbindung mit den Zähler-/Zeitgeberkanälen 83
9.4.1. Erzeugung einer Rechteck-Impulsfolge 84
9.4.2. Messung der Impulsdauer 84
9.4.3. Messung der Periodendauer einer Impulsfolge 86
9.5. Verarbeitung von Analogsignalen 87

10. Eingabegerät mit Einchipmikrorechner und Ankopplung an einen übergeordneten Rechner 89
10.1. Hardware 89
10.2. Steuerprogramm des Eingabegeräts 92
10.3. Ankopplung des übergeordneten Rechners 96

11. Alphanumerische Anzeigeeinheit mit *VQC10*, EMR-Steuerung und serieller Kopplung zu einem *U880*-System 100
11.1. Schaltungstechnik 101
11.1.1. Anzeigebaugruppe *VQC10* mit Einchiprechner 101
11.1.2. *U880*-Baugruppe mit SIO und CTC 102
11.2. Steuerprogramme 104
11.2.1. Taktsynchronisation 104
11.2.2. Steuerprogramm der Anzeigebaugruppe 105
11.2.3. *U880*-Steuerprogramm 110
11.2.4. Erzeugung von Laufschrift 113

12. Alphanumerische Tastatur mit serieller und paralleler Schnittstelle 114
12.1. Hardware 115
12.2. Software 115

13. Minimalsystem zur Programmentwicklung für Einchiprechner mit dem Kleincomputer *KC85/1* bzw. *KC87* . . . 119
13.1. 2-Wege-RAM 120
13.2. EPROM-Programmiereinrichtung 123
13.3. Software 123

Sachwortverzeichnis 127

Literatur 128

1. Einordnung der Einchipmikrorechner

1.1. Abgrenzung von anderen Bauelementen der Mikrorechentechnik

Seit etwa 1975 sind Schaltkreise der Mikrorechentechnik international verfügbar. Ausgehend von Erfahrungen bei der Herstellung elektronischer Datenverarbeitungsanlagen sowie bei der Fertigung von Tischrechnern waren die Hersteller stets bemüht, die für diese Erzeugnisse benötigten Bauelemente zu verkleinern und ihre Fertigung kostengünstig zu gestalten.

Der technologische Fortschritt bei der Herstellung integrierter Schaltkreise und dabei insbesondere die Einführung der MOS-Technik erlaubte bereits Anfang der siebziger Jahre die Herstellung von komplexen integrierten Schaltungen auf einem Chip. Unter dem Aspekt möglichst großer Produktionsstückzahlen bei gleichzeitig breitem Einsatzfeld entstanden die ersten Mikroprozessoren. Trotz unterschiedlicher Eigenschaften wie Wortbreite, Befehlsausführungsdauer u. ä. kann man heute von einer «traditionellen» Struktur eines Mikrorechners sprechen. Das Kernstück eines solchen Systems bildet die Zentrale Verarbeitungseinheit mit ihrer CPU (central processing unit). Durch die Eigenschaften des Prozessors werden wesentliche Leistungsparameter des Mikrorechners festgelegt. Ein funktionsfähiges System ist jedoch nur durch Hinzufügen weiterer Schaltkreise bzw. Baugruppen realisierbar. Neben einem Taktgenerator für die Erzeugung des Systemtakts handelt es sich dabei um Speicherschaltkreise für die Abspeicherung von Programmen und Daten sowie um Ein-/Ausgabeschaltkreise.

Die Eigenschaften der einzelnen Schaltkreise eines Systems sind optimal aufeinander abgestimmt, so daß man in diesem Zusammenhang von Mikrorechnerfamilien spricht. Der Vorteil dieses Prinzips liegt in seiner großen Flexibilität, da Speicherkapazität sowie Art und Anzahl der Ein- und Ausgänge vom Anwender in weiten Grenzen beeinflußbar sind. Die Palette reicht dabei vom Minimalsystem auf einer Leiterkarte (Einkartenrechner) bis hin zu Mehrrechnersystemen. Ein typisches Beispiel für eine derartige Anordnung ist das System *U 880*. Neben der CPU *U 880* gehören zu dieser Familie folgende LSI-Schaltkreise:

U 855 – PIO (programmierbare parallele Ein-/Ausgabe),
U 856 – SIO (programmierbare serielle Ein-/Ausgabe),
U 857 – CTC (programmierbarer Zähler/Zeitgeber),
U 858 – DMA (Controller für direkten Speicherzugriff).

Für zahlreiche Aufgaben ist der Vorteil von Systemen auf der Basis von Einzelschaltkreisen mit jeweils klar abgegrenzter Funktion unumstritten. Die Entwicklung geht dabei einerseits in Richtung größerer Verarbeitungsgeschwindigkeit mit dem Ziel eines beschleunigten Datendurchsatzes; andererseits sind die Bestrebungen auf eine Vergrößerung der Verarbeitungsbreite gerichtet, um speziell arithmetische Operationen mit größerer Genauigkeit ausführen zu können. Es verbleibt jedoch ein Bereich potentieller Anwendungen, in dem traditionelle Mikrorechnersysteme nicht effektiv einsetz-

bar sind. Es handelt sich dabei meist um Aufgabenstellungen mit einer eingeschränkten Anzahl von Ein-/Ausgabeleitungen und begrenztem Datendurchsatz aus den Bereichen Steuerungs- und Regelungstechnik, Meßtechnik und Konsumgüterproduktion. Um die Vorteile der Rechentechnik auf diesen Gebieten voll ausschöpfen zu können, mußten kostengünstige und dennoch hinsichtlich der Anwendung sehr flexibel einsetzbare Lösungen geschaffen werden. Im Ergebnis dieser Bemühungen entstanden die Einchipmikrorechner. Es handelt sich dabei um hochintegrierte Schaltkreise, die zusammen mit wenigen externen Bauelementen einen funktionsfähigen Mikrorechner bilden.

Einchipmikrorechner zeichnen sich vor allem durch die Unterbringung von Arbeitsspeicher und CPU auf einem Chip aus. Im Gegensatz zu herkömmlichen Schreib-Lese-Speichern bilden die im Einchipmikrorechner implementierten Speicherzellen einen Registersatz, bei dem jedes Register direkt mit der Arithmetik-Logik-Einheit (ALU) Daten austauschen kann. Der vom *U880*-System und anderen Mikroprozessoren her bekannte Akkumulator ist nicht gesondert vorhanden. Jedes Universalregister des Registersatzes des Einchipmikrorechners kann Ziel oder Quelle einer Operation sein. Die damit verbundene Einsparung zahlreicher Transportbefehle zwischen Speicher und CPU führt zu einer wesentlichen Erhöhung des Datendurchsatzes.

Ein weiterer Unterschied zu traditionellen Mikrorechnersystemen besteht in der Vereinfachung von Ein-/Ausgabeoperationen. Die als Anschlußleitungen des Schaltkreises zugänglichen Ein-/Ausgabetore werden durch das Programm wie Universalregister behandelt, spezielle Ein-/Ausgabebefehle sind daher nicht notwendig. Das Lesen der Eingabedaten geschieht durch Abfrage und die Ausgabe von Daten durch Laden der zugeordneten Register, wenn das betreffende Tor für die jeweilige Betriebsart initialisiert ist.

Durch die Unterbringung der gesamten Rechnerstruktur auf einem Chip wird auch die Interruptverarbeitung vereinfacht. Da die einzelnen Interruptquellen exakt definiert sind, ist eine externe Prioritätslogik nicht notwendig. Sie wird durch eine programmtechnisch festlegbare Wichtung der einzelnen Quellen ersetzt.

1.2. Übersicht zu den Einchipmikrorechnern U881 und U882

Die vom VEB *Kombinat Mikroelektronik «Karl Marx»* gefertigten Einchipmikrorechner bestehen aus den Grundtypen *U881* und *U882* und sind funktionell kompatibel zu den Vergleichstypen *Z8601* und *Z8602* der Firma *Zilog*. Die Unterschiede zwischen den Typen *U881* und *U882* beziehen sich nicht auf den Befehlssatz oder andere wesentliche Leistungsparameter, sondern auf die Bauform. Als Anwenderschaltkreis im ursprünglichen Sinne steht der *U881* zur Verfügung. Er ist gekennzeichnet durch:

– 40poliges DIL-Gehäuse, interner maskenprogrammierter Programmspeicher von 2 KByte.

Der *U882* war zunächst als Entwicklungsversion für den *U881* gedacht. Der Programmspeicher ist bei diesem Typ extern anschließbar. Dazu sind die ansonsten zum internen ROM führenden Adreß- und Datenleitungen nach außen geführt. Diese Maßnahme erfordert notwendigerweise eine veränderte Bauform:

– 64poliges QUIL-Gehäuse, externer 2-KByte-Programmspeicher.

Außer den zusätzlichen Programmspeicherleitungen sind beim *U882* noch einige Sta-

tussignale extern zugänglich, die die Entwicklungsarbeit unterstützen. Gegenüber dem ursprünglich konzipierten Einsatz als reine Entwicklungsversion hat der *U882* wesentliche Bedeutung für Anwendersysteme erreicht. Immer dann, wenn geringe Stückzahlen die Herstellung einer ROM-Maske nicht rechtfertigen und darüber hinaus eine Verfügbarkeit aller Ein-/Ausgabeleitungen gefordert ist, stellt der Einsatz des *U882* eine günstige Variante dar. Nachteilig für den Einsatz beim Amateur ist das Fehlen geeigneter Steckfassungen für diese Bauform.

Als spezielle Version des *U881* wird der *U883* gefertigt. Sein interner Programmspeicher enthält neben einem BASIC-Interpreter auch ein sogenanntes Bootstrap-Ladeprogramm, das den Sprung auf einen extern angeschlossenen Programmspeicher ermöglicht. Durch den dazu notwendigen zusätzlichen Daten- und Adreßbus, der über Ein-/Ausgabeleitungen aufgebaut werden muß, ist eine Reduzierung der allgemein nutzbaren Ein-/Ausgabefunktionen in Kauf zu nehmen. Je nach zu installierender Speicherkapazität sind dennoch 16, 20 oder 24 Leitungen zur Ein-/Ausgabe für den Anwender verfügbar.

Eine weitere Variante des *U881* ist der Typ *U886*. Es handelt sich dabei um ein kostengünstiges Bauelement, bei dem der interne Programmspeicher nicht verwendbar ist. Durch eine spezielle Anlaufroutine, die im sonst anders verwendeten Test-ROM läuft, wird ebenfalls in den externen Programmspeicherbereich gesprungen. Dieser Sprung und damit der Start des eigentlichen Anwenderprogrammes erfordert jedoch eine zeitweilige Spannungsüberhöhung am RESET-Eingang, die nur mit zusätzlichen Bauelementen zu erreichen ist. Beim Anschluß des externen Programmspeichers an einen durch entsprechend konfigurierte Ein-/Ausgabeleitungen gebildeten Daten- und Adreßbus bestehen ähnliche Anwendungsmöglichkeiten wie beim *U883*.

Eine Weiterentwicklung des *U882* ist der *U884* (kompatibel zum *Z8612*). Er unterscheidet sich vom Grundtyp durch eine auf 4 KByte erweiterte Anschlußmöglichkeit für den internen Programmspeicher.

Alle genannten Einchipmikrorechner sind in 2 Bondversionen erhältlich: Die Standardversion läßt den direkten Anschluß eines Schwingquarzes zur Takterzeugung zu, während die sogenannte power-down-Version den Anschluß einer Stützspannung für die getrennte Stromversorgung des internen Registersatzes erlaubt. Damit besteht in dieser Version die Möglichkeit des Datenerhalts bei Betriebsspannungsausfall im internen Registersatz. Die Taktfrequenz muß in dieser Variante allerdings mit einem externen Generator erzeugt werden.

Neben den bereits genannten Unterschieden werden die Bauelemente noch hinsichtlich der maximal möglichen Taktfrequenz in 3 Gruppen eingeteilt:
UB-Variante – 8 MHz,
UC-Variante – 5 MHz,
UD-Variante – 3,6 MHz.

In der Tabelle 1.1 sind die wesentlichen Eigenschaften der genannten Versionen nochmals zusammengefaßt.

	max. Taktfrequenz 8 MHz	5 MHz	3,6 MHz
Interner ROM, 2 KByte 40polig, Standardversion	UB 8810 D		UD 8810 D
Interner ROM, 2 KByte 40polig, power-down-Version	UB 8811 D		UD 8811 D
externes Speicherport für 2KByte, 64polig Standardversion	UB 8820 M	UC 8820 M	UD 8820 M
externes Speicherport für 2KByte, 64polig power-down-Version	UB 8821 M	UC 8821 M	UD 8821 M
BASIC-Interpreter, Bootstrap-Lader, 40polig Standardversion	UB 8830 D	UC 8830 D	UD 8830 D
BASIC-Interpreter, Bootstrap-Lader, 40polig power-down-Version	UB 8831 D	UC 8831 D	UD 8831 D
kein interner ROM, 40polig Standardversion	UB 8860 D	UC 8860 D	UD 8860 D
kein interner ROM, 40polig power-down-Version	UB 8861 D	UC 8861 D	UD 8861 D
externes Speicherport für 4KByte, 64polig Standardversion	UB 8840 M	UC 8840 M	UD 8840 M
externes Speicherport für 4KByte, 64polig power-down-Version	UB 8841 M	UC 8841 M	UD 8841 M

In jüngster Zeit ist dieses Typenspektrum durch einen weiteren Typ ergänzt worden, bei dem oben vorgestellter Bezeichnungsschlüssel der internationalen Norm angeglichen wurde:

```
Ux 8611 DC - 08 - xxx
 I  I   II    I    I-------  Bitmuster des internen 4 KByte-ROM
 I  I   II    I------------  8 MHz externe Taktfrequenz
 I  I   II----  ------------  0 bis 70 °C Einsatzbereich
 I  I   I-------------------  40poliges DIL-Gehäuse
 I  I----------------------  internationale Typennummer
 I-------------------------  ohne = Standardversion
                               L = power-down-Version
```

Tabelle 1.1 Wesentliche Eigenschaften der EMR

2. Struktureller Aufbau der Einchipmikrorechner *U 881/U 882*

2.1. Wesentliche Parameter und Komponenten

Die Schaltkreise *U 881/U 882* sind 8-Bit-Einchipmikrorechner in n-Kanal-Silicon-Gate-Technologie. Ein leistungsfähiger Befehlssatz sowie kurze Befehlsausführungszeiten kennzeichnen diese Bauelemente. Folgende Eigenschaften sind für den Anwender wesentlich:

- Befehlssatz mit 47 Basisbefehlen,
- Registersatz mit 144 8-Bit-Registern; davon 124 Universalregister, 4 E/A-Register sowie 16 Spezialregister für die Initialisierung und Statusabfrage des Prozessors,
- 32 Ein-/Ausgabeleitungen, organisiert als 4 Tore zu 8 Bit,
- 2 Zähler/Zeitgeber-Kanäle mit einem Zählumfang von 8 Bit und 6-Bit-Vorteilern,
- Möglichkeit der asynchronen Datenübertragung durch interne SIO,
- leistungsfähige Interruptstruktur für die Behandlung von 6 Interruptanforderungen; die Interruptquellen arbeiten nach dem Vektorprinzip, sind maskierbar und bei zeitgleichem Auftreten priorisierbar,

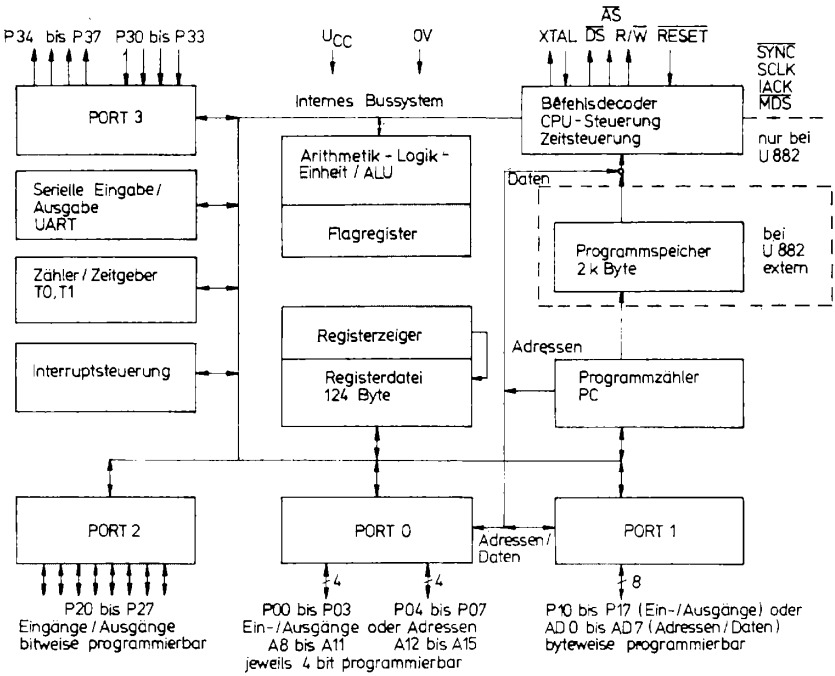


Bild 2.1 Übersichtsschaltbild der Einchipmikrorechner

- on-chip-Taktgenerator in der Standardversion für den direkten Anschluß eines Quarzes bzw. eines *LC*-Schwingkreises, maximale Taktfrequenz 8 MHz,
- getrennter Stromversorgungsanschluß für den Registersatz bei der power-down-Version,
- TTL-Kompatibilität an allen Ein-/Ausgängen bei einer TTL-Last,
- Versorgungsspannung 5 V, Stromaufnahme typisch 200 mA,
- 40poliges Gehäuse bei *U 881* und Ableittypen, 64poliges Gehäuse bei *U 882* und *U 884*.

Im Bild 2.1 ist das Übersichtsschaltbild der Einchipmikrorechner dargestellt.

2.2. Beschreibung einzelner Funktionskomplexe

CPU-Steuerung

Die Befehlsabarbeitung geschieht in bekannter Weise, indem der 16-Bit-Programmzähler (PC) den intern oder extern angeordneten Programmspeicher adressiert. Der gelesene Operationscode wird im Befehlsregister zwischengespeichert und durch den Befehlsdecoder ausgewertet. Die CPU-Steuerung veranlaßt die Befehlsausführung.

Sie beeinflußt insbesondere die Daten- und Adreßbussteuerung und gibt Statussignale an die Peripherie aus. Bei arithmetischen oder logischen Befehlen erfolgt die Verknüpfung der Operanden über die Arithmetik-Logik-Einheit (ALU). In Abhängigkeit vom Ergebnis werden 6 Bit innerhalb eines 8-Bit-Flagregisters verändert, die restlichen 2 Bit stehen dem Anwender zur Verfügung. Als Besonderheit verfügt der Einchipmikrorechner *U 881/882* über einen vorausschauenden Befehlslesemechanismus (sogenanntes Pipelining). Damit kann der nächste Befehl bereits während der Bearbeitung des vorhergehenden Befehls gelesen werden. Das führt zu einer weiteren Verkürzung der Programmlaufzeiten.

Der on-chip-Taktgenerator ist ebenfalls der CPU-Steuerung zuzuordnen. Die durch den Anschluß eines externen Quarzes erzeugte Frequenz wird um den Faktor 2 geteilt und bildet den Systemtakt. Die Einspeisung einer durch einen externen Generator erzeugten Taktfrequenz ist möglich, wobei der 2 : 1-Teiler dabei ebenfalls wirksam wird.

Registersatz

Der interne Registersatz umfaßt 144 8-Bit-Register. Funktionell werden 124 Universalregister, 4 Ein-/Ausgaberegister sowie 16 Status- und Steuerregister unterschieden. Die Universalregister können als RAM-Bereich aufgefaßt und als Operativspeicher, z. B. für die Abspeicherung von Zwischenergebnissen, genutzt werden. Jedes Register kann direkt mit der ALU zusammenarbeiten, d. h. Quelle oder Ziel von Operationen sein. Die Ein-/Ausgabetore 0 bis 3 werden programmtechnisch ebenfalls als Universalregister aufgefaßt. Spezielle Ein-/Ausgabebefehle für das Lesen oder Setzen von Ausgabeleitungen sind daher nicht vorhanden. Durch die Programmierung der 16 Steuer- und Statusregister lassen sich die Funktionen der Einchipmikrorechner vielfältig variieren und damit auf den konkreten Einsatzfall festlegen. Einige Steuerregister sind nicht lesbar und einige Statusregister nicht beschreibbar. Bild 2.2 zeigt den Aufbau des Registersatzes, die Register 180 bis 239 sind nicht vorhanden.

Die Adressierung der Register kann direkt oder indirekt über eine 8-Bit-Adresse erfolgen. Der Registersatz wird in 9 Gruppen zu je 16 Registern aufgeteilt. Innerhalb

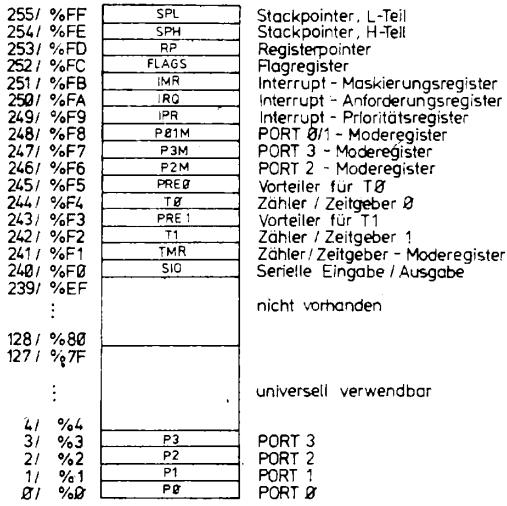


Bild 2.2
Registersatz

einer solchen Arbeitsregistergruppe liefern die entsprechenden Befehle die niederwertigen 4 Adreßbit, die oberen 4 Bit werden dem sogenannten Registerzeiger (RP, Register 253) entnommen. Die Vorteile dieser Adressierungsart bestehen in einer Verkürzung der Befehlslänge (meist ein Byte kürzer) sowie in einer Minimierung der Befehlsausführungsdauer. Diesen Vorteilen steht als möglicher Nachteil eine geringere Übersichtlichkeit der Programme gegenüber, falls im Maschinencode programmiert werden muß.

Interner und externer Speicher

Der 16-Bit-Programmzähler gestattet die unmittelbare Adressierung von 64 KByte Speicherraum. Der interne Programmspeicher belegt beim *U 881* die unteren 2 KByte (Adressen 0 bis 2047). Bei der Entwicklungsversion *U 882* ist der Programmspeicher extern anschließbar, jedoch ebenfalls als interner Speicher wie beim *U 882* zu behandeln. Der *U 884* verfügt über die Möglichkeit, 4 KByte internen Programmspeicher zu adressieren (Adressen 0 bis 4095).

Bei vollständiger Verwendung der Ein-/Ausgabetore 0 und 1 zur Adressen- und Datenausgabe sind beim *U 881/U 882* die verbleibenden 62 KByte Speicherraum nutzbar. Eine nochmalige Verdopplung des externen Speicherbereiches ist durch die Unter-

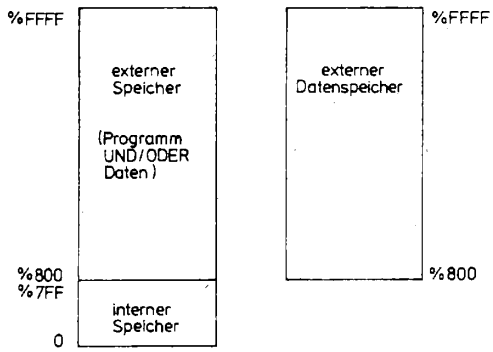


Bild 2.3
Speicherräume

scheidung in einen Daten- und einen Programmspeicher möglich. Das dazu notwendige Unterscheidungssignal /DM (data memory select, Datenspeicherauswahl) läßt sich bei entsprechender Programmierung am Tor 3 generieren. Der Zugriff auf beide Speicherräume erfolgt dann in Abhängigkeit vom jeweiligen Befehlscode. Wird die /DM-Unterscheidung nicht programmiert, wirken einige sonst nur auf einen der beiden Bereiche zugreifenden Befehle gleich. Die möglichen Speicherräume der *U881/U882* sind zusammenfassend im Bild 2.3 dargestellt.

Stack

Für die Rettung von Registerinhalten, Programmzählerstand und Flags bei Unterprogramm- oder Interruptbearbeitung läßt sich ein interner (in den Registern) oder externer (im RAM) Stack definieren, der über einen 16-Bit-Pointer (SP) adressiert wird. Ein interner Stack ist immer dann erforderlich, wenn ein externer RAM im Anwendungsfall nicht existiert. Die Zeigerfunktion wird dann nur über die niederwertigen 8 Bit des Pointers ausgeführt. Ein externer Stack kann beliebig im Adreßraum 2048 bis 65535 lokalisiert werden. Bei Verwendung des /DM-Signals muß der Stack ein Teil des Datenspeichers sein (/DM = L).

Ein-/Ausgabeleitungen

Die 32 Ein-/Ausgabeleitungen des *U881/U882* sind in 4 Tore (Ports) zu je 8 Bit zusammengefaßt. Die wesentlichen Eigenschaften dieser Tore sind:

Tor 0

Dieses Tor ist unterteilt in 2 Halbbytes von 4 Bit, wobei jedes Halbbyte einzeln als Adreßtor für den Anschluß externer Speicher oder als einfaches Ein-/Ausgabetor programmiert werden kann. In der Funktion als Adreßtor werden über das untere Halbbyte (P00 bis P03) die Adressen A8 bis A11 und über das obere Halbbyte (P04 bis P07) die Adressen A12 bis A15 ausgegeben. Da die Adreßleitungen A12 bis A15 nur benötigt werden, wenn mehr als 4 KByte externer Speicher zu adressieren sind, steht das obere Halbbyte in den meisten Fällen als Ein-/Ausgabetor zur Verfügung.

Tor 1

Die Festlegung der Funktion des Tores 1 ist nur für das gesamte Byte in gleicher Art möglich. Wird kein externer Speicher angeschlossen, sind die Leitungen P10 bis P17 gruppenweise als Ein- oder Ausgänge nutzbar. Ist dagegen ein externer Speicherbereich vorhanden, erfolgt über das Tor 1 die Ausgabe der Adressen A0 bis A7 und die Ein-/Ausgabe der Daten D0 bis D7 im Multiplexbetrieb. Der aktuelle Status am Tor 1 wird dann durch die Signale /AS (adress strobe) und /DS (data strobe) gekennzeichnet.

Tor 2

Die 8 Leitungen des Tores 2 sind einzeln als Ein- oder Ausgänge festlegbar. Im Falle der Nutzung als Ausgangsleitung kann in der Initialisierungsphase festgelegt werden, ob ein interner Lastwiderstand wirksam wird oder der Ausgang mit einer open-drain-Ausgangsstufe genutzt wird.

Tor 3

An diesem Tor sind jeweils 4 Leitungen fest als Eingang (P30 bis P33) und Ausgang (P34 bis P37) konfiguriert. Neben der möglichen universellen Nutzung sind die Leitungen von Tor 3 für zahlreiche Sonderfunktionen (s. Tabelle 2.1) programmierbar. Über die Leitungen P30 bis P33 können Interruptanforderungen von externen Baugruppen

E/A-Leitung	Signal	Verwendung
P30	IRQ3	Interruptauslösung mit HL-Flanke
	SIOin	Serielle Schnittstelle - Eingang
P31	IRQ2	Interruptauslösung mit HL-Flanke
-	Tin	externer Eingang Timer 1
	DAV2/RDY2	Handshakesignale für Tor 2
P32	IRQ0	Interruptauslösung mit HL-Flanke
	DAV0/RDY0	Handshakesignale für Tor 0
P33	IRQ1	Interruptauslösung mit HL-Flanke
	DAV1/RDY1	Handshakesignal für Tor 1
P34	RDY1/DAV1	Handshakesignal für Tor 1
	/DM	Steuersignal data memory select
P35	RDY0/DAV0	Handshakesignal für Tor 0
P36	RDY2/DAV2	Handshakesignal für Tor 2
	Tout	Ausgang von Timer 0 oder Timer 1
P37	SIOout	Serielle Schnittstelle - Ausgang

Tabelle 2.1 Steuerfunktionen des Tores 3

eingespeist werden, die konkrete Nutzung der einzelnen Leitungen wird im Steuerregister P3M festgelegt.

Handshake-Steuerung

Die Tore 0, 1 und 2 können über die Signale /DAV und RDY im Handshake-Modus arbeiten. Diese Betriebsart ermöglicht den Datenaustausch im Quittungsbetrieb zwischen einem Ein-/Ausgabetor und einem peripheren Gerät, das z. B. auch ein Einchipmikrorechner sein kann. Das Signal /DAV (data available, Daten verfügbar) ist L-aktiv. Seine Bedeutung hängt von der Richtung des Datenflusses ab. Gleiches gilt auch für das H-aktive Signal RDY (Sender bereit für Ausgabe bzw. Empfänger bereit zur Eingabe). Da die für den Handshake-Betrieb genutzten Eingänge von Tor 3 interruptfähig sind, kann der Datenaustausch auch interruptgesteuert erfolgen.

Handshake-Betriebsart Ausgabe

– Ausgangssignal /DAV,
– Eingangssignal RDY.

Für die Initialisierung des Quittungsbetriebes muß eine definierte Reihenfolge eingehalten werden:

- Programmierung des Tores bzw. der benötigten Bits auf Ausgabe,
- /DAV-Signal rücksetzen (auf H-Pegel, da L-aktiv),
- Quittungsbetrieb durch entsprechende Programmierung von P3M auswählen.

Für die Ausgabe der Daten sind folgende Schritte kennzeichnend:

- /DAV wird nach erfolgter Datenausgabe aktiv (L-Pegel), falls RDY ebenfalls aktiv ist und damit die Bereitschaft des Empfängers zur Datenübernahme kennzeichnet.
- Rücksetzen des RDY-Signals auf L-Pegel, wenn der Empfänger die Daten übernommen hat. Die dabei auftretende HL-Flanke kann im Sender zur Auslösung einer Interruptbehandlung zum Senden weiterer Daten genutzt werden.
- Rücksetzen des /DAV-Signals, da wegen RDY = L vom Empfänger keine neuen Daten gefordert sind.
- Anforderung neuer Daten vom Empfänger durch Setzen von RDY auf H.

Dieser Ablauf kann zyklisch fortgesetzt werden. Zu beachten ist allerdings, daß Ausgabeoperationen auf das im Quittungsbetrieb arbeitende Tor jederzeit möglich sind, was zu Übertragungsfehlern führen kann. Es muß also unbedingt eine programmtechnische Überwachung der Quittungssteuerleitungen erfolgen. Neben der besonders ef-

fektiven Nutzung des Interruptbetriebes kann das auch durch eine Abfrage der Signale am Tor 3 im Pollingbetrieb erfolgen.

Handshake-Betriebsart Eingabe

Bei der Eingabe im Handshakemodus sind beide Quittungsleitungen wie folgt definiert:

- Ausgabesignal RDY,
- Eingangssignal /DAV.

Auch hier muß bei der Initialisierung eine bestimmte Reihenfolge eingehalten werden:

- Programmierung des Tores bzw. der für den Datenaustausch benötigten Bits auf Eingabe,
- RDY-Signal setzen (H-Pegel), damit sofortige Eingabebereitschaft,
- Quittungsbetrieb durch entsprechende Programmierung von P3M (R 247) auswählen.

Die Dateneingabe läuft dann in folgenden Schritten ab:

- RDY-Signal zeigt die Empfangsbereitschaft des Einchipmikrorechners an (H-Pegel).
- Nachdem das Peripheriegerät (Sender) neue Daten bereitgestellt hat, setzt es das /DAV-Signal auf L-Pegel und zeigt damit die Verfügbarkeit der Daten an. Die HL-Flanke des /DAV-Signals bewirkt die Übernahme der Daten in das Torregister. Eine gleichzeitige Interruptanforderung kann für die weitere Verarbeitung der Eingabedaten genutzt werden.
- Nach der Übernahme durch den Einchipmikrorechner wird RDY zurückgesetzt (L-Pegel), um dem sendenden Peripheriegerät anzuzeigen, daß keine Empfangsbereitschaft mehr vorliegt.
- Nach dem Rücksetzen des /DAV-Signals vom Peripheriegerät (H-Pegel) bewirkt eine Leseoperation vom Eingabetor die Aktivierung des RDY-Signals, womit die Bereitschaft zum Empfang neuer Daten angezeigt wird.

Es ist zu beachten, daß mit Sicherheit gültige Eingabedaten nur im Moment der HL-Flanke des /DAV-Signals vorliegen. Demnach muß eine Überwachung dieses Signals erfolgen, wobei der Interruptbetrieb ebenfalls die effektivste Verfahrensweise darstellt.

Die Quittungsleitungen können als weitere Methode für eine getaktete Ein-/Ausgabe verwendet werden. Bei Eingabe bleibt dazu RDY ungenutzt, und die Übernahme der Daten in das Torregister geschieht mit der HL-Flanke am /DAV-Eingang. Der Abstand von 2 aufeinanderfolgenden HL-Flanken muß in jedem Fall so groß sein, daß das Lesen der Daten vom Eingaberegister noch sicher gewährleistet ist. Bei Ausgabe müssen /DAV und RDY zusammengeschaltet werden, damit bei jeder Ausgabeoperation ein kurzer L-Impuls entsteht und empfangsseitig eine Datenübernahme erfolgen kann.

Zähler/Zeitgeber

Die Einchipmikrorechner enthalten 2 8-Bit-Zähler/Zeitgeber mit vorgeschalteten 6-Bit-Teilern. Ähnlichkeiten zur CTC *U 857* des *U 880*-Systems sind vorhanden, jedoch ist hinsichtlich der Anwendung durch zahlreiche Programmiermöglichkeiten eine noch größere Flexibilität gegeben. Beide Kanäle sind gleichartig aufgebaut, ein Unterschied besteht aber im Ansteuerverhalten mit Zählimpulsen. Während Kanal T0 nur mit einer aus dem internen Systemtakt abgeleiteten Frequenz angesteuert werden kann, ist bei Kanal T1 auch die Einspeisung einer externen Impulsfolge möglich. Diese Einspeisung

erfolgt über P31 bei entsprechender Programmierung des Betriebsartenregisters TMR (R 241). Die Leitung P31 kann außerdem als Tor- oder Triggereingang benutzt werden.

Der 8-Bit-Rückwärtszähler jedes Kanals erzeugt bei Nulldurchgang eine Übertragsinformation, die bei entsprechender Programmierung eine Interruptanforderung auslösen kann. Die Ausgabe dieses Übertrags ist für einen der beiden Kanäle an Leitung P36 möglich. An diesem Ausgang wird bei Nulldurchgang des Zählers ein Pegelwechsel vollzogen, so daß bei kontinuierlicher Arbeitsweise des Rückwärtszählers eine Rechteckimpulsfolge mit symmetrischem Tastverhältnis entsteht. Bei Kaskadierung beider Zähler/Zeitgeber-Kanäle ist diese ausgangsseitige 2:1-Teilung mit in die Berechnung einzubeziehen. Soll ein 16-Bit-Zähler durch Kaskadierung von T0 und T1 aufgebaut werden, so ist die direkte Verbindung von Ausgang P36 mit Eingang P31 nicht möglich. Die dazu notwendige Frequenzverdopplung der P36-Impulse muß durch eine externe Schaltung realisiert werden. In den meisten Fällen ist es daher günstiger, eine Erweiterung des Zählumfangs auf 16 Bit durch die Nutzung der Interruptanforderung des Zählers zu realisieren. In der Interruptbehandlung wird dann ein Register inkrementiert, dessen Inhalt dem höherwertigen Teil des Zählerstandes entspricht.

Der 6-Bit-Vorteiler jedes Kanals hat ein nur beschreibbares Laderegister (PRE0, PRE1), während die 8-Bit-Rückwärtszähler (T0, T1) als Schreib-Lese-Register ausgeführt sind. Diese Register sind einerseits mit einem Anfangswert ladbar (Wert 0 bis 255), andererseits ist jederzeit der aktuelle Zählerstand lesbar. Die Festlegung der Betriebsart der Kanäle erfolgt durch Programmierung der Register TMR, PRE0, PRE1 und P3M.

Der 8-Bit-Zähler eines Kanals kann jederzeit gestartet, gelesen oder gestoppt werden. Ebenso ist das Laden des Anfangswertes ohne Einschränkungen möglich. Beim Erreichen des Zählerstandes 0 bestehen in Abhängigkeit von der gewählten Betriebsart im wesentlichen 2 Möglichkeiten:

- *single pass mode*: Die Zählerfunktion wird nach Erreichen des Zählerstandes 0 gesperrt. Ein erneutes Anlaufen des Zählers muß durch entsprechende Steuerbefehle ausgelöst werden.
- *continous mode*: Bei Nulldurchgang des Zählers wird die im Laderegister eingestellte Zeitkonstante automatisch nachgeladen und der Zählvorgang fortgesetzt. Ein Verändern des Laderegisterinhaltes während des laufenden Zählvorgangs ist möglich, der neue Wert wird beim nachfolgenden Nulldurchgang in den Rückwärtszähler übernommen.

Serielle Schnittstelle (UART)

Die asynchrone serielle Schnittstelle der Einchipmikrorechner *U 881/U 882* stellt eine günstige Variante der Kopplung mit anderen Rechnersystemen dar. Die Schnittstelle arbeitet im Vollduplex-Betrieb, die Leitung P30 ist dabei der Dateneingang und P37 der Datenausgang. Das serielle Datenregister SIO (R 240) wird gleichzeitig als Sende- und Empfangsregister benutzt, Lesebefehle werden nur auf das Empfangsregister und Schreibbefehle nur auf das Senderegister wirksam.

Die Serien-Parallel-Wandlung beim Empfang über P30 erfolgt durch ein dem SIO-Register vorgelagertes Schieberegister. Nachdem dieses gefüllt ist, werden die Daten im Empfangsregister abgelegt. Gleichzeitig wird eine Interruptanforderung ausgelöst, falls sinnvollerweise eine entsprechende Freigabe erteilt ist. Mit der zugeordneten Interruptroutine kann ein Transport der empfangenen Daten in das Zielregister erfolgen.

Da gegen das Überschreiben von Daten, die im Empfangsregister bereitstehen und noch nicht gelesen worden sind, kein Schutz besteht, muß die Leseoperation unmittelbar nach der Interruptanforderung erfolgen. Umfangreichere Auswertungsprogramme sollten deshalb nicht in der Interruptbehandlung selbst erfolgen, sondern zu einem günstigeren Zeitpunkt der Programmbearbeitung ausgeführt werden. Die Auswertung der empfangenen Daten im Pollingbetrieb ist prinzipiell möglich, aber nur in den wenigsten Anwendungsfällen sinnvoller als eine Interruptauswertung.

Der Sendebetrieb wird mit dem Beschreiben des SIO-Registers ausgelöst. Mit einem Schreibbefehl auf das SIO-Register erfolgt die Übernahme der Daten in das Sendeschieberegister und die serielle Datenausgabe an P37. Nach dem Senden eines Bytes wird bei entsprechender Freigabe eine Interruptanforderung ausgelöst. Das zugehörige Behandlungsprogramm bewirkt danach das Laden neuer Daten in das Senderegister. Wird das Senderegister geladen, bevor das Schieberegister die Parallel-Serien-Wandlung eines Datenbytes beendet hat, kommt es zum Überschreiben der Sendedaten und damit zu Übertragungsfehlern. Dieser Fehler kann nur bei der Datenübertragung im Pollingbetrieb auftreten, die Nutzung des Interruptbetriebes schließt diesen Fehler von vornherein aus.

Der Betrieb der seriellen Schnittstelle ist nur in Zusammenhang mit dem Zeitgeberkanal T0 möglich, dessen Zählintervall die Bitrate der Übertragung festlegt. Der genaue Wert der Übertragungsrate ist im Prinzip willkürlich festlegbar. Eine Übereinstimmung zwischen Sender und Empfänger ist jedoch Voraussetzung für eine Kopplung von Geräten. Dazu haben sich in der Informationsübertragung standardisierte Baudraten (Bits/Sekunde) herausgebildet, die bei einem Datenaustausch mit industriellen Geräten eingehalten werden müssen.

Die allgemein üblichen seriellen Übertragungsarten, wie sie z. B. vom SIO-Schaltkreis des Systems *U 880* bekannt sind, lassen sich mit der seriellen Schnittstelle des Einchipmikrorechners nur eingeschränkt realisieren. So ist die Übertragung nur asynchron möglich, es werden ein Startbit, 2 Stoppbits und 8 Datenbits je Sendezeichen benutzt. Falls über das Steuerregister R 247 eine Übertragung mit Parität (nur ungerade Parität wählbar) festgelegt wird, ist die Länge des eigentlichen Datenwortes auf 7 Bit reduziert. In die höchstwertige Bitstelle (D7) wird in diesem Fall das Paritätsbit eingetragen. Eine ausführliche Beschreibung der seriellen Schnittstelle erfolgt im Abschnitt 8.

Interruptsteuerung

Die Einchipmikrorechner *U 881/U 882* lassen ohne zusätzlichen Hardwareaufwand die Bearbeitung von 6 Interrupts (IRQ0 bis IRQ5) aus insgesamt 8 Interruptquellen zu. Die Abarbeitung der zugehörigen Interrupt-Bedienprogramme erfolgt ähnlich wie im System *U 880* vektorisiert. Die Interruptstartadressen stehen immer am Anfang des Programmspeichers auf den Adressen 0 bis 11. Den einzelnen Interruptanforderungen sind dazu je 16 Bit, wie im Bild 2.3 dargestellt, zugeordnet. Die Abarbeitung des eigentlichen Anwender-Hauptprogramms kann demzufolge nach RESET erst ab Adresse 12 erfolgen. Der Programmzähler wird deshalb bei RESET automatisch auf diesen Wert voreingestellt.

Bei den Schaltkreistypen *U 883* bzw. *U 886* sind die Interruptstartadressen fest mit den ersten 18 Speicherzellen des externen Programmspeichers (ab Adresse 2048) programmiert, wobei für jede Interruptanforderung 3 Byte zur Eintragung eines Sprung-

befehls reserviert sind. Damit können die zugehörigen Interrupt-Behandlungsprogramme im gesamten Programmspeicherraum plaziert werden.

Die Sperrung oder Freigabe der Interruptanmeldungen ist mit Hilfe des Interrupt-Maskierungsregisters IMR (R 251) möglich. Jeder Interruptanforderung ist ein Bit in diesem Register zugeordnet, das gesetzt (Freigabe) oder rückgesetzt (Sperrung) werden kann. Alle Interrupts werden außerdem dann gesperrt, wenn Bit 7 des IMR-Registers rückgesetzt wird.

Die Interruptanmeldungen der 6 Interruptquellen werden im Interrupt-Anmelderegister IRQ (R 250) abgespeichert. In ähnlicher Art wie beim IMR-Register ist jeder Quelle ein Bit fest zugeordnet. Über diese Bits ist neben der üblichen Interrupt-Behandlung auch eine Abfrage von Anmeldungen im Polling möglich.

Falls es zur gleichzeitgen Interruptanforderung aus mehreren Quellen kommt, wird die Reihenfolge ihrer Bearbeitung durch das Interrupt-Prioritätsregister IPR (R 249) festgelegt. Neben der Beeinflussung des Interruptverhaltens über die genannten Steuerregister ist die globale Sperrung und Freigabe mit den Befehlen DI (disable interrupt) und EI (enable interrupt) möglich.

3. Anschlußbelegung und -beschreibung

Bild 3.1 zeigt die Anschlußbelegung beider Schaltkreise. Gleichartig bezeichnete Anschlüsse sind auch funktionell identisch. Aufgrund der beiden Gehäusebauformen sind die Anschlußbelegungen natürlich unterschiedlich. Eine Reihe von Anschlüssen existiert nur bei der 64poligen Entwicklungsversion *U 882*. Neben dem herausgeführten Speichertor handelt es sich dabei um zusätzliche Status- und Steuersignale, deren Auswertung speziell bei der Programmentwicklung und im Testfall von Interesse ist.

Die Anschlüsse haben im einzelnen folgende Funktion:

P00 bis P07, Ein-/Ausgabetor 0; P20 bis P27, Ein-/Ausgabetor 2;
P10 bis P17, Ein-/Ausgabetor 1; P30 bis P37, Ein-/Ausgabetor 3.

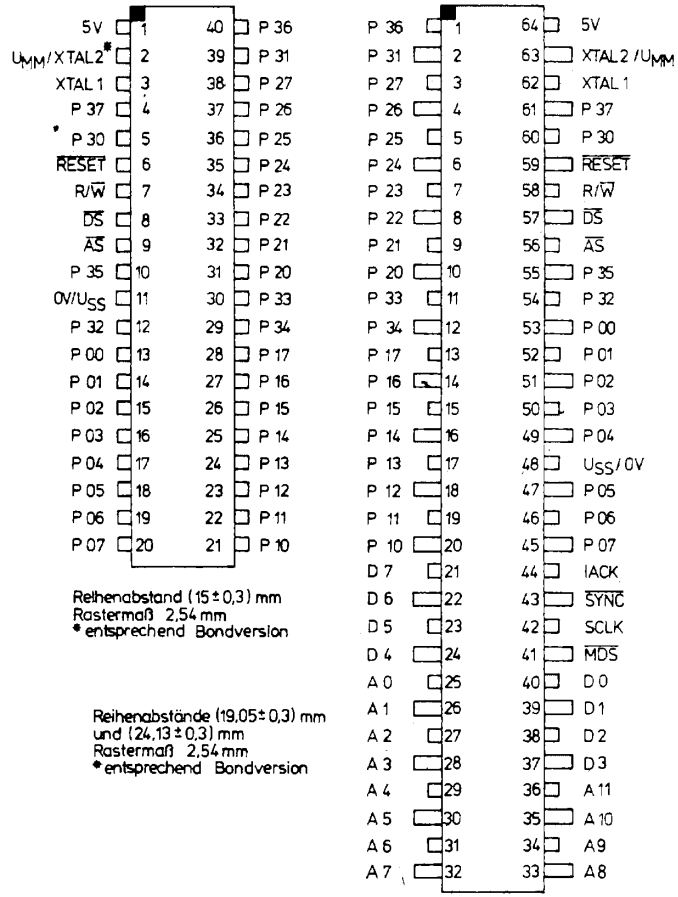


Bild 3.1
Anschlußübersicht und Anschlußbelegung der EMR

/RESET Rücksetzeingang, L-aktiv
Das Rücksetzsignal bringt den Prozessor in einen definierten Anfangszustand. Der Programmzähler wird auf Adresse 12 gesetzt, auf der das 1. Byte des Anwenderprogramms stehen muß. Die Steuerregister werden mit einer definierten Anfangsbelegung versehen. Da der /RESET-Eingang über einen internen pull-up-Widerstand von etwa 100 kΩ auf H-Potential gezogen wird, besteht die externe Beschaltung im einfachsten Fall aus einem Kondensator von etwa 1 µF.

Durch eine Spannungsüberhöhung am Rücksetzeingang auf etwa 8 V läßt sich beim *U 881* ein sogenannter Testmodus einschalten. Dabei werden die unteren 64 Speicherzellen des internen Programmspeichers durch Testbefehle ersetzt. Diese Betriebsart war ursprünglich für Testzwecke des Herstellers vorgesehen, kann aber beim *U 886* für einen Absprung in den externen Programmspeicher genutzt werden. Für den *U 882* existiert der Testmodus nicht, weil angesichts des frei zugänglichen Programmspeichers dafür keine Notwendigkeit besteht.

Der /RESET-Impuls sollte eine Dauer von 50 ms haben. Vom Hersteller wird eine Dauer von mindestens 18 Taktzyklen nach Anliegen der Betriebsspannung und Einschwingen des Taktgenerators gefordert.

Bei aktivem /RESET-Signal wird über das Signal /AS der interne Systemtakt ausgegeben, das Signal /DS schaltet auf L-Pegel, und R/W nimmt H-Pegel an. Da außerhalb des RESET-Zustands /AS und /DS niemals gleichzeitig L-Pegel führen, läßt sich durch entsprechende logische Verknüpfung ein Signal bilden, das peripheren Baugruppen den RESET-Status anzeigt.

Nach einem RESET sind die Tore P0, P1 und P2 auf die Betriebsart Eingabe initialisiert, die Ausgänge von Tor P3 bleiben auch während der Dauer des RESET-Impulses aktiv.

/AS adress strobe (Ausgang, tristate, L-aktiv)
Dieses Signal wird beim Anschluß externer Speicherbauelemente benötigt, um den aktuellen Zustand auf dem gemultiplexten Adreß-/Datenbus anzuzeigen. Eine Aktivierung erfolgt grundsätzlich bei Beginn eines Maschinenzyklus. Bei allen externen Speicheroperationen sind gültige Adreßinformationen an Tor P1 mit der Rückflanke (LH-Flanke) des Signals /AS vorhanden. /AS kann damit zur direkten Ansteuerung des benötigten Adreßpuffers genutzt werden. Durch entsprechende Programmierung des Steuerregisters R 248 ist der hochohmige Zustand (tristate) an der Ausgangsleitung /AS erreichbar.

/DS data strobe (Ausgang, tristate, L-aktiv)
Das Signal /DS kennzeichnet ebenfalls den aktuellen Status auf dem gemultiplexten Adreß-/Datenbus. Bei allen externen Speicheroperationen zeigt es das Vorhandensein gültiger Daten an (Daten gültig mit /DS = L). Beim Speicher-Lesezyklus werden die Daten mit der Rückflanke (LH-Flanke) des /DS-Signals übernommen. Der hochohmige Zustand von /DS ist wieder softwaremäßig einstellbar. Wenn das Tor 1 nicht für gemultiplexte Adreß-/Datenausgabe programmiert wird, erfolgt an /DS die Ausgabe des Signals /SYNC, das von seiner Funktion her etwa dem /M1-Signal des Prozessors *U 880* entspricht und während eines Operationscode-Lesezyklus aktiv ist.

R/W read/write (Ausgang, tristate)
Das Signal R/W wird mit dem entsprechenden Anschluß externer Schreib-Lese-Speicher verbunden. R/W = L kennzeichnet die Datenausgabe, R/W = H die Dateneingabe über den Multiplexbus an Tor 1 bei Zugriffsoperationen zum externen Speicher. Die Hochohmigkeit läßt sich auch bei diesem Signal durch entsprechende Programmierung des Registers R 248 erreichen.

XTAL1 crystal 1 (Taktgenerator, Eingang)
Bei der Standardbeschaltung wird zwischen XTAL1 und XTAL2 ein Schwingquarz angeschlossen. Die maximale Taktfrequenz beträgt 8 MHz in der Version *UB 881/882.* Anstelle des Quarzes ist auch die Beschaltung mit einer *LC*- oder *RC*- Kombination möglich, natürlich auf Kosten der Stabilität, aber mit Preisvorteilen.

Über XTAL1 ist auch ein externer Takt mit TTL-Pegel einspeisbar, in der power-down-Version der Prozessoren ist das sogar zwingend erforderlich, da dort XTAL2 für andere Zwecke genutzt wird.

XTAL2 crystal 2 (Taktgenerator, Ausgang)
Während XTAL2 in der Standardversion mit einem Anschluß des Schwingquarzes verbunden ist, wird in der power-down-Version eine Stützspannung für den internen Registersatz eingespeist. Damit ist ein Datenerhalt bei Ausfall der Grundstromversorgung des Prozessors möglich.

/MDS program memory data strobe (nur bei *U882*, Ausgang, L-aktiv)
Das Signal wird nur bei Befehlsholezyklen im «internen» Programmspeicher aktiv, d.h. im Adreßbereich 0 bis 2047. Es kann für Testzwecke verwendet werden.

/SYNC instruction syncronisation (nur bei *U882*, Ausgang, L-aktiv)
Das Statussignal /SYNC kennzeichnet den 1. Befehlsholezyklus einer Operation. Es wird dazu in der dem entsprechenden Maschinenzyklus vorangehenden Taktperiode aktiviert. /SYNC ist in Testsystemen (Emulatoren) zur Realisierung eines Befehls-Schrittbetriebs verwendbar.

IACK interrupt acknowledge (nur bei *U882*, Ausgang, H-aktiv)
Das Signal meldet die Quittierung einer Interruptanforderung durch den Prozessor und bleibt während des gesamten Quittierungszyklus aktiv.

SCLK system clock (nur bei *U882*, Ausgang)
Über SCLK ist der interne Systemtakt verfügbar. Er entspricht der halben Quarzfrequenz bzw. der halben Frequenz des extern eingespeisten Taktes.

D0 bis D7 program data (nur bei *U882*, Eingänge)
An D0 bis D7 werden die Datenleitungen des Programmspeichers für die unteren 2 KByte (Adressen 0 bis 2047) angeschlossen. Da es sich hierbei um den nur in der Entwicklungsversion (*U882*) zugänglichen «inneren» Speicherbereich handelt, ist ein erweitertes Speicherzeitverhalten nicht programmierbar.

A0 bis A11 program memory adress (nur bei *U882*, Ausgänge)
Die Adreßleitungen A0 bis A10 sind direkt mit dem Programmspeicher für die unteren 2 KByte des Adreßraumes verbunden. Im Standardfall kommt dafür ein EPROM *U2716* zur Anwendung. Der Anschluß A11 ist beim *U882* nicht beschaltet, das entsprechende Adreßbit wird nur beim *U884* mit 4 KByte «internem» Programmspeicher genutzt.

4. Zeitverhalten

Pipelining-Befehlsverarbeitung

Bei den Einchipmikrorechnern werden kurze Programmlaufzeiten nicht nur durch den effektiven Befehlssatz erreicht, sondern auch durch die sogenannte Pipelining-Befehlsverarbeitung. Dabei findet eine zeitliche Überlappung von Befehlshole- und Ausführungszyklus statt. Der Operationscode des nachfolgenden Befehls wird bereits gelesen, während die Ausführung des aktuellen Befehls noch läuft. Im Bild 4.1 ist das Prinzip der Pipelining-Befehlsverarbeitung dargestellt.

Für den Anwender ergeben sich aus dem genannten Prinzip keine Konsequenzen, bei der Berechnung von Programmlaufzeiten muß die genannte Überlappung allerdings berücksichtigt werden.

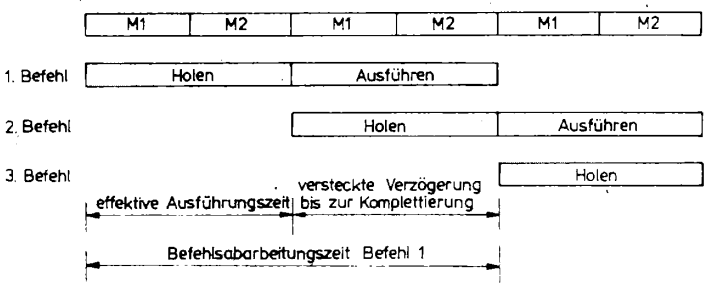
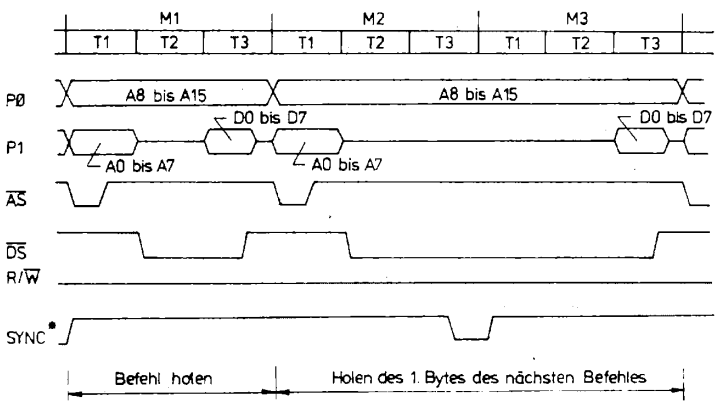


Bild 4.1
Prinzip der Pipelining-Befehlsverarbeitung

Befehlszyklus-Zeitverläufe

Das Bild 4.2 zeigt den Befehlszyklus-Zeitverlauf für das Holen eines Befehls vom externen Speicher (1-Byte-Befehl). Die vom Tor 0 gelieferten Adressen bleiben während eines Maschinenzyklus konstant, während die über das Tor 1 ausgegebenen Adressen



• Signal nur bei U882 vorhanden

Bild 4.2
Befehlszyklus-Zeitverlauf für Holen des 1. Bytes

A0 bis A7 nur in der T1-Phase verfügbar sind. Mit der Rückflanke (LH-Flanke) von /AS ist die Gültigkeit der Adressen garantiert, daher sollte die Übernahme in ein Adreßlatch mit dieser Flanke erfolgen. Am Ende der T1-Phase wird Tor 1 auf Eingabe geschaltet, um Daten empfangen zu können. Das Lesen der Daten erfolgt bei aktivem Signal /DS in der Phase T3.

Alle Befehlsholezyklen haben beim *U881/U882* dieselben Maschinenzeitverläufe bei internem oder externem Speicher. Eine Ausnahme hinsichtlich der Zeitverläufe besteht während des Operationscodeholens eines Befehls, der auf einen 1-Byte-Befehl folgt. Der 1-Byte-Befehl erfordert 2 Maschinenzyklen, das Pipelining-Verfahren ermöglicht jedoch im 2. Teil das Lesen des folgenden Operationscodes. Bild 4.3 zeigt die Befehlszyklus-Zeitverläufe für 2- und 3-Byte-Befehle.

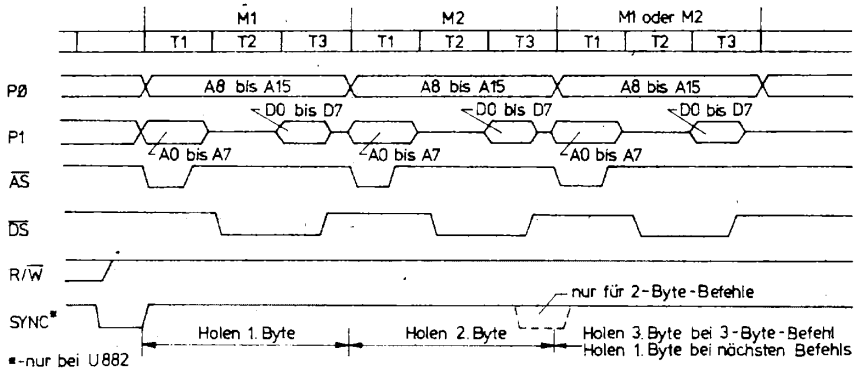


Bild 4.3 Befehlszyklus-Zeitverlauf bei 2- und 3-Byte-Befehlen

Zugriff auf externe Speicher

Auf einen externen Speicher kann bei entsprechender Konfiguration der Tore 0 und 1 zugegriffen werden. Eine Unterscheidung in einen Daten- und Programmspeicherbereich ist durch das /DM-Signal wahlweise möglich. Im Bild 4.4 sind die Zeitverläufe für einen Speicherlese- und Speicherschreibzyklus dargestellt. Die Abbildungen gelten für normales Speicherzeitverhalten. Im Falle der Programmierung eines erweiterten Speicherzeitverhaltens wird zwischen den Taktzuständen T2 und T3 ein zusätzlicher Takt TW eingefügt.

Der Zeitverlauf beim Speicherlesen unterscheidet sich prinzipiell nicht vom Befehlscode-Holezyklus. Die Datenübernahme erfolgt in der ersten Hälfte von T3, also 1,5 Taktperioden nach Aktivierung von /DS. Bei erweitertem Zeitverhalten vergrößert sich diese Zeit auf 2,5 Perioden des Systemtakts.

Beim Schreiben in externe Speicher bestehen hinsichtlich der Adressenausgabe keine Unterschiede zum Lesezyklus. Tor 1 verbleibt jedoch nach der Adressenausgabe in Ausgabestellung. Die Bereitstellung der Daten erfolgt mit Beginn des Taktzustands T2. Da das Signal /DS erst in der 2. Hälfte von T2 aktiv wird, sind die Daten bereits mit der HL-Flanke von /DS gültig.

Speicherzeitverhalten beim Anschluß externer RAMs

Bei der Nutzung externer RAMs zur Datenspeicherung sind die typgebundenen minimalen Zugriffszeiten t_{acc} zu den Speicherschaltkreisen zu beachten. Je nach interner

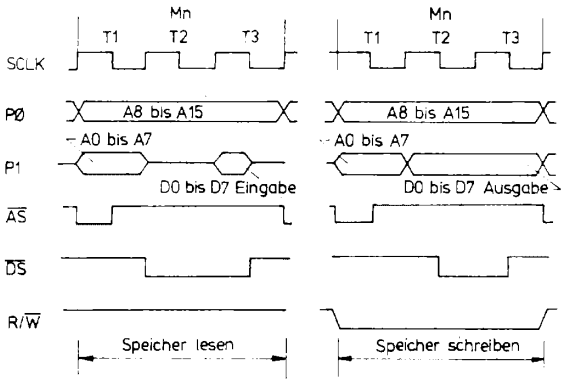


Bild 4.4
Speicherlese- und Schreibzyklus

Organisation der Speichermatrix und des Zugriffsverfahrens sind 2 Gruppen von RAMs zu unterscheiden: adressenaktivierte und chipaktivierte Speicher.

Bei adressenaktivierten Speichern beginnt der Speicherzugriffszyklus mit dem Wechsel der Information auf den Adreßleitungen. Da zur Ansteuerung der RAMs über den gemultiplexten Tor-1-Bus meist ein Adreßlatch (*DS 8212, DS 8282,* usw.) genutzt wird, werden die Wechsel auf den Adreßleitungen mit der LH-Flanke von /AS gültig an die RAMs weitergegeben. Damit ergibt sich bei adressenaktiven Speichern eine Zugriffszeit

$$t_{acc} = n \cdot t_{pc} - t_d.$$

t_{pc} – Periodendauer externer Systemtakt,
t_d – Verzögerungszeit Adreßlatch,
n = 3; normales Zeitverhalten,
n = 5; erweitertes Zeitverhalten.
Bei einer Systemtaktfrequenz von 8 MHz und normalem Zeitverhalten ergibt sich z. B.:

$$t_{acc} = 3 \cdot 125\ ns - 25\ ns = 350\ ns.$$

Ein typischer Vertreter der adressenaktivierten Speicher ist der *U 214.*

Bei chipaktiven Speichernbeginntder Speicherzugriffszyklus mit dem Anliegen eines aktiven Chip-Select am Speicherschaltkreis. Dieses Signal wird üblicherweise durch die HL-Flanke von /DS ausgelöst. Es ergibt sich eine Zugriffszeit:

$$t_{acc} = n \cdot t_{pc} - t_d.$$

t_{pc} – Periodendauer externer Systemtakt,
t_d – Verzögerungszeit /CS-Decoder,
n = 2; normales Zeitverhalten,
n = 3; erweitertes Zeitverhalten.
Bei einer Systemtaktfrequenz von 8 MHz und normalem Zeitverhalten ergibt sich daraus:

$$t_{acc} = 2 \cdot 125\ ns - 25\ ns = 225\ ns.$$

Ein Beispiel für die chipaktiven Speicher ist der CMOS-Speicher *U 224.* Da dieser Schaltkreis höchstens ab einer Zugriffszeit größer 350 ns (*U 224 D35*) verfügbar ist,

muß bei einer Frequenz f_{XTAL} = 8 MHz immer mit erweitertem Zeitverhalten gearbeitet werden.

Zugriffe auf den internen Speicher

Bei Zugriff auf die Speicherzellen mit den Adressen 0 bis 2047 sind die Zeitverläufe weitgehend mit den für den externen Speicherbereich angegebenen Diagrammen identisch, bis auf die nicht vorhandene Wahlmöglichkeit eines erweiterten Zeitverhaltens. Neben Operationscode-Holezyklen sind auch im internen Bereich Lesezugriffe möglich. Die Steuersignale /DS und R/W führen bei Zugriff auf den internen Bereich immer H-Pegel, sind also inaktiv. Die Speicheradresse ist bei entsprechender Konfiguration auch bei Operationen im internen Bereich an den Toren 0 und 1 verfügbar, /AS kennzeichnet in gewohnter Weise das Vorliegen gültiger Adressen. Bild 4.5 zeigt die entsprechenden Zeitverläufe.

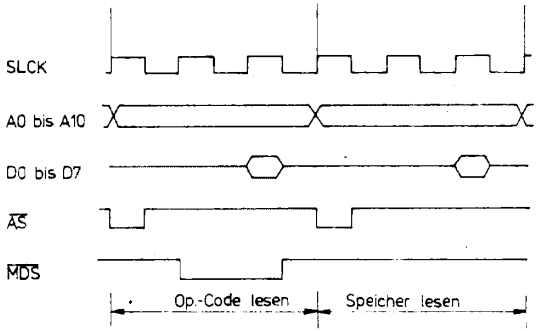


Bild 4.5
Zugriff auf den internen Speicher

Interrupt

Vor jedem Befehlsholezyklus wird getestet, ob Interrupt-Anforderungen vorliegen. Der Interrupt-Quittierungszyklus umfaßt einschließlich Testphase mindestens 24 interne Taktperioden. Es muß zwischen sogenannten externen und internen Interrupts unterschieden werden. Externe Interrupts sind bei entsprechender Freigabe durch HL-Flanken an den Toreingängen P30 bis P33 auslösbar. Sie werden bereits 2 Taktperioden vor jedem Befehlscode-Holezyklus abgefragt. Interne Interruptanforderungen durch die Zähler/Zeitgeber bzw. die serielle Schnittstelle werden erst eine halbe Taktperiode vor dem M1-Zyklus abgefragt.

Ist der Test auf Interruptanforderung positiv, wird der begonnene Befehlszyklus noch ausgeführt, die Daten werden aber nicht ausgewertet. Während der nachfolgenden Maschinenzyklen M2, M3 und M4 erfolgen die internen Aktivitäten der CPU. Es handelt sich dabei um die Ermittlung der Interruptpriorität, das Lesen des Inter-

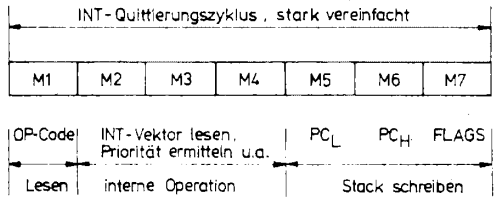


Bild 4.6
Interrupt-Quittierungszyklus

ruptvektors, das Setzen des Programmzählers auf die Anfangsadresse des Interrupt-Behandlungsprogramms sowie die Beeinflußung des Interrupt-Maskierungsregisters. Die folgenden Maschinenzyklen M4 bis M7 werden für die Rettung des alten PC-Zählerstands sowie des Flag-Registers in den Stack benötigt. Die Abspeicherung beginnt mit dem L-Teil des alten PC-Zählerstands, zuletzt erfolgt die Rettung des Flag-Registers. Der Interrupt-Quittierungszyklus ist stark vereinfacht in Bild 4.6 dargestellt.

5. Status- und Steuerregister

5.1. Funktionsbeschreibung

Die Register 240 bis 255 bilden die Gruppe der Status- und Steuerregister. Die einzelnen Funktionen dieser Register werden nachfolgend beschrieben.

R248 (%F8) P01M – Tor-0- und 1-Betriebsart
Dieses Register legt die Betriebsart der Tore 0 und 1 fest, definiert den Stack als intern (in Registern) oder extern (im RAM) und ermöglicht ein erweitertes Taktregime bei Zugriff auf externe Speicher.

```
-------------------------------
D7  D6  D5  D4  D3  D2  D1  D0
-------------------------------
I   I   I   I   I   I   I   I      P00 bis P03
I   I   I   I   I   I   I---I----  00 Ausgang
I   I   I   I   I   I              01 Eingang
I   I   I   I   I   I              1x Ausgabe der Adressen A8 bis A11
I   I   I   I   I   I
I   I   I   I   I   I------------  Stack
I   I   I   I   I                  0  extern
I   I   I   I   I                  1  intern
I   I   I   I   I
I   I   I   I---I----------------  P10 bis P17
I   I   I                          00 Ausgang
I   I   I                          01 Eingang
I   I   I                          10 Multiplexbus AD0 bis AD7
I   I   I                          11 hochohmig
I   I   I
I   I   I                          Zeitverhalten
I   I   I------------------------  0  normal
I   I                              1  erweitert
I   I
I   I                              P04 bis P07
I---I------------------------------  00 Ausgang
                                   01 Eingang
                                   1x Ausgabe Adressen A12 bis A15
-------------------------------
```

Beim *U882* bzw. *U884* sind die Tore 0 und 1 nach einem RESET auf Eingabe festgelegt, der Stack ist intern definiert und normales Zeitverhalten eingestellt.

Um beim *U883* einen Sprung in den externen Speicherbereich zu ermöglichen, wird nach RESET die gemultiplexte Adreß-/Datenausgabe am Tor 1 programmiert, über Tor 0 werden die Adressen A8 bis A15 ausgegeben, und es wird das erweiterte Zeitverhalten in der Ansteuerung externer Speicher eingestellt.

Das Bit D7 des Registers 248 ist dominant, d. h., falls dieses Bit gesetzt ist, werden am Tor 0 die Adressen A8 bis A15 ausgegeben, unabhängig von der Belegung der Bits D0 und D1.

Bei gesetztem Bit D2 wird der Stack innerhalb der Universalregister über das Register 255 (SPL) lokalisiert. Das Register 254 (SPH) bleibt in diesem Fall für die allgemeine Nutzung frei. Bei rückgesetztem D2 ist der Stack im externen Speicher lokalisiert. Für die Adressierung wird dann der vollständige 16-Bit-Stackpointer (SPL und SPH) benötigt.

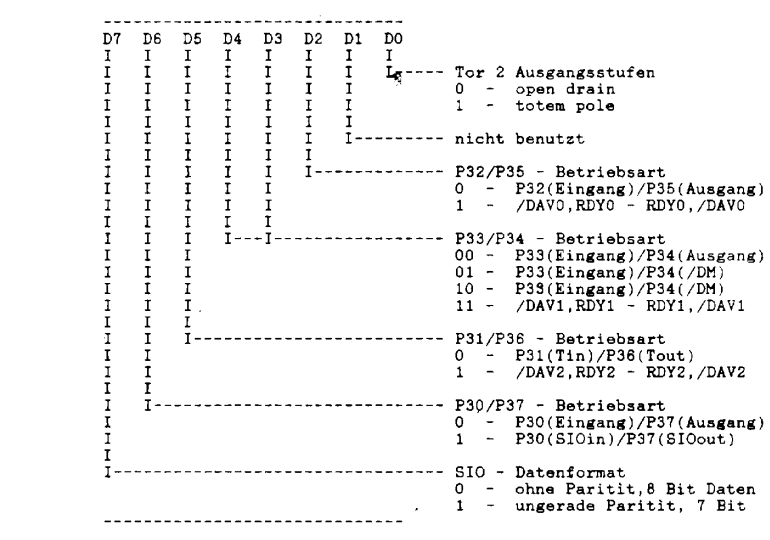
Mit D5 wird das Speicherzeitverhalten festgelegt. Ist dieses Bit gesetzt, so wird der Speicherzugriff auf den externen Speicher um eine Taktperiode erweitert, um gegebenenfalls auch langsamere Speicherschaltkreise verwenden zu können.

R 246 (%F6) P2M – Tor-2-Betriebsart

Jeder Leitung von Tor 2 ist das entsprechende Bit von R 246 zugeordnet. Ist ein Bit gesetzt, so ist die zugehörige Torleitung als Eingang definiert, andernfalls als Ausgang.

R 247 (%F7) P3M – Tor-3-Betriebsart

Das Betriebsartenregister für Tor 3 dient zur Festlegung der Funktionen der Zähler/Zeitgeber und der seriellen Schnittstelle. Als Besonderheit dieses Registers ist zu bemerken, daß Bit 0 die Ausgangsstufen des Tores 2 beeinflußt. Das Register 247 hat folgende Bitbelegung:



Das Bit D0 muß gesetzt sein, wenn Tor 2 ganz oder teilweise in der Betriebsart Ausgabe genutzt wird und die nachfolgende Schaltung TTL-Pegel zur Ansteuerung erfordert. Die open-drain-Charakteristik gestattet ein Zusammenschalten mehrerer gleichartiger Ausgänge (wired OR) über einen gemeinsamen externen Lastwiderstand.

Das Bit D2 legt die Verwendung der Leitungen P32 und P35 fest. Mit D2 = 0 werden diese Leitungen für die universelle Ein-/Ausgabe spezifiziert, andernfalls erfolgt die Nutzung dieses Paares für eine Handshake-Steuerung von Tor 0.

Die Bits D3 und D4 des Registers 247 bestimmen in ähnlicher Weise die Verwendung des Leitungspaares P33/P34. Neben der universellen Nutzung und der Bereitstellung der Handshake-Steuersignale für Tor 1 besteht hier eine dritte Möglichkeit in der Generierung des Signals /DM. Dieses Signal gestattet die Aufteilung des externen Speichers in einen Daten- und Programmspeicher. Voraussetzung ist die Einbeziehung des Signals in die Adreßdekodierung des externen Speichers, P33 ist in dieser Betriebsart als universell nutzbare Eingangsleitung festgelegt.

Mit D5 wird die Verwendung der Leitungen P31 und P36 beeinflußt. Es besteht zum

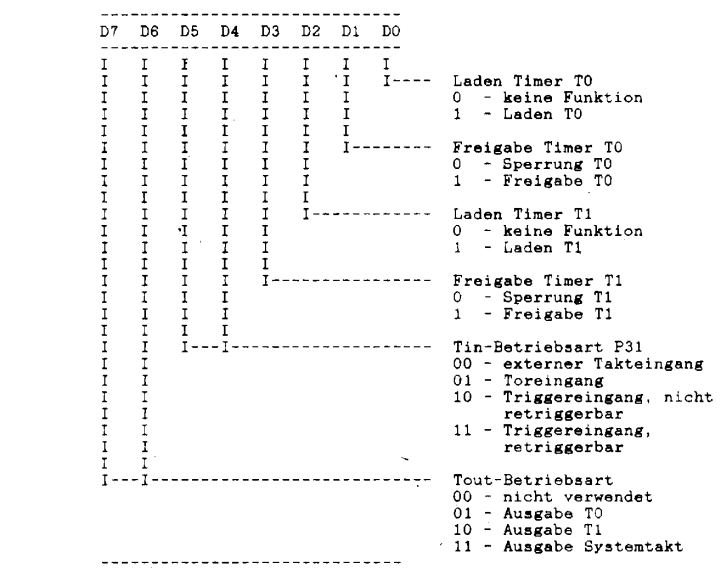
einen die Möglichkeit, diese Leitungen als Handshake-Steuersignale für Tor 2 zu verwenden (D5 = 1), andererseits ist mit D5 = 0 die Nutzung als universelles Ein-/Ausgabeleitungspaar festlegbar. Dabei ist jedoch die Programmierung des Zähler/Zeitgeberregisters (TMR) zu beachten (T_{in}/T_{out} über P31/P36).

Das Bit D6 wählt die Betriebsart des Leitungspaares P30/P37 aus. Falls dieses Bit gesetzt ist, bildet P30 den seriellen Dateneingang und P37 den entsprechenden Datenausgang. Mit D6 = 0 sind beide Leitungen als universelles Ein-/Ausgabepaar nutzbar.

Das Bit D7 ist nur in Zusammenhang mit dem gesetzten Bit D6 sinnvoll verwendbar. Bei D7 = 0 erfolgt die asynchrone Datenübertragung von 8 Datenbit ohne Paritätsberechnung und -auswertung. Mit gesetztem D7 wird in jedem zu sendenden Zeichen das 8. Datenbit durch das ungerader Parität entsprechende Datenbit ersetzt.

R 241 (%F1) TMR – Zähler/Zeitgeberbetriebsart

Das Register TMR (timer mode register) steuert die Betriebsart der beiden Zähler/Zeitgeberkanäle. Es gilt folgende Bitbelegung:



Nach dem Setzen von D0 werden die in den Laderegistern stehenden Werte für Teilungsfaktor und Anfangswert in die entsprechenden Zähler des Kanals T0 übernommen, das anschließende Rücksetzen von D0 erfolgt automatisch. Der Zählvorgang wird mit dem Laden der Anfangswerte noch nicht gestartet, sondern die Freigabe des Kanals T0 muß mit D1 = 1 ausgelöst werden. In den meisten Fällen ist ein gleichzeitiges Setzen von D0 und D1 gestattet, der Zählvorgang wird dann unmittelbar nach dem Laden gestartet. In einigen ausgewählten Fällen führt dieses Prinzip jedoch zu Fehlfunktionen, deshalb sollte die Steuerung des Lade- und Startvorgangs durch 2 aufeinanderfolgende Befehle realisiert werden.

Die Bits D2 und D3 beeinflussen Kanal T1 in analoger Weise wie D0 und D1 den Kanal T0.

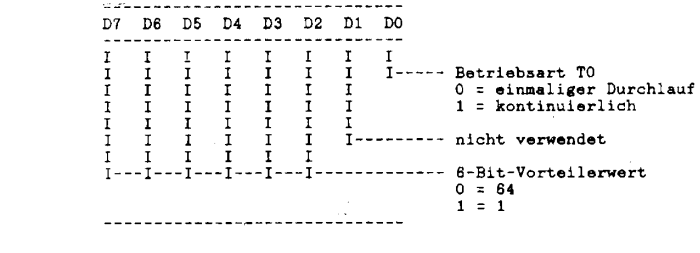
Die Bits D4 und D5 wirken nur in Zusammenhang mit T1. Für diesen Kanal kann P31 als externer Eingang (T_{in}) mit nachfolgenden Möglichkeiten definiert werden:

- externer Takteingang: an P31 liegt ein externer Takt, dieser wird dem Vorteiler von T1 mit aktiver HL-Flanke zugeführt.
- Toreingang: Der Kanal T1 wird nur dann mit dem internen Takt ($f_{XTAL}:8$) angesteuert, wenn P31 H-Pegel führt.
- Triggereingang, nicht retriggerbar: Nach einer HL-Flanke an P31 wird T1 geladen und gestartet, weitere HL-Flanken an P31 haben keinen Einfluß.
- Triggereingang, retriggerbar: Nach einer HL-Flanke an P31 wird T1 geladen und gestartet. Bei jeder weiteren HL-Flanke werden Vorteiler und Rückwärtszähler neu geladen. Die Zählerfreigabe bleibt erhalten.

Mit den Bits D6 und D7 wird die Verwendung der Leitung P36 in Zusammenhang mit den Zähler/Zeitgeberfunktionen definiert, falls P3M die Leitung P36 als allgemeinen Ausgang festlegt. Wird P36 als Ausgang T_{out} von T0 oder T1 verwendet, so gilt folgende Besonderheit:
Zwischen dem Ausgang des Rückwärtszählers und P36 befindet sich ein sogenanntes Ausgangs-Flipflop, das als 2:1-Teiler wirkt. Bei kontinuierlichem Zählbetrieb wird daher an P36 stets eine Impulsfolge mit einem Tastverhältnis von 0,5 (Impulsdauer = Pausendauer) ausgegeben. Diese Tatsache führt dazu, daß eine unmittelbare Kaskadierung beider Zähler/Zeitgeberkanäle nur hinsichtlich der Zeitgeberfunktion möglich ist. Die Kaskadierung der Kanäle zur Realisierung eines 16-Bit-Zählers erfordert eine hardwaremäßige Frequenzverdopplung zwischen P36 (T_{out}) und P31 (T_{in}). Für die Erweiterung des Zählumfangs ist daher die Nutzung der Interruptfunktion beider Zählkanäle (Erhöhung eines Universalregisters bei jedem Übertrag innerhalb der Interrupt-Behandlungsroutine) meist die geeignetere Lösung.

R 245 (%F5) PRE0 – T0-Vorteilerregister

PRE0 ist das Laderegister des 6-Bit-Vorteilers von Kanal 0. Außerdem wird durch Bit 0 dieses Registers die Betriebsart von T0 festgelegt. Es hat folgenden Aufbau:



Falls D0 nicht gesetzt ist, wird der Zählvorgang beim Erreichen des Zählerstands 0 automatisch abgebrochen. Für ein erneutes Zählen ist ein Freigabekommando durch Setzen des entsprechenden Bit im Register TMR erforderlich. Mit D0 = 1 läuft der Zählvorgang kontinuierlich ab, d. h., im Nulldurchgang des Zählers wird die Zeitkonstante bzw. der Zähleranfangswert aus dem Laderegister T0 übernommen und erneut dekrementiert. Während des laufenden Zählvorgangs können die Register PRE0 und T0 beschrieben werden, die neuen Werte werden beim nächsten Nulldurchgang übernommen. Da das Register PRE0 nicht lesbar ist, kann eine direkte softwaremäßige Testung des Registerinhalts nicht erfolgen.

R 244 (%F4) T0 – Zähler/Zeitgeberregister Kanal 0

Das Register T0 kann auf 2 Arten genutzt werden. Bei der Anwendung als Schreibregister dient es zum Einschreiben der Zeitkonstante bzw. des Zähleranfangswerts von T0. Wird es gelesen, so entspricht sein Inhalt dem aktuellen Zählerstand des Rückwärtszählers. Um den Anfangswert 256 zu erhalten, muß das Register 244 mit 0 geladen werden. In allen anderen Fällen ist der geladene Wert mit dem tatsächlichen Zählumfang identisch.

R 243 (%F3) PRE1 – T1-Vorteilerregister

Das Register PRE1 bildet das Vorteilerregister für den Kanal T1. Im Gegensatz zu PRE0 besteht ein funktioneller Unterschied hinsichtlich der Bedeutung des dort nicht benutzten Bits D1, welches die Taktquelle für T1 definiert. Es gilt folgende Bitbelegung:

```
-------------------------------
D7  D6  D5  D4  D3  D2  D1  D0
-------------------------------
I   I   I   I   I   I   I   I
I   I   I   I   I   I   I   I-----  Betriebsart T1
I   I   I   I   I   I   I           0 - einmaliger Durchlauf
I   I   I   I   I   I   I           1 - kontinuierlich
I   I   I   I   I   I   I
I   I   I   I   I   I   I---------  T1-Taktquelle
I   I   I   I   I   I               0 - extern über P31 (Tin)
I   I   I   I   I   I               1 - intern ( fXAL/8 )
I   I   I   I   I   I
I---I---I---I---I---I-------------  6-Bit-Vorteilerwert
                                    0 = 64
                                    1 = 1
-------------------------------
```

R 242 (%F2) T1 – Zähler/Zeitgeberregister

Dieses Register entspricht funktionell dem Register T0. Beim Schreiben wird der Zähleranfangswert übergeben, beim Lesen der aktuelle Zählerstand erhalten.

R 251 (%FB) IMR – Interrupt-Maskierungsregister

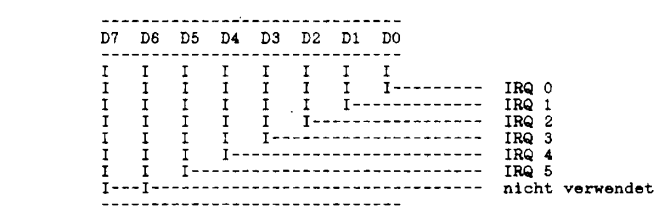
Das Interrupt-Maskierungsregister bildet zusammen mit den Registern 249 und 250 eine Gruppe für die Festlegung der Interruptbetriebsarten. Jeder der 6 möglichen Interruptquellen ist ein Bit des Registers 251 zugeordnet. Es gilt folgende Bitbelegung:

```
-------------------------------
D7  D6  D5  D4  D3  D2  D1  D0
-------------------------------
I   I   I   I   I   I   I   I
I   I   I   I   I   I   I   I-----  IRQ0  0 = gesperrt, 1 = frei
I   I   I   I   I   I   I---------  IRQ1  0 = gesperrt, 1 = frei
I   I   I   I   I   I-------------  IRQ2  0 = gesperrt, 1 = frei
I   I   I   I   I-----------------  IRQ3  0 = gesperrt, 1 = frei
I   I   I   I---------------------  IRQ4  0 = gesperrt, 1 = frei
I   I   I-----------------------------  IRQ5  0 = gesperrt, 1 = frei
I   I
I   I-------------------------------  nicht verwendet
I
I---------------------------------  0 = alle IRQ gesperrt
                                    1 = alle IRQ erlaubt
-------------------------------
```

Das Bit D7 korrespondiert direkt mit den Befehlen EI (enable interrupt) bzw. DI (disable interrupt). Da D7 auch während eines Interrupt-Behandlungsprogramms zurückgesetzt wird, besteht so lange Interruptverbot, wie dies nicht durch einen Befehl EI oder den Rückkehrbefehl RETI der Interruptroutine aufgehoben wird. Ein Beschreiben des Registers IMR durch das Programm darf nur bei Interruptverbot erfolgen, es muß also vor einem Ladebefehl immer der Befehl DI stehen. Nach einem RESET ist D7 immer zurückgesetzt.

R 250 (%FA) IRQ – Interrupt-Anmelderegister

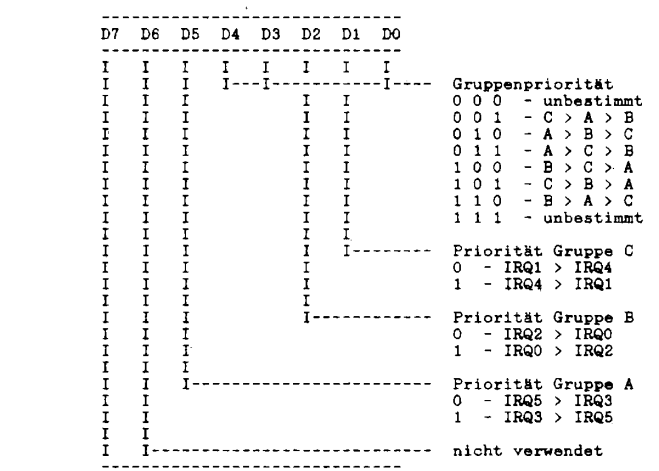
Im Register IRQ (interrupt request) ist jeder Interruptquelle ein Bit zugeordnet. Einem gesetzten Bit entspricht dabei eine vorliegende Interruptanmeldung. Innerhalb des Registers besteht folgende Zuordnung:



Da das Register IRQ ein Schreib/Leseregister ist, kann durch Abfrage der Anmeldebits ein Pollingbetrieb realisiert werden, d. h., es kann eine Programmverzweigung in Abhängigkeit von einer eventuellen Interruptanmeldung programmiert werden. Nach einem RESET sind alle Interruptanmeldebits zurückgesetzt.

R 249 (%F9) IPR – Interrupt-Prioritätsregister

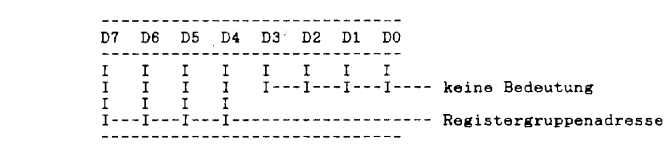
Das IPR legt die Reihenfolge der Abarbeitung der Interruptbehandlungsprogramme für den Fall fest, daß verschiedene Interruptquellen gleichzeitig ein Interruptgesuch anmelden. Es erfolgt eine Einteilung in 3 Gruppen (A, B, C), deren Prioritätsfolge festgelegt werden kann. Innerhalb der Gruppen ist die Reihenfolge durch weitere Bits des Registers IPR zu bestimmen. Das Register IPR hat folgende Bitbelegung:



Das Register IPR kann im Programm unberücksichtigt bleiben, wenn ein gleichzeitiges Auftreten mehrerer Interruptanforderungen entweder nicht möglich ist oder die Reihenfolge der Abarbeitung keine Rolle spielt. Ein laufendes Interruptbehandlungsprogramm kann unabhängig vom Interrupt-Prioritätsregister durch jede andere Interruptquelle unterbrochen werden, sofern das durch einen EI-Befehl überhaupt erlaubt ist.

R 253 (%FD) RP – Registerpointer

Der Registerpointer stellt eine 4-Bit-Adresse für die Adressierung einer Registergruppe, bestehend aus 16 Registern, bereit. Wenn Befehle mit Arbeitsregisteradressierung abgearbeitet werden, bilden die Bits D4 bis D7 des Registerpointers den H-Teil der 8-Bit-Registeradresse. Die niederwertigen 4 Adreßbits sind im entsprechenden Befehlscode verschlüsselt. Auf diese Weise kann eines der 16 Register einer Gruppe adressiert werden. Der Registerpointer hat folgenden Aufbau:

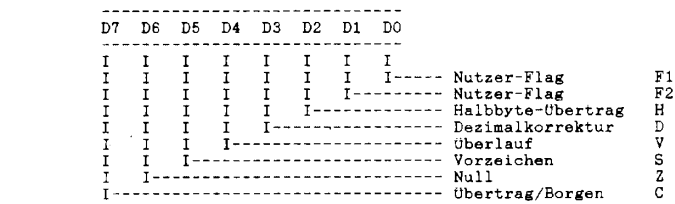


Ein Beschreiben der Bits D0 bis D3 hat keine Auswirkung. Im Gegensatz zu anderen Registern steht für das Laden des Registerpointers ein spezieller 2-Byte-Befehl zur Verfügung. Die Anwendung eines allgemeinen Ladebefehls ist prinzipiell ebenfalls möglich.

In der Befehlsliste des Einchipmikrorechners sind nicht alle Befehle mit Arbeitsregisteradressierung ausgewiesen. Diese Adressierungsart ist dennoch bei allen Befehlen möglich, wenn die oberen 4 Bit der Registeradresse durch %E ersetzt werden. In diesem Falle wird automatisch der Registerpointer eingeblendet. Da Registeradressen mit der Bitkombination %E in den hochwertigen 4 Adreßbits nicht existieren, bleiben die Verhältnisse eindeutig. Werden Befehle, die normalerweise nicht für Arbeitsregisteradressierung vorgesehen sind, auf diese Weise modifiziert, so ergeben sich zwar weder hinsichtlich der Befehlsabarbeitungszeit noch in bezug auf die Zahl der Befehlsbytes Vorteile, jedoch ist es damit möglich, komplette Unterprogramme ausschließlich mit Arbeitsregisteradressierung zu schreiben. Durch Umladen des Registerpointers kann das entsprechende Programm dann auch in einer anderen Registergruppe abgearbeitet werden. Diese Methode ist vor allem bei Standardprozeduren sinnvoll, die bei verschiedenen Aufgabenstellungen immer wieder Verwendung finden sollen.

R 252 (%FC) FLAGS – Flagregister

In Abhängigkeit vom Ergebnis einer Operation werden von der ALU 6 Flags beeinflußt. Die Bits D0 und D1 stehen in keinem Zusammenhang mit der Wirkung von Befehlen. Sie stehen dem Anwender zur Verfügung und können zur Kennzeichnung eines bestimmten Zustands dienen. Da das Flagregister beschrieben und gelesen werden kann, stehen für die Bitmanipulation alle dafür vorgesehenen Befehle zur Verfügung. Das Übertragsflag C ist durch spezielle 1-Byte-Befehle beeinflußbar (Setzen, Rücksetzen, Komplementieren). Das Flagregister hat folgende Bitbelegung:



Befehle zur bedingten Programmverzweigung (Sprungbefehle) testen die Flags V, S, Z und C. Dagegen haben die Flags H und D ausschließlich für die BCD-Arithmetik Bedeutung.

R 240 (%F0) SIO – serielles Ein-/Ausgaberegister

Das SIO-Register hat Bedeutung, wenn die Torleitungen P30 und P37 als Ein- bzw. Ausgang des UART definiert sind. Es wird dann beim Senden mit den zu übertragenden Daten beschrieben. Beim seriellen Empfang entspricht der Registerinhalt dem empfangenen Datenbyte. Das Datenformat beim Senden und Empfangen serieller Daten ist über Bit D7 des Registers P3M wählbar. Es werden entweder 8 Datenbits ohne Parität übertragen, oder das Datenbit D7 wird durch das Paritätsbit ersetzt. Die eigentlichen Datenbits sind demnach in diesem Falle D0 bis D6. Beim Lesen des SIO-Registers ist ein Paritätsfehler vorhanden, wenn Bit D7 gesetzt ist.

R 255 (%FF) SPL – Stackpointer, niederwertiger Teil

Das Register bildet den L-Teil des 16-Bit-Stackpointers. Falls der Stack im internen Registersatz definiert ist (durch Bit D2 des Registers P01M festgelegt), erfolgt die Stackadressierung ausschließlich mit diesem Register. Ist der Stack Teil des externen Speichers, arbeitet SPL zusammen mit SPH als 16-Bit-Stackpointer.

R 254 (%FE) SPH – Stackpointer, höherwertiger Teil

Das Register SPH bildet die hochwertigen 8 Adreßbits des 16-Bit-Stackpointers. Bei einem im Registersatz lokalisierten Stack kann es prinzipiell als Universalregister genutzt werden. Eine Beeinflussung durch Überlauf des SPL-Registers ist allerdings zu berücksichtigen.

Tabelle 5.1 faßt die Belegungen der Status- und Steuerregister nach einem RESET zusammen. Ein «x» kennzeichnet einen nicht definierten Zustand des betreffenden Bits.

Register	D7	D6	D5	D4	D3	D2	D1	D0	Bemerkung
%F0 SIO	x	x	x	x	x	x	x	x	unbestimmt
%F1 TMR	0	0	0	0	0	0	0	0	beide Zählkanäle gesperrt
%F2 T1	x	x	x	x	x	x	x	x	unbestimmt
%F3 PRE1	x	x	x	x	x	x	0	0	externer Takt, einmaliger Lauf
%F4 T0	x	x	x	x	x	x	x	x	unbestimmt
%F5 PRE0	x	x	x	x	x	x	0	0	einmaliger Lauf
%F6 P2M	1	1	1	1	1	1	1	1	P20 bis P27 sind Eingänge
%F7 P3M	0	0	0	0	0	0	x	0	P30 bis P33 sind Eingänge, P34 bis P37 sind Ausgänge, P2 mit open-drain-Ausgang
%F8 P01M	0	1	0	0	1	1	0	1	P0 und P1 auf Eingabe, interner Stack, normales Speicherzeitverhalten
%F9 IPR	x	x	x	x	x	x	x	x	unbestimmt
%FA IRQ	x	x	0	0	0	0	0	0	alle Anforderungen gelöscht
%FB IMR	0	x	x	x	x	x	x	x	Interrupts global gesperrt
%FC FLAGS	x	x	x	x	x	x	x	x	unbestimmt
%FD RP	x	x	x	x	x	x	x	x	unbestimmt
%FE SPH	x	x	x	x	x	x	x	x	unbestimmt
%FF SPL	x	x	x	x	x	x	x	x	unbestimmt

5.2. *Hinweise zur Nutzung der Interrupt-Steuerregister*

Die Schaltkreise *U881/U882* haben eine besonders leistungsfähige Interruptlogik, die über die Steuerregister IMR, IRQ und IPR beeinflußbar ist. Zur Interruptstruktur des *U880* bestehen grundsätzliche Unterschiede. Gemeinsamkeiten sind praktisch nur im Hinblick auf die vektorisierte Interruptbehandlung gegeben. Die Interruptvektoren stehen beim Einchipmikrorechner prinzipiell auf den unteren 12 Plätzen des Programmspeichers in der Reihenfolge H-Teil, L-Teil. Jeder der 6 möglichen Interruptanforderungen ist demnach eine feste Adresse für den Interruptvektor zugewiesen.

Bei den Schaltkreisen *U883* und *U886* weisen die Interruptvektoren auf die ersten 18 Zellen des extern angeschlossenen Speichers (Adressen %800 bis %812). Dort kann für jede Anforderung ein 3-Byte-Sprungbefehl auf die eigentliche Anfangsadresse des Interruptbehandlungsprogramms eingetragen werden.

Bei der Arbeit mit den Interrupt-Steuerregistern ist grundsätzlich zu beachten, daß eine Beeinflussung dieser Register stets bei globaler Interruptsperrung vorzunehmen ist (DI-Befehl). Wird das nicht beachtet, können unkontrollierte Interruptanforderungen das Programm zum Absturz bringen. Nach einem RESET besteht immer eine globale Interruptsperre, die Anmeldebits im Register IRQ sind zurückgesetzt. Durch das Setzen der entsprechenden Bits im Interrupt-Maskierungsregister erfolgt eine Freigabe der zugehörigen Interruptquellen. Interruptanforderungen führen aber nur dann zum Setzen des Anmeldebits, wenn vorher der EI-Befehl abgearbeitet wurde.

Eine besondere Möglichkeit der Einchipmikrorechner besteht in der Interruptbedienung mit Hilfe von Polling. Dabei erfolgt die Testung der Bits im Interrupt-Anmelderegister. Das entsprechende Bit im Maskierungsregister muß in diesem Falle mit 0 belegt sein, d. h., die Interruptfreigabe für die im Polling getestete Anforderung ist zu sperren. Wird eine Anforderung nach positivem Testergebnis bearbeitet, ist das zugehörige Bit im Anmelderegister per Programm zurückzusetzen. Bei der «normalen» Bedienung von Interruptanforderungen wird dieser Vorgang von der Interruptlogik automatisch ausgeführt. Die Bedienung von Interrupts über Polling ist u. a. dann sinnvoll, wenn die Anmeldung einer Forderung zwar erkannt, die Reaktion in Form eines Behandlungsprogramms aber nicht sofort ausgelöst werden soll.

Hinsichtlich der Festlegung von Interruptprioritäten hat das Register IPR nur dann Einfluß, wenn nach einer globalen Interruptfreigabe mehrere Forderungen gleichzeitig anstehen. Dagegen ist die Festlegung von Prioritäten hinsichtlich der Unterbrechungsmöglichkeit eines laufenden Behandlungsprogramms durch eine andere Interruptforderung über die Modifizierung des IMR in der laufenden Behandlungsroutine und einem nachfolgenden EI-Befehl möglich.

In vielen praktisch auftretenden Fällen muß die Wahrscheinlichkeit des gleichzeitigen Auftretens mehrerer Interruptforderungen nicht besonders berücksichtigt werden, auf eine Beeinflussung des IPR-Registers kann dann verzichtet werden. Die Abarbeitung von freigegebenen Anforderungen ist prinzipiell gegeben, wobei die Reihenfolge der Bedienung bei gleichzeitigem Auftreten dann unbekannt bleibt. Ein direktes Lesen des IPR-Registers ist nicht möglich, da es sich um ein reines Schreibregister handelt.

Im Unterschied zum Interruptverhalten des *U880* erfolgt bei den Einchipmikrorechnern nicht nur die Rettung der Rückkehradresse, sondern auch die Abspeicherung des Flagregisters in den Stack. Durch den Rückkehrbefehl IRET wird der aktuelle Zustand

des Flagregisters vor dem Eintritt in die Interruptbehandlung also wiederhergestellt. Außerdem erfolgt durch diesen Befehl automatisch eine globale Interruptfreigabe. Soll das unterbunden werden, ist der Befehl IRET durch die Befehlsfolge
POP FLAGS
RET
zu ersetzen.

6. Der Befehlssatz

6.1. Befehlsgruppen

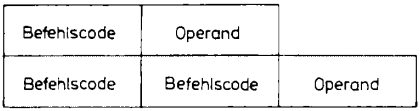
Die Liste der Befehle der Einchipmikrorechner *U881/U882* läßt sich in folgende Gruppen einteilen:
- Lade- und Transportbefehle,
- Arithmetik- und Logikbefehle,
- Rotations- und Verschiebebefehle,
- Programmsteuerbefehle,
- CPU-Steuerbefehle.

Die Befehle beziehen sich auf einzelne Bits, Halbbytes (Nibbles), Bytes und Doppelbytes (Worte). Für das letztgenannte Datenformat existieren allerdings nur Befehle zum Inkrementieren und Dekrementieren. Eine 16-Bit-Arithmetik wie beim System *U880* ist nicht implementiert.

6.2. Adressierungsarten

In den Befehlen werden die Operanden durch ihre Adressen definiert. Das sind entweder Registeradressen oder Adressen im internen oder externen Speicherbereich. Die Leistungsfähigkeit der EMR beruht u. a. auf der Möglichkeit, Befehle in den unterschiedlichsten Adressierungsarten ausführen zu können. Man unterscheidet:
- unmittelbare Adressierung,
- direkte Adressierung,
- relative Adressierung,
- relative Registeradressierung,
- indizierte Registeradressierung.

Unmittelbare Adressierung
Der Operand bei diesem Befehlstyp ist eine Konstante. Sein Wertebereich reicht von −128 bis +127 oder von 0 bis 255, je nach Darstellungsart. Der Operationskode besteht aus 1 oder 2 Bytes Befehlskode und 1 Byte Operand.



Direkte Adressierung
Die direkte Adressierung bezieht sich auf eine 16-Bit-Adresse, mit der der Befehlszähler bei Ausführung des Befehls geladen wird. Es handelt sich um direkte Ruf- (CALL)

und Sprungbefehle. Im Gegensatz zum *U880* folgt nach dem Operationskode immer der H-Teil der 16-Bit-Adresse. Der Befehlsaufbau ist demnach folgendermaßen:

Operationscode 1. Byte	Adresse H-Teil 2. Byte	Adresse L-Teil 3. Byte

Relative Adressierung

Bei den Relativsprungbefehlen und bei dem Schleifenbefehl DJNZ folgt auf den Operationskode eine 8-Bit-Distanz relativ zum aktuellen PC-Stand. Ähnlich wie beim Prozessor *U880* ist diese Distanz im Bereich von −128 bis +127 definiert. Die Befehle haben stets eine Länge von 2 Bytes.

Operationscode 1. Byte	Abstandsoperand 2. Byte

Registeradressierung

Bei der Registeradressierung ist der Operand Inhalt des adressierten Registers oder Registerpaares. Jedes Register läßt sich entweder durch eine 8-Bit-Adresse oder durch eine 4-Bit-Arbeitsregisteradresse spezifizieren, wobei im letzteren Fall die oberen 4 Adreßbits dem Registerpointer entnommen werden.

Durch diese beiden Möglichkeiten ergeben sich insgesamt 4 Modifikationen für die Registeradressierung:

- Adressierung eines Einzelregisters (8-Bit-Adresse),
- Adressierung eines Arbeitsregisters (4-Bit-Arbeitsregisteradresse),
- Adressierung eines Registerpaares (8-Bit-Adresse),
- Adressierung eines Arbeitsregisterpaares (4-Bit-Arbeitsregisteradresse).

Bei der Adressierung von Registerpaaren ist zu beachten, daß das niederwertigste Adreßbit ignoriert wird. Es handelt sich demnach effektiv um eine 7-Bit-Adresse. Die praktische Folge besteht darin, daß nur Registerpaare mit geradzahligen Adressen angewählt werden können. Das gilt auch bei der Verwendung von 4-Bit-Arbeitsregisteradressen. Die Angabe einer ungeradzahligen Adresse im Programm ist zwar prinzipiell möglich, tatsächlich wird aber auch dann ein geradzahliges Registerpaar adressiert. Aus der Unkenntnis dieses Sachverhalts können sich scheinbar «unerklärliche» Fehler bei der Programmabarbeitung ergeben.

Indirekte Registeradressierung

Bei dieser Adressierungsart steht im adressierten Register die Adresse des Registers, Arbeitsregisters, Registerpaares oder Arbeitsregisterpaares, dessen Inhalt der Operand ist. Insgesamt lassen sich wiederum 4 Adressierungsmöglichkeiten ableiten:

- indirekte Einzelregisteradressierung,
- indirekte Arbeitsregisteradressierung,
- indirekte Registerpaaradressierung,
- indirekte Arbeitsregisterpaaradressierung.

Bei der indirekten Einzelregisteradressierung wird stets ein Register angesprochen, dessen Inhalt die Adresse des eigentlichen Operanden, d. h. eines Registers, liefert. Werden Doppelregister indirekt adressiert, bildet deren Inhalt eine 16-Bit-Adresse, die auf eine Zelle des Programm- oder Datenspeichers weist.

Indizierte Registeradressierung
Diese Adressierungsart ist vergleichbar mit den entsprechenden Befehlen für das IX- bzw. IY-Register des Prozessors *U880*. Die Indexadresse ergibt sich aus einer Basisadresse (8-Bit-Einzelregisteradresse) und einer dazu addierten Verschiebekonstante, die durch den Inhalt eines Arbeitsregisters gebildet wird. Es handelt sich um 3-Byte-Befehle, die effektiv bei der Abarbeitung von Tabellen einsetzbar sind. In der Praxis besteht nur selten die Notwendigkeit, von dieser Adressierungsart Gebrauch zu machen. Die vorher genannten Varianten der Adressierung sind übersichtlicher und in den meisten Fällen völlig ausreichend.

6.3. Befehlsbeschreibung

Für die Einchipmikrorechner *U881/U882* hat sich in bezug auf die Darstellung von Konstanten und Befehlen eine Symbolik durchgesetzt, die von den bekannten Darstellungsformen innerhalb des Systems *U880* abweicht. Die folgende Übersicht faßt die verwendeten Symbole zusammen. Teilweise wird eine vereinfachte Schreibweise verwendet, die für den Amateur übersichtlicher ist.

Symbol	Bedeutung
%	Der folgende Operand ist in hexadezimaler Form dargestellt.
#	Der folgende Operand ist eine Konstante (Direktoperand).
Z	Zieloperand bzw. 8-Bit-Adresse des Zielregisters.
z	4-Bit-Adresse des Zielregisters bzw. Zielregisterpaares bei Arbeitsregisteradressierung.
Q	Quelloperand bzw. 8-Bit-Adresse des Quellregisters.
q	4-Bit-Adresse des Quellregisters bzw. Quellregisterpaares bei Arbeitsregisteradressierung.
Rn	Arbeitsregister (n für Register-Nr. 0 bis 15).
nn	Absolutregister.
#D	Konstante (Direktoperand).
@	Kennzeichnung der indirekten Adresse.
x	Verschiebekonstante (bei indizierter Adressierung).
cc	Bedingungskode (bei Sprungbefehlen).
DA	Direktadresse (DAH: H-Teil, DAL: L-Teil).
RA	Relativadresse (Sprungweite).

Flags	
.	Flag unbeeinflußt.
0	Flag = 0, nicht gesetzt.
1	Flag = 1, gesetzt.
I	Flag entsprechend Ergebnis beeinflußt.
X	Flag undefiniert.

Ladebefehle

LD Z,Q Ladebefehl (load)

Der Inhalt des Quellregisters wird in das Zielregister geladen.

Flags: unbeeinflußt

LDC Z,Q Datenaustausch mit Programmspeicher (load constant)

Der Befehl dient zum Laden eines Arbeitsregisters mit dem Inhalt einer Speicherzelle des externen Programmspeichers bzw. umgekehrt. Die Adresse der Speicherzelle wird durch den Inhalt eines Arbeitsregisters definiert. Falls zwischen dem externen Programmspeicher und dem Datenspeicher nicht unterschieden wird (/DM-Signal nicht verwendet), ist die Wirkung der Befehle LDC und LDE identisch.

Flags: unbeeinflußt

LDCI Z,Q Datenaustausch mit Programmspeicher und automatischem Inkrementieren der Adreßzeiger (load constant and autoincrement)

Durch diesen Befehl können ebenfalls Daten mit dem externen Programmspeicher ausgetauscht werden. Die Adressierung des verwendeten Arbeitsregisters erfolgt aber indirekt, d. h. über ein Zeiger- oder Pointerregister. Nach jedem Datenaustausch werden beide Zeigerregister inkrementiert. Der Befehl ist daher besonders für einen Blocktransfer geeignet. Wird nicht zwischen externem Datenspeicher und Programmspeicher unterschieden, ist die Wirkung der Befehle LDCI und LDEI identisch.

Flags: unbeeinflußt

LDE Z,Q Datenaustausch mit externem Datenspeicher (load external data)

Der Unterschied zum Befehl LDC besteht im Zugriff auf den externen Datenspeicher, wenn das Unterscheidungssignal /DM in die Adreßdekodierung überhaupt einbezogen ist. Anderenfalls ist die Wirkung der Befehle LDE und LDC identisch.

Flags: unbeeinflußt

LDEI Z,Q Datenaustausch mit externem Datenspeicher und automatischem Inkrementieren der Adreßzeiger (load external data and autoincrement)

Der Befehl greift auf den externen Datenspeicher zu. Ansonsten ist seine Wirkung mit LDCI identisch. Bei nicht generiertem /DM-Signal können beide Befehle wahlweise verwendet werden.

Flags: unbeeinflußt

CLR Z Laden mit Null (clear)

Das Zielregister wird mit dem Wert Null geladen.

Flags: unbeeinflußt

PUSH Q Laden des Kellerspeichers (push)

Der Inhalt des adressierten Registers wird in die durch den Stackpointer adressierte Speicherzelle (externer Stack) bzw. in das durch den Stackpointer adressierte Register (interner Stack) geladen. Vor dem Datentransport erfolgt das Dekrementieren des Stackpointers.

Flags: unbeeinflußt

POP Z Datentransport vom Kellerspeicher (pop)

Das Zielregister wird mit dem Inhalt der durch den Stackpointer adressierten Speicherzelle (externer Stack) bzw. mit dem Inhalt des durch den Stackpointer adressierten Registers (interner Stack) geladen. Nach dem Datentransport wird der Stackpointer inkrementiert.

Flags: unbeeinflußt.

Zur Verdeutlichung der Wirkungsweise von Ladebefehlen folgen einige Beispiele.

Op.-Code	Mnemonik	Erklärung
B0 10	CLR %10	Register %10 wird mit 0 geladen
E6 10 15	LD %10,#%15	Register %10 wird mit der Konstanten %15 geladen
E6 00 10	LD 0,#%10	über Port 0 wird die Konstante %10 ausgegeben, wenn dieses Port auf auf Ausgabe festgelegt ist
E4 15 10	LD %10,%15	Register %10 wird mit dem Registerinhalt von %15 geladen
E5 15 10	LD %10,@15	Register %10 wird mit dem Inhalt eines Registers geladen, dessen Adresse in %15 steht
F5 15 10	LD @%10,%15	Ein Register, dessen Adresse in %10 steht, wird mit dem Inhalt von %15 geladen definierte Registerinhalte zur Demonstration des LDE-Befehls herstellen
31 10	SRP #%10	Arbeitsregistergruppe %10 bis %1F
0C 10	LD R0,#%10	H-Teil Adresse laden
1C 00	LD R1,#0	L-Teil Adresse laden
82 20	LDE R2,@RR0	Arbeitsregister R2 (absolut %12) wird mit dem Inhalt der durch R0 und R1 adressierten Zelle (im Beispiel %1000) des externen Datenspeichers geladen. (Speicher Lesen)
92 20	LDE @RR0,R2	Die durch R0 und R1 adressierte Speicherzelle wird mit dem Inhalt von R2 geladen. (Speicher Schreiben)

Arithmetik- und Logikbefehle

ADD Z,Q Addition (add)

Quell- und Zielregister werden addiert. Das Ergebnis steht im Zielregister.
Flags:
C = 1, wenn ein Übertrag vom höchstwertigen Bit entsteht.
Z = 1, wenn das Ergebnis Null ist.
V = 1, wenn arithmetischer Überlauf eintritt.
S = 1, wenn das Ergebnis kleiner Null ist.
H = 1, wenn ein Übertrag von der niederwertigen Tetrade auftritt.
D = 0.

ADC Addition mit Berücksichtigung des C-Flags (add with carry)

Quellbyte, Zielbyte und C-Bit werden addiert. Das Ergebnis steht im Zielregister.
Flags: Beeinflussung entsprechend ADD-Befehl.

SUB Z,Q Subtraktion (subtract)

Das Quellbyte wird vom Zielbyte subtrahiert. Das Ergebnis steht im Zielregister.
Flags:
C = 1, wenn Übertrag vom höchstwertigen Bit entsteht.
Z = 1, wenn das Ergebnis Null ist.
V = 1, wenn arithmetischer Überlauf eintritt.
S = 1, wenn das Ergebnis kleiner Null ist.
H = 1, wenn ein Übertrag von der niederwertigen Tetrade auftritt.
D = 1.

SBC Z,Q Subtraktion mit Berücksichtigung des C-Flags (subtract with carry)

Quellbyte und Carrybit werden vom Zielregister subtrahiert. Das Ergebnis steht im Zielregister. Flags: Beeinflussung entsprechend SUB-Befehl.

DA Z Dezimalkorrektur (decimal adjust)
Der Befehl bewirkt eine Korrektur des Ergebnisses einer Addition bzw. Subtraktion von gepackten BCD-Daten. Nach Ausführung des Befehls steht das Ergebnis im Zielregister ebenfalls in BCD-Form.
Flags:
C = 1, wenn ein Übertrag vom höchstwertigen Bit entsteht.
Z = 1, wenn das Ergebnis Null ist.
V unbestimmt.
S = 1, wenn das Ergebnisbit D7 = 1 ist.
H unbeeinflußt.
D unbeeinflußt.

CP Z,Q Vergleich (compare)
Der Inhalt des Quellregisters wird vom Inhalt des Zielregisters subtrahiert. Die Flags werden entsprechend dem Ergebnis beeinflußt. Ziel- und Quellregister sind nach der Operation unverändert.
Flags:
C = 1, wenn ein Übertrag vom höchtwertigen Bit entsteht.
Z = 1, wenn das Ergebnis Null ist.
V = 1, wenn arithmetischer Überlauf eintritt.
S = 1, wenn das Ergebnis kleiner Null ist.
H unbeeinflußt.
D unbeeinflußt.

INC Z Inkrementieren (increment)
Der Inhalt des Zielregisters wird um 1 erhöht.
Flags:
C unbeeinflußt
Z = 1, wenn das Ergebnis Null ist.
V = 1, wenn arithmetischer Überlauf eintritt
S = 1, wenn das Ergebnisbit D7 = 1 ist.
H unbeeinflußt.
D unbeeinflußt.

DEC Z Dekrementieren (decrement)
Der Inhalt des Zielregisters wird um 1 vermindert.
Flags: Beeinflussung entsprechend INC-Befehl.

INCW Z Doppelregister inkrementieren (increment word)
Das Zielregister bildet zusammen mit dem nächsthöheren Register ein 16-Bit-Doppelregister, dessen Inhalt durch den Befehl um 1 erhöht wird.
Flags:
C unbeeinflußt.
Z = 1, wenn das Ergebnis Null ist.
V = 1, wenn arithmetischer Überlauf eintritt.
S = 1, wenn das Ergebnisbit D15 = 1 ist.
H unbeeinflußt.
D unbeeinflußt.
Das adressierte Zielregister muß stets geradzahlig sein. Es bildet den höherwertigen Teil des 16-Bit-Registers.

DECW Z Doppelregister dekrementieren (decrement word)
Das Zielregister bildet zusammen mit dem nächsthöheren Register ein 16-Bit-Doppelregister, dessen Inhalt durch den Befehl um 1 vermindert wird.
Flags: Beeinflussung entsprechend INCW-Befehl.
Das adressierte Zielregister muß stets geradzahlig sein. Es bildet den höherwertigen Teil des 16-Bit-Registers.

AND Z,Q Logisches UND (and)
Ziel- und Quellbyte werden logisch UND-verknüpft. Das Ergebnis steht im Zielregister.
Flags:
C, unbeeinflußt.
Z = 1, wenn das Ergebnis Null ist.
V = 0.
S = 1, wenn das Ergebnisbit D7 = 1 ist.
H unbeeinflußt.
D unbeeinflußt.

OR Z,Q Logisches ODER (or)
Ziel- und Quellbyte werden logisch ODER-verknüpft. Das Ergebnis steht im Zielregister.
Flags: Beeinflussung entsprechend AND-Befehl.

XOR Exklusiv-ODER (exclusive or)
Ziel- und Quellbyte werden logisch Exklusiv-ODER verknüpft. Das Ergebnis steht im Zielregister.
Flags: Beeinflussung entsprechend AND-Befehl.

TM Z,Q Testen unter Maske (test under mask)
Quell- und Zielbyte werden logisch UND-verknüpft, ohne das Ergebnis in eines der beteiligten Register zu laden. Es werden ausschließlich die Flags beeinflußt.
Flags: Beeinflussung entsprechend AND-Befehl.

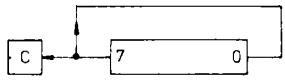
TCM Z,Q Testen des komplementierten Zielbytes (test complement under mask)
Das komplementierte Zielbyte (bitweise negiert) wird mit dem Quellbyte logisch UND-verknüpft. Das Ergebnis verändert die beteiligten Register nicht. Es werden ausschließlich die Flags beeinflußt.
Flags: Beeinflussung entsprechend AND-Befehl.

COM Z Komplement (complement)
Das Zielregister wird komplementiert (bitweise Negation).
Flags:
C = 1, wenn Bit D7 den Wert 1 hatte.
Z = 1, wenn das Ergebnis Null ist.
V = 1, wenn ein arithmetischer Überlauf eintritt.
S = 1, wenn das Ergebnisbit D7 = 1 ist.
H unbeeinflußt.
D unbeeinflußt.

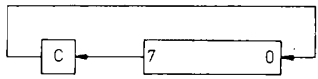
Rotations- und Verschiebebefehle

RL Z Rotieren nach links (rotate left)
Der Inhalt des Zielregisters rotiert um eine Bitposition nach links. Bit D7 wird in das Carryflag und in die Bitposition D0 geladen.
Flags:

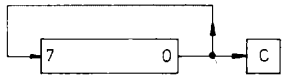
C = 1, wenn Bit D7 den Wert 1 hatte.
Z = 1, wenn das Ergebnis Null ist.
V = 1, wenn ein arithmetischer Überlauf eintritt.
S = 1, wenn das Ergebnisbit D7 = 1 ist.
H unbeeinflußt.
D unbeeinflußt.



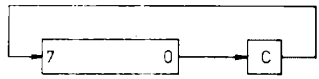
RLC Z Rotieren nach links über das Carryflag (rotate left trough carry)
Der Inhalt des Zielregisters rotiert um eine Bitposition nach links. Das Carryflag ist in die Rotation einbezogen. Nach der Operation steht der alte Inhalt des C-Flags in der Bitposition D0, während das C-Flag den vorherigen Wert des Bit D7 annimmt.
Flags: Beeinflussung entsprechend RL-Befehl.



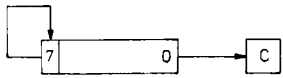
RR Z Rotieren nach rechts (rotate right)
Der Inhalt des Zielregisters rotiert um eine Bitposition nach rechts. Bit D0 wird in das Carryflag und in die Bitposition D7 geladen.
Flags:
C = 1, wenn Bit D0 den Wert 1 hatte.
Z = 1, wenn das Ergebnis Null ist.
V = 1, wenn arithmetischer Überlauf eintritt.
S = 1, wenn das Ergebnisbit D7 = 1 ist.
H unbeeinflußt.
D unbeeinflußt.



RRC Z Rotieren nach rechts über Carry (rotate right trough carry)
Der Inhalt des Zielregisters rotiert um eine Bitposition nach rechts. Das C-Flag ist in die Rotation eingeschlossen. Nach der Operation steht der alte Wert des C-Flags in der Bitposition D7, während das C-Flag den vorherigen Wert von Bit D0 annimmt.
Flags: Beeinflussung entsprechend RR-Befehl.

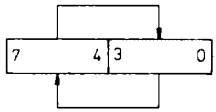


SRA Z Arithmetische Verschiebung nach rechts (shift right arithmetic)
Der Inhalt des Zielregisters wird um eine Bitposition nach rechts verschoben, Bit D7 bleibt unverändert. In das C-Flag wird Bit D0 eingetragen.
Flags:
C = 1, wenn Bit D0 den Wert 1 hatte.
Z = 1, wenn das Ergebnis Null ist.
V = 0.
S = 1, wenn das Ergebnisbit D7 = 1 ist.
H unbeeinflußt.
D unbeeinflußt.



SWAP Z Tetradentausch (swap nibbles)
Beide Tetraden des Zielregisters werden gegeneinander ausgetauscht.
Flags:
C undefiniert.
Z = 1, wenn das Ergebnis Null ist (praktisch nur bei Registerinhalt Null möglich).
V undefiniert.

S = 1, wenn das Ergebnisbit D7 = 1 ist.
H unbeeinflußt.
D unbeeinflußt.



Ausgewählte Beispiele zur Anwendung der Arithmetik- und Logikbefehle:

Op.-Code	Mnemonik	Erklärung
06 10 20	ADD %10,#%20	Zum Inhalt des Register %10 wird die Konstante %20 addiert. Das Ergebnis steht in %10
04 20 10	ADD %10,%20	Die Registerinhalte von %10 und %20 werden addiert, das Ergebnis steht in %10
05 11 10	ADD %10,@%11	Zum Inhalt von %10 wird der Inhalt eines Registers addiert, dessen Adresse in %11 steht
A6 10 15	CP %10, #%15	Der Registerinhalt von %10 wird mit Konstanten %15 verglichen. Die Flags werden entsprechend dem Ergebnis beeinflußt. Praktisch erfolgt eine Subtraktion, ohne das Ergebnis in eines der beteiligten Register zu laden
A5 15 10	CP %10,@%15	Der Inhalt von %10 wird mit dem Inhalt eines Registers verglichen, dessen Adresse in %15 steht
A0 10	INCW %10	Die Register %10 und %11 bilden ein Doppelregister, dessen Inhalt nach Ausführung um 1 erhöht ist. Der H-Teil steht in %10.
56 10 7F	AND %10,#%7F	Register %10 wird mit der Konstanten %7F logisch UND-verknüpft. Im Beispiel wird Bit D7 von %10 zurückgesetzt
54 15 10	AND %10,%15	Die Register %10 und %15 werden UND-verknüpft. Das Ergebnis steht in %10
46 10 80	OR %10,#%80	Register %10 wird mit der Konstanten %80 logisch ODER-verknüpft. Im Beispiel erfolgt ein Setzen von Bit D7 in %10
46 00 40	OR 0,#%40	Bit D2 von Port 0 wird gesetzt, sofern Ausgabe festgelegt ist
76 10 30	TM %10,#%30	Der Inhalt von %10 wird mit der Testmaske logisch UND-verknüpft. Im Beispiel läßt sich mit der Konstanten %30 testen, ob die Bits D0 und D1 von %10 gemeinsam 0 sind. Nur dann ist das Z-Flag gesetzt
66 10 30	TCM %10,#%30	Die Testmaske wird mit dem negierten Testbyte logisch UND-verknüpft, wiederum ohne das Register %10 zu verändern. Im Beispiel läßt sich feststellen, ob die Bits D0 und D1 gemeinsam auf 1 gesetzt sind. Nur dann ist das Z-Flag gesetzt.
F0 10	SWAP %10	Der Registerinhalt von %10 sei %0F. Nach Ausführung steht in diesem Register der Wert %F0
E6 10 FF	LD %10,#%FF	definierten Registerinhalt für Beispiel herstellen
D0 10	SRA %10	In %10 steht nach Ausführung %FF, das C-Flag ist gesetzt, da Bit D0 vorher 1 war

Programmsteuerbefehle

JP cc,DA Absoluter Sprung (jump)

Es wird ein Sprung zur angegebenen Adresse ausgeführt, sofern die Sprungbedingung erfüllt ist (bedingter Sprung). Es existiert auch ein unbedingter Sprungbefehl JP nn, wobei das Register nn und das Register nn + 1 ein 16-Bit-Doppelregister bilden, dessen Inhalt die Sprungadresse ist.
Flags: unbeeinflußt

JR cc,RA Relativer Sprung (jump relative)

Es wird ein Sprung mit der angegebenen Distanz ausgeführt, wenn die Sprungbedingung erfüllt ist.

Flags: unbeeinflußt

DJNZ Rn,Z Dekrementieren und Sprung bei ungleich Null (decrement and jump if not zero)

Das Arbeitsregister Rn bildet einen Zähler, der bei Befehlsausführung dekrementiert wird. Ist der Inhalt des Zählerregisters ungleich Null, erfolgt ein Sprung mit der angegebenen Distanz (Sprungweite). Anderenfalls wird das Programm beim folgenden Befehl fortgesetzt. Der Befehl ist nur für Arbeitsregisteradressierung definiert.

Flags: unbeeinflußt

CALL DA Unterprogrammaufruf (call)

Der alte Stand des Befehlszählers (PC) wird im Stack abgelegt und anschließend zur angegebenen Adresse gesprungen.

Flags: unbeeinflußt

RET Rückkehr vom Unterprogramm (return)

Der Befehlszählerstand bei Aufruf des Unterprogramms wird aus dem Stack in den PC zurückgeladen und anschließend der Stackpointer um 2 erhöht.

Flags: unbeeinflußt

IRET Rückkehr vom Interrupt (interrupt return)

Der PC-Stand beim Aufruf des Interrupt-Behandlungsprogramms wird aus dem Stack in den PC zurückgeladen. Die ebenfalls in den Stack geretteten Flags werden in das Flagregister eingetragen. Weiterhin erfolgt ein Setzen von Bit D7 des Interrupt-Maskierungsregisters, womit nach dem IRET-Befehl Interrupts stets global freigegeben sind.

Flags: Entsprechend dem Zustand bei Aufruf des Interrupt-Behandlungsprogramms gesetzt.

Einige Beispiele zur Anwendung der Programmsteuerbefehle:

Op.-Code	Mnemonik	Erklärung
8D 10 00	JP %1000	Im Unterschied zum U 880 steht die Adresse stets in der Reihenfolge H-Teil, L-Teil
31 10	SRP #%10	
0C 0A	LD R0,#%0A	Definierter Registerinhalt für Beispiel
0A FE M1:	DJNZ R0,M1	Es wird ein Relativsprung zur Marke M1 ausgeführt, falls der Inhalt von %10 ungleich Null ist. Nach Befehlsausführung wird das Zählregister dekrementiert
30 20	JP @%20	Die Register %20 und %21 bilden ein Doppelregister, dessen Inhalt die Sprungadresse ist. Es existert keine Sprungbedingung
D6 10 00	CALL %1000	Nach dem Op.-Code für den Unterprogrammaufruf steht die Eintrittsadresse in der Reihenfolge H-Teil, L-Teil
D4 20	CALL @%20	In den Registern %20 und %21 steht die 16-Bit-Adresse des aufzurufenden Unterprogramms

CPU-Steuerbefehle

NOP Keine Operation (no operation)

Die Abarbeitung des Befehls veranlaßt keine Reaktionen bezüglich des Programmablaufs. Die weitere Programmabarbeitung wird lediglich entsprechend der Befehlsausführungszeit verzögert.

Flags: unbeeinflußt

SRP Registerpointer setzen (set registerpointer)

Der Registerpointer RP wird mit einer Konstanten geladen, wobei nur die höherwertige Tetrade des geladenen 8-Bit-Wertes relevant ist. Diese 4 Bit stellen den höherwertigen Teil der 8-Bit-Registeradressen bei Arbeitsregisteradressierung dar.

Flags: unbeeinflußt

EI Globale Interruptfreigabe (enable interrupt)

Bit D7 des Interrupt-Maskierungsregisters wird auf 1 gesetzt. Die Interrupts sind damit global freigegeben.

Flags: unbeeinflußt

DI Interruptverbot (disable interrupt)

Bit D7 des Interrupt-Maskierungsregisters wird auf 0 gesetzt. Interrupts sind damit global gesperrt.

Flags: unbeeinflußt

SCF Setzen des Carryflags (set carry flag)

Nach Befehlsausführung ist das Carryflag gesetzt. Andere Flags bleiben unbeeinflußt.

RCF Rücksetzen des Carryflags (reset carry flag)

Nach Befehlsausführung ist das Carryflag zurückgesetzt. Andere Flags bleiben unbeeinflußt.

CCF Komplementieren des Carryflags (complement carry flag)

Nach Befehlsausführung ist das Carryflag negiert. Andere Flags bleiben unbeeinflußt.

6.4. Zusammenfassende Befehlsangabe

Befehl	Ziel	Quelle	1.Byte	2.Byte	3.Byte	C	Z	S	V	D	H
a) Ladebefehle											
LD Z,Q	Rn	#D	zC	D	-						
	Rn	nn	z8	Q	-						
	nn	Rn	q9	Z	-						
	@Rn	Rn	F3	zq	-						
	Rn	@Rn	E3	zq	-						
	nn	nn	E4	Q	Z						
	nn	@nn	E5	Q	Z						
	nn	#D	E6	Z	D						
	@nn	#D	E7	Z	D						
	@nn	nn	F5	Q	Z						
	Rn	nn(Rn)	C7	zx	Q						
	nn(Rn)	Rn	D7	qz	Z						
CLR Z	nn	-	B0	Z	-						
	@nn	-	B1	Z	-						
LDC Z,Q	Rn	@RRn	C2	zq	-						
	@RRn	Rn	D2	qz	-						
LDE Z,Q	Rn	@RRn	82	zq	-						
	@RRn	Rn	92	qz	-						
LDCI Z,Q	@Rn	@RRn	C3	zq	-						
	@RRn	@Rn	D3	qz	-						
LDEI Z,Q	@Rn	@RRn	83	zq	-						
	@RRn	@Rn	93	qz	-						

Befehl	Ziel	Quelle	1.Byte	2.Byte	3.Byte	C	Z	S	V	D	H
PUSH Q	nn	-	70	Q	-	.	.	.	.	.	.
	@nn	-	71	Q	-	.	.	.	.	.	.
POP Z	nn	-	50	Z	-	.	.	.	.	.	.
	@nn	-	51	Z	-	.	.	.	.	.	.

b) Arithmetik- und Logikbefehle

ADD Z,Q	Rn	Rn	02	zq	-	!	!	!	!	0	!
	Rn	@Rn	03	zq	-	!	!	!	!	0	!
	nn	nn	04	Q	Z	!	!	!	!	0	!
	nn	@nn	05	Q	Z	!	!	!	!	0	!
	nn	#D	06	Z	D	!	!	!	!	0	!
	@nn	#D	07	Z	D	!	!	!	!	0	!
ADC Z,Q	Rn	Rn	12	zq	-	!	!	!	!	0	!
	Rn	@Rn	13	zq	-	!	!	!	!	0	!
	nn	nn	14	Q	Z	!	!	!	!	0	!
	nn	@nn	15	Q	Z	!	!	!	!	0	!
	nn	#D	16	Z	D	!	!	!	!	0	!
	@nn	#D	17	Z	D	!	!	!	!	0	!
SUB Z,Q	Rn	Rn	22	zq	-	!	!	!	!	1	!
	Rn	@Rn	23	zq	-	!	!	!	!	1	!
	nn	nn	24	Q	Z	!	!	!	!	1	!
	nn	@nn	25	Q	Z	!	!	!	!	1	!
	nn	#D	26	Z	D	!	!	!	!	1	!
	@nn	#D	27	Z	D	!	!	!	!	1	!
SBC Z,Q	Rn	Rn	32	zq	-	!	!	!	!	1	!
	Rn	@Rn	33	zq	-	!	!	!	!	1	!
	nn	nn	34	Q	Z	!	!	!	!	1	!
	nn	@nn	35	Q	Z	!	!	!	!	1	!
	nn	#D	36	Z	D	!	!	!	!	1	!
	@nn	#D	37	Z	D	!	!	!	!	1	!
DA Z	nn	-	40	Z	-	!	!	!	X	.	
	@nn	-	41	Z	-	!	!	!	X	.	
CP Z,Q	Rn	Rn	A2	zq	-	!	!	!	!		
	Rn	@Rn	A3	zq	-	!	!	!	!		
	nn	nn	A4	Q	Z	!	!	!	!		
	nn	@nn	A5	Q	Z	!	!	!	!		
	nn	#D	A6	Z	D	!	!	!	!		
	@nn	#D	A7	Z	D	!	!	!	!		
INC Z	nn	-	20	Z	-		!	!	!		
	@nn	-	21	Z	-		!	!	!		
	Rn	-	zE	-	-		!	!	!		
DEC Z	nn	-	00	Z	-		!	!	!		
	@nn	-	01	Z	-		!	!	!		
INCW Z	nn	-	A0	Z	-		!	!	!		
	@nn	-	A1	Z	-		!	!	!		
DECW Z	nn	-	80	Z	-		!	!	!		
	@nn	-	81	Z	-		!	!	!		
AND Z,Q	Rn	Rn	52	zq	-		!	!	0		
	Rn	@Rn	53	zq	-		!	!	0		
	nn	nn	54	Q	Z		!	!	0		
	nn	@nn	55	Q	Z		!	!	0		
	nn	#D	56	Z	D		!	!	0		
	@nn	#D	57	Z	D		!	!	0		
OR Z,Q	Rn	Rn	42	zq	-		!	!	0		
	Rn	@Rn	43	zq	-		!	!	0		
	nn	nn	44	Q	Z		!	!	0		
	nn	@nn	45	Q	Z		!	!	0		
	nn	#D	46	Z	D		!	!	0		
	@nn	#D	47	Z	D		!	!	0		
XOR Z,Q	Rn	Rn	B2	zq	-		!	!	0		
	Rn	@Rn	B3	zq	-		!	!	0		
	nn	nn	B4	Q	Z		!	!	0		
	nn	@nn	B5	Q	Z		!	!	0		
	nn	#D	B6	Z	D		!	!	0		
	@nn	#D	B7	Z	D		!	!	0		

Befehl	Ziel	Quelle	1.Byte	2.Byte	3.Byte	C	Z	S	V	D	H
TM Z,Q	Rn	Rn	72	zq	-		!	!	0		
	Rn	@Rn	73	zq	-		!	!	0		
	nn	nn	74	Q	Z		!	!	0		
	nn	@nn	75	Q	Z		!	!	0		
	nn	#D	76	Z	D		!	!	0		
	@nn	#D	77	Z	D		!	!	0		
TCM Z,Q	Rn	Rn	62	zq	-		!	!	0		
	Rn	@Rn	63	zq	-		!	!	0		
	nn	nn	64	Q	Z		!	!	0		
	nn	@nn	65	Q	Z		!	!	0		
	nn	#D	66	Z	D		!	!	0		
	@nn	#D	67	Z	D		!	!	0		
COM Z	nn	-	60	Z	-	.	!	!	0	.	.
	@nn	-	61	Z	-	.	!	!	0	.	.
c) Rotations- und Verschiebebefehle											
RL Z	nn	-	90	Z	-	!	!	!	!		
	@nn	-	91	Z	-	!	!	!	!		
RLC Z	nn	-	10	Z	-	!	!	!	!		
	@nn	-	11	Z	-	!	!	!	!		
RR Z	nn	-	E0	Z	-	!	!	!	!		
	@nn	-	E1	Z	-	!	!	!	!		
RRC Z	nn	-	C0	Z	-	!	!	!	!		
	@nn	-	C1	Z	-	!	!	!	!		
SRA Z	nn	-	D0	Z	-	!	!	!	0		
	@nn	-	D1	Z	-	!	!	!	0		
SWAP Z	nn	-	F0	Z	-	X	!	!	X	.	.
	@nn	-	F1	Z	-	X	!	!	x	.	.
d) Programmsteuerbefehle											
JP @nn	nn	-	30	Z	-	.					
JP cc	DA	-	cD	DAH	DAL						
JR cc	RA	-	cB	RA	-						
DJNZ Rn	RA	-	nA	RA	-						
CALL @nn	nn	-	D4	Z	-						
	DA	-	D6	DAH	DAL						
RET	-	-	AF	-	-	.			.		.
IRET	-	-	BF	-	-	!	!	!	!	!	!
e) CPU-Steuerbefehle											
NOP	-	-	FF	-	-						
SRP	-	#D	31	D	-						
EI	-	-	9F	-	-						
DI	-	-	8F	-	-						
SCF	-	-	DF	-	-	1					
RCF	-	-	CF	-	-	0					
CCF	-	-	EF	-	-	!	.	.	.	.	.

6.5. Arbeitstabelle für 1. Byte des OP-Kodes

Arbeitsregister:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Befehl:																
LD Rn,nn	08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8
LD nn,Rn	09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9
LD nn,#D	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC
INC Rn	0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE
DJNZ Rn	0A	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA

Befehl	Code	Befehl	Code nn	Code @nn	Befehl	Code
LD Rn,@Rn	E3	RL	90	91	CALL @nn	D4
LD nn,nn	E4	RLC	10	11	CALL DA	D6
LD nn,@nn	E5	RR	E0	E1	RET	AF
LD nn,#D	E6	RRC	C0	C1	IRET	BF
LD @nn,#D	E7	SRA	D0	D1	EI	9F
LD Rn,nn(Rn)	C7	PUSH	70	71	DI	8F
LD nn(Rn),Rn	D7	POP	50	51	SCF	DF
LDC Rn,@RRn	C2	SWAP	F0	F1	RCF	CF
LDC @RRn,Rn	D2	CLR	B0	B1	CCF	EF
LDE Rn,@RRn	82	DA	40	41	NOP	FF
LDE @RRn,Rn	92	COM	60	61	SRP	31
LDCI @Rn,@RRn	C3	INC	20	21		
LDCI @RRn,@Rn	D3	DEC	00	01		
LDEI @Rn,@RRn	83	INCW	A0	A1		
LDEI @RRn,@Rn	93	DECW	80	81		

	Rn,Rn	Rn,@Rn	nn,nn	nn,@nn	nn,#D	@nn,#D
ADD	02	03	04	05	06	07
ADC	12	13	14	15	16	17
SUB	22	23	24	25	26	27
SBC	32	33	34	35	36	37
OR	42	43	44	45	46	47
AND	52	53	54	55	56	57
TCM	62	63	64	65	66	67
TM	72	73	74	75	76	77
CP	A2	A3	A4	A5	A6	A7
XOR	B2	B3	B4	B5	B6	B7

	--	Z	NZ	C	NC	PL	MI	OV	NOV	@nn
JR	8B	6B	EB	7B	FB	DB	5B	4B	CB	
JP	8D	6D	ED	7D	FD	DD	5D	4D	CD	30

7. Schaltungstechnik

7.1. Allgemeine Hinweise

Die Schaltungstechnik von Baugruppen mit Einchipmikrorechnern zeichnet sich gegenüber Einkartenrechnern auf der Basis des *U880*-Systems meist durch wesentlich verringerten Bauelementeaufwand aus.

Nachfolgend werden Grundschaltungen funktionsfähiger Rechnerbaugruppen mit den Schaltkreisen *U881/U882* beschrieben. Die später vorgestellten Programme sind mit diesen Schaltungen erprobt. Für den Anwender stellt sich häufig die Frage, welcher Auswahltyp der Palette *U881* bis *U886* für den speziellen Anwendungsfall besonders effektiv einsetzbar ist. Einige allgemeine Hinweise dazu sollen die Auswahl erleichtern. Mit der 64poligen Version *U882* lassen sich nahezu alle potentiellen Aufgabenstellungen lösen. Alle Ein-/Ausgabeleitungen sind uneingeschränkt verfügbar. Es ist weiterhin hervorzuheben, daß das 64polige QUIL-Gehäuse auch hinsichtlich der Leiterkartengestaltung Vorteile gegenüber den 40poligen Typen aufweist, da die Ein-/Ausgabetore sowie Adreß- und Datenleitungen des Speicheranschlusses sehr regelmäßig auf das Gehäuse verteilt sind. Baugruppen auf der Basis von Einebenen-Leiterkarten lassen sich daher meist realisieren. Der Hardwareaufwand ist bei Verwendung des *U882* in der Grundversion (d.h. ohne Speichererweiterung) stets kleiner als bei Verwendung einer 40poligen Version, da kein zusätzliches Adreßlatch benötigt wird. Die Notwendigkeit, einen Schaltkreis *U884* einzusetzen, besteht nur in wenigen Ausnahmefällen, wenn einerseits eine Beschränkung der Ein-/Ausgabeleitungen nicht in Kauf genommen werden kann und andererseits Programme von mehr als 2 KByte abzuarbeiten sind.

Oftmals muß auf die 40poligen Schaltkreisvarianten aufgrund der Verfügbarkeit ausgewichen werden. Außerdem sind diese Bauelemente deutlich billiger. Falls die spezielle Anwendung einen zusätzlichen Speicherbereich erfordert, ist der Einsatz der 64poligen Typen meist nicht sinnvoll, da die Zahl universell verfügbarer E/A-Leitungen in diesen Fällen ohnehin reduziert und die Einbindung des Programmspeichers in den externen Speicherbereich ohne nennenswerten Zusatzaufwand möglich ist. Bei typischen Ausbauvarianten mit 2 KByte EPROM und 1 KByte RAM werden für den Anschluß des externen Speichers Tor 1 und die niederwertigen 4 Bit von Tor 0 benötigt. Tor 2 und Tor 3 sowie die Torleitungen P04 bis P07 sind universell verwendbar.

Die gleichen Überlegungen gelten für den Einsatz des Anfalltyps *U886*, bei dem die interne ROM-Maske nicht zur Verfügung steht. Da die Absprungroutine in den externen Speicherbereich im Gegensatz zum *U883* im sogenannten Test-ROM untergebracht ist, muß die Erzeugung des RESET-Signals allerdings modifiziert werden (Spannungsüberhöhung auf etwa 8 V, s. auch Bild 7.7). Ein weiterer Unterschied zwischen dem *U883* und dem *U886* besteht darin, daß die Bootstrap-Routine des *U883* prinzipiell erweitertes Speicherzeitverhalten festlegt. Ein normales Speicherzeitverhalten kann im Programm über Modifizierung des P01M-Registers erreicht werden. Beim

U 886 ist dagegen nach RESET das Standardzeitverhalten festgelegt. Der im *U 883* untergebrachte BASIC-Interpreter hat in der Praxis keine große Bedeutung erlangt, so daß vom internen ROM nur die Bootstrap-Routine Verwendung findet, um die Programmabarbeitung ab Adresse %812 beginnen zu können. In dieser Hinsicht sowie in bezug auf die Eintragung der Absolutsprünge für eventuelle Interrupt-Behandlungen auf den Speicherplätzen %800 bis %811 unterscheiden sich *U 883* und *U 886* nicht.

Die Verwendung des *U 881* ist dann sinnvoll, wenn das implementierte Programm auf der ROM-Maske einen Absprung in den externen Speicherraum ermöglicht. In diesem Falle kann der Einsatz alternativ zum *U 883* erfolgen. Schaltkreise *U 881* mit dem aufgedruckten Bitmuster 004 erfüllen beispielsweise die genannte Forderung.

Bei allen Typen mit externem Programmspeicher ist zu beachten, daß die Bitbelegung des Moderegisters für Tor 0 und Tor 1 (P01M) nicht vom laufenden Programm in unzulässiger Weise verändert wird, da sonst eine kontrollierte Programmabarbeitung nicht mehr möglich ist.

Die zu allen Grundtypen vorhandenen power-down-Versionen kommen vorzugsweise zur Anwendung, wenn die Aufgabenstellung den Datenerhalt des Registersatzes bei abgeschalteter Betriebsspannung fordert. Grundsätzlich lassen sich jedoch auch die power-down-Typen universell verwenden, wenn der zusätzliche Hardwareaufwand in Kauf genommen wird. Bei nicht genutztem power-down-Mode muß der Stützspannungseingang fest mit der Betriebsspannung verbunden sein.

7.2. U882 ohne externen Speicher

Die Schaltung nach Bild 7.1 stellt hinsichtlich des Hardwareaufwands die einfachste

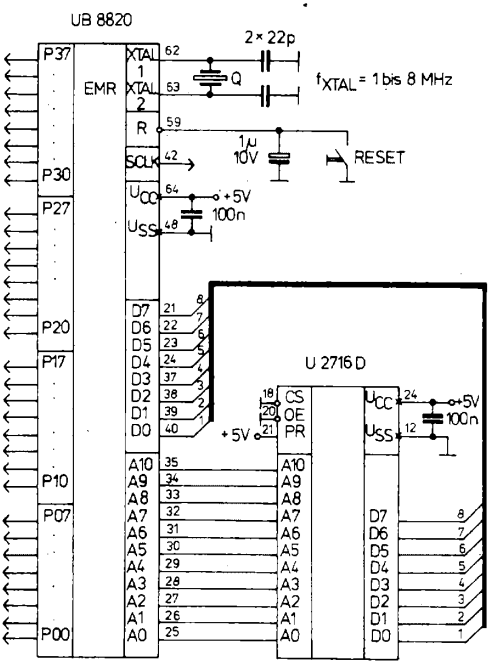


Bild 7.1
U 882 ohne externe Speicherbauelemente (Grundschaltung)

Variante dar. Sie ist immer dann vorteilhaft einsetzbar, wenn die Entwicklungsversion *U882* verfügbar ist und kein externer Speicherbereich zusätzlich benötigt wird. Alle 4 Ein-/Ausgabetore stehen dem Anwender uneingeschränkt zur Verfügung. Die Takterzeugung mit dem an die Eingänge XTAL1 und XTAL2 angeschlossenen Schwingquarz entspricht der Standardschaltung.

7.3. U882 mit 0,5 KByte RAM

Die Struktur der Einchipmikrorechner *U881/U882* ermöglicht den Anschluß externer Speicher. Dazu müssen die Tore 0 und 1 in Abhängigkeit der zu adressierenden Speichergröße so konfiguriert werden, daß sie die erforderliche Anzahl von Adreßleitungen stellen können.

Eine mögliche Unterscheidung zwischen Daten- und Programmspeicher über das Steuersignal /DM (data memory select) erlaubt die nochmalige Verdoppelung des externen Speicherraumes (s. auch Bild 2.3). Das Signal /DM muß in diesem Falle durch entsprechende Programmierung an der Torleitung P34 aktiviert (Bits D3 und D4 in Register R 247) und in die Adreßdekodierung der externen Speicherbauelemente einbezogen werden. Bleibt /DM unbenutzt, ist der externe Speicherraum auf 62 KByte begrenzt und die Wirkung der Befehle LDC und LDE sowie LDCI und LDEI identisch.

Tor 1 wird immer als Adreß-/Datenbus im Multiplexbetrieb für die Adressen A0 bis A7 und die Daten D0 bis D7 verwendet. Damit ergibt sich schon bei ausschließlicher Nutzung dieses Tores ein externer Speicherraum von 256 Bytes bzw. 512 Bytes unter Verwendung von /DM. Diese Konfiguration ist für Schaltungen typisch, bei denen eine Erweiterung des internen Registersatzes durch einen externen Schreib-Lese-Speicher gewünscht wird, ohne daß eine zu starke Einschränkung der verfügbaren Ein-/Ausgabeleitungen erfolgt. In Bild 7.2 ist eine entsprechende, besonders für den *U882* geeignete Schaltungsvariante angegeben.

Die benutzten statischen Schreib-Lese-Speicher *U214* mit einer Kapazität von 1 k × 4 Bit werden in dieser Schaltung zwar nur zu 25 % bzw. 50 % ausgenutzt, Tor 0

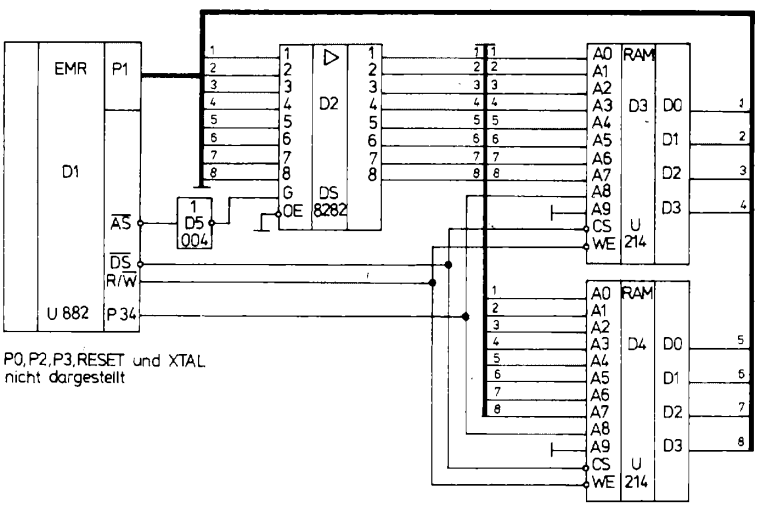


Bild 7.2
U882 mit 0,5 KByte RAM

bleibt aber für universelle Anwendung frei. Eine vollständige Nutzung des externen RAM läßt sich durch Verbinden der Adreßleitungen A9 der beiden *U214* mit einer freien Ausgabeleitung des EMR in Abhängigkeit vom konkreten Einsatzfall erreichen. Durch das Programm muß dann vor einem externen Speicherzugriff die mit A9 auszuwählende Speicherbank adressiert werden.

Größere Ausbaustufen des externen Speichers lassen sich durch Einbeziehung des niederwertigen Halbbytes von Tor 0 (P00 bis P03) als A8 bis A11 (s. Abschnitt 7.4.) bzw. des gesamten Tores 0 (A8 bis A15) realisieren.

7.4. Schaltung mit U883 und 2 KByte externem Programmspeicher

Diese Schaltung stellt die Alternative zu der in Abschnitt 7.2. vorgestellten Variante unter Einbeziehung des EMR-Schaltkreises *U883* dar. Da der interne Speicher für allgemeine Anwendung nicht nutzbar ist (BASIC-Interpreter!), muß das Programm auf einem externen Speicher abgelegt werden. Die Bootstrap-Routine des *U883* sichert einen ordnungsgemäßen Programmanlauf, wenn das 1. Befehlsbyte auf der Adresse %812 steht. Von Adresse %800 an können maximal 6 Sprungbefehle (JP DA) eingetragen werden, die auf die zugehörigen Interrupt-Behandlungsprogramme weisen. Prinzipiell ist auch die Verwendung von Relativsprüngen zur Interruptbehandlung möglich, wenn die Sprungdistanz in den zulässigen Grenzen bleibt. Praktisch ist diese Tatsache jedoch bedeutungslos, da für die Sprungbefehle stets 3 Byte reserviert sind.

Da die niederwertigen 8 Adreßbits sowie die Datenbits zeitmultiplex über Tor 1 ausgegeben werden, ist bei allen 40poligen Schaltkreisvarianten stets ein zusätzlicher Pufferspeicher (Adreßlatch) für den externen Speicherbereich erforderlich, um die Speicheradresse während eines Schreib- oder Lesezugriffs stabil zu halten. Die höherwertigen Adreßbits können dagegen direkt von Tor 0 an die entsprechenden Anschlüsse des Speichers geführt werden. Damit stehen für allgemeine Anwendungen das gesamte Tor 1 sowie die Torleitungen P00 bis P03 nicht mehr zur Verfügung. Die

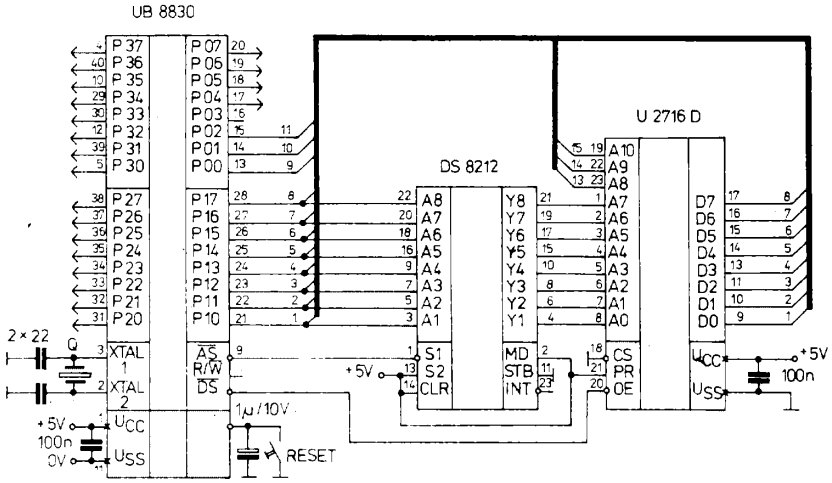


Bild 7.3 U883 mit 2 KByte ROM als Programmspeicher

genannten Leitungen dienen ausschließlich der Bedienung des externen Speichers. In der Schaltung nach Bild 7.3 ist als Adreßlatch ein *DS 8212* eingesetzt. Die Verwendung dieses Schaltkreises ist als Kompromiß anzusehen. Die elegantere Lösung besteht im Einsatz des *DS 8282,* der jedoch nicht immer zur Verfügung steht. Nachteilig bei der Verwendung des *DS 8282* als Adreßlatch ist allerdings die Notwendigkeit einer zusätzlichen Negation des Signals /AS (Adress Strobe), da die Datenübernahme bei diesem Schaltkreis mit der LH-Flanke am Eingang STROBE erfolgt.

7.5. *U 883 mit 2 KByte Programmspeicher und 1 KByte Schreib-Lese-Speicher*

Die Schaltung nach Bild 7.4 stellt eine typische Ausbauvariante dar. Programme von 2 KByte Länge reichen zur Lösung EMR-spezifischer Aufgabenstellungen meist aus, jedoch ergeben sich bei der ausschließlichen Nutzung des Universalregistersatzes als Operativspeicher häufig Probleme infolge der geringen Zahl der so verfügbaren «RAM»-Zellen. Die Notwendigkeit, das gesamte Programm mit einer minimalen Zahl benutzter Register abzuarbeiten, um Speicherplatz für Zwischen- und Endergebnisse frei zu halten, kompliziert die Software und kann auch eine Verlängerung der Programmlaufzeit zur Folge haben. In diesen Fällen ist es besser, einen zusätzlichen Schreib-Lese-Speicher (RAM) vorzusehen. Im Gegensatz zum *U 882* ist der zusätzliche

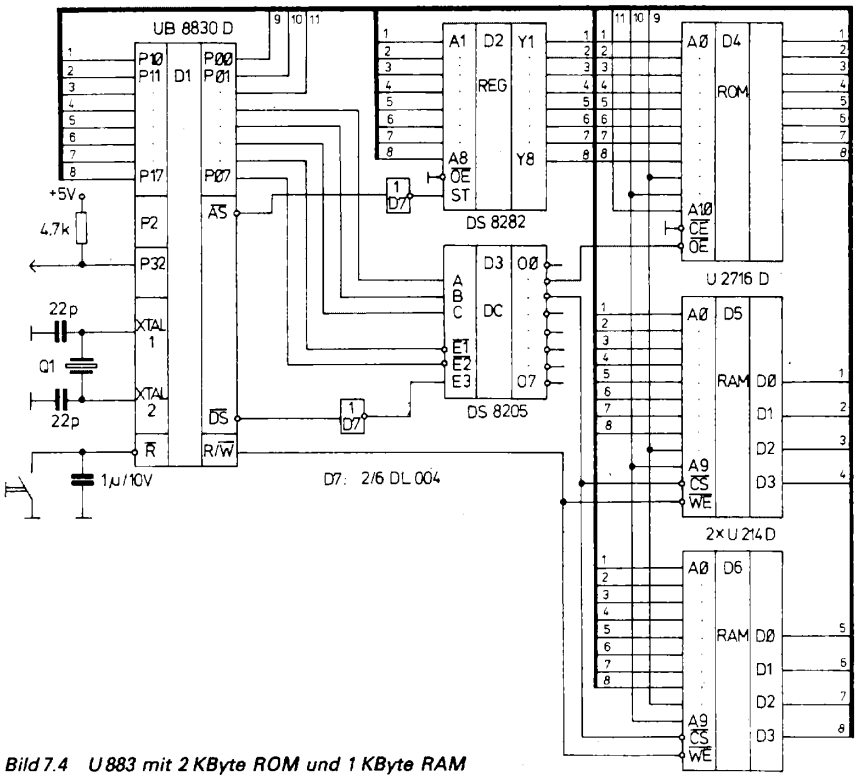


Bild 7.4 U 883 mit 2 KByte ROM und 1 KByte RAM

Hardwareaufwand dafür unerheblich, da die entsprechenden Leitungen von Tor 0 und 1 sowieso für die Bedienung des externen Programmspeichers konfiguriert sein müssen. Es ist demnach lediglich die Dekodierung eines zusätzlichen Adreßbereiches erforderlich. Der in der Schaltung für diesen Zweck eingesetzte 1-aus-8-Dekoder *DS 8205* wird nicht voll ausgenutzt. Die noch freien Dekoderausgänge lassen eine weitere Vergrößerung der Kapazität des externen Speichers zu. Sie können jedoch auch zur Erweiterung der Ein-/Ausgabemöglicheiten herangezogen werden und aktivieren dann entsprechende E/A-Tore (*DS 8282, DS 8212, U 40098* o.ä.). Ein- bzw. Ausgabeoperationen sind in diesem Falle als Speicherlese- bzw. Speicherschreibbefehle zu behandeln.

In der Schaltung nach Bild 7.4 ist der RAM-Bereich ab Adresse %1000 verfügbar. Der Programmspeicher belegt den Bereich %800 bis %FFF. Das Signal /DM ist nicht generiert. Daher kann das Lesen und Beschreiben des Speichers gleichermaßen mit den Befehlen LDC und LDE bzw. LDCI und LDEI erfolgen.

7.6. Besonderheiten bei der Erzeugung des RESET-Signals

In den bisher vorgestellten Schaltungen wird das RESET-Signal durch eine Minimalbeschaltung erzeugt. Sie besteht aus der Parallelschaltung eines Kondensators mit dem RESET-Taster. Die Schaltungsfunktion ist nur in Zusammenhang mit einem in integrierter Form auf dem Chip vorhandenen Pull-up-Widerstand vom RESET-Anschluß gegen die Betriebsspannung verständlich.

Nach dem Zuschalten der Versorgungsspannung bewirkt die Zeitkonstante des *RC*-Glieds einen verzögerten Spannungsanstieg am RESET-Eingang (power-on-RESET). Für höhere Ansprüche muß zur Erzeugung des RESET-Signals zusätzlicher Hardwareaufwand getrieben werden. Schwierigkeiten beim Programmanlauf sind vor allem dann zu erwarten, wenn die Anstiegsgeschwindigkeit der Versorgungsspannung nach dem Zuschalten nicht ausreichend groß ist. In diesem Falle läßt sich nicht mehr garantieren, daß das RESET-Signal noch für die geforderte Zahl von Taktperioden aktiv bleibt, nachdem die Versorgungsspannung ihren Minimalwert erreicht hat. Die RESET-Erzeugung durch einen zusätzlich vorhandenen Taster kann bei einigen Aufgabenstellungen nicht als gangbare Lösung angesehen werden, in anderen Fällen verbietet sich diese Variante von selbst, z.B. dann, wenn die EMR-Baugruppe normalerweise für den Bediener nicht zugänglich ist.

In Bild 7.5 ist eine Schaltung dargestellt, die unabhängig von der Anstiegsgeschwindigkeit der Versorgungsspannung ein RESET-Signal erzeugt. Auch nach einem Spannungseinbruch auf der Versorgungsleitung, dessen Dauer und zeitlicher Verlauf beliebig sein können, kann auf diese Weise ein erneuter Programmablauf erreicht werden.

Der aus 2 *DL 000*-Gattern bestehende Schwellwertschalter schaltet durch den Spannungsteiler am Eingang erst bei einer Versorgungsspannung größer als 4,5 V ein und

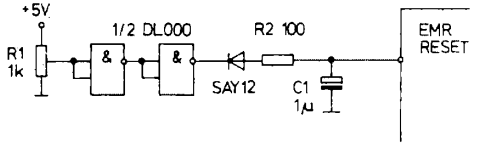


Bild 7.5
Power-on-Resetschaltung für beliebige Anstiegsgeschwindigkeit der Betriebsspannung

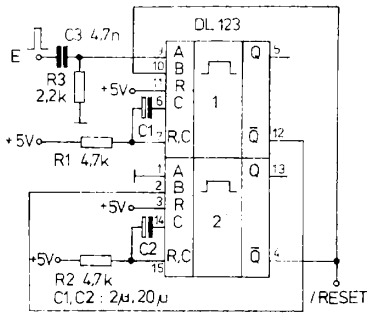


Bild 7.6
Programmlauf-Überwachungsschaltung mit DL 123

hebt damit das RESET-Signal für den Rechner auf. *C1* verzögert diesen Vorgang, so daß die Zeitbedigungen für dieses Signal sicher eingehalten werden. *R2* begrenzt den Entladestrom von *C1* bei erneut aktiviertem RESET-Signal. Die RESET-Auslösung soll bei einer Versorgungsspannung erfolgen, mit der die Arbeitsfähigkeit des Rechners noch gewährleistet ist. Erfahrungsgemäß läßt sich dafür ein Minimalwert von 4 V veranschlagen, obwohl damit der vom Hersteller garantierte Betriebsspannungsbereich für den EMR bereits verlassen ist.

Bei einigen Anwendungen genügt es nicht, lediglich den Programmanlauf nach Zuschalten der Versorgungsspannung abzusichern, sondern es muß auch der Störfall, z. B. der Programmabsturz infolge eingefangener Störimpulse, mit berücksichtigt werden. Das gilt vor allem für Steuerungen, welche nicht ständig einer Überwachung unterliegen und bei denen ein Ausfall Folgeschäden auslösen kann. In derartigen Fällen kann eine Wachschaltung entsprechend Bild 7.6 zum Einsatz kommen.

Kernstück der Schaltung ist ein *DL 123,* der ohne Triggerimpulse am Eingang E als astabiler Multivibrator arbeitet. Er erzeugt L-Ausgangsimpulse mit einer Dauer von etwa 2 ms, deren Folgefrequenz mit *C1* auf die konkrete Aufgabenstellung zugeschnitten ist. Mit der angegebenen Dimensionierung beträgt sie etwa 20 Hz. Der Triggereingang E ist an eine freie Torleitung des Einchipmikrorechners anzuschließen.

Sobald die bei normaler Programmabarbeitung an E per Befehl ausgegebenen Triggerimpulse ausbleiben, kippt Monoflop 1 zurück und aktiviert dadurch Monoflop 2. Er bleibt für die Dauer seiner Haltezeit (etwa 40 ms) aktiv und liefert einen RESET-Impuls von ebenfalls 40 ms. Mit dem Rückkippen des Monoflop 2 wird Monoflop 1 erneut aktiviert. Treffen nun innerhalb der Haltezeit von Monoflop 1 Triggerimpulse an E ein, ist der Rechner «wach», ansonsten folgt erneut ein RESET.

Die Wirksamkeit der Schaltung hängt in großem Maße davon ab, wie die Erzeugung der Triggerimpulse in die Software eingebunden ist. Auf jeden Fall müssen die entsprechenden Befehle an einer Stelle des Programms stehen, die zyklisch durchlaufen wird. Die für die Wachfunktion verwendete Torleitung ist prinzipiell frei wählbar. Eine Ausnahme bildet P36, da ein Absturz des laufenden Programms häufig zur Ausgabe des Systemtakts auf dieser Leitung führt. Von der Wachschaltung werden die Taktimpulse als ein scheinbar störungsfreies Arbeiten des Rechners bewertet, was die Auslösung von RESET-Impulsen verhindert. Dementsprechend kann ein erneuter Programmanlauf nicht stattfinden, die Wachschaltung bleibt also wirkungslos.

Sollen der interne Test-ROM beim *U881* bzw. das Anwenderprogramm beim *U886* nach einem RESET angesprungen werden, ist eine Spannungsüberhöhung auf etwa

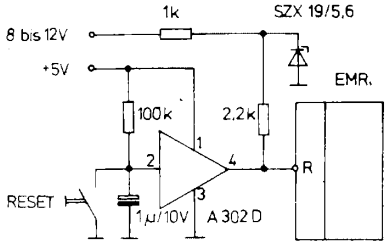


Bild 7.7
Schaltung zur Spannungsüberhöhung am RESET-Eingang für Ansprung des internen Test-ROM

8 V an diesem Eingang zu verwirklichen. Eine sehr einfache Schaltung dazu ist in Bild 7.7 dargestellt. Prinzipiell läßt sich die Spannungsüberhöhung auch über eine Gleichrichtung der Systemtaktfrequenz erreichen, indem diese Gleichspannung auf die Betriebsspannung von 5 V aufgestockt wird.

7.7. Erzeugung des Systemtakts

In Abhängigkeit von der konkreten Aufgabenstellung sind für die Erzeugung des Systemtakts verschiedene Varianten möglich. Als Standardschaltung anzusehen ist der Anschluß eines Schwingquarzes an die Eingänge des on-chip-Taktgenerators XTAL1 und XTAL2. Der Hersteller fordert außerdem je einen Kondensator von etwa 22 pF an den XTAL-Anschlüssen gegen Masse. Bei den bisher vorgestellten Schaltungen wurde daher stets diese Variante angegeben. Falls die Genauigkeit und Konstanz der Taktfrequenz nicht entscheidend für die ordnungsgemäße Funktion der EMR-Baugruppe sind, kann auf einen Quarz verzichtet und eine *LC*-Beschaltung zur Takterzeugung eingesetzt werden. Eine erprobte Schaltung dafür zeigt Bild 7.8. Die erforderliche Induktivität von etwa 20 µH wurde beim aufgebauten Muster durch eine sogenannte Entstördrossel realisiert. Mit der angegebenen Dimensionierung beträgt die interne Systemtaktfrequenz etwa 1 MHz, entsprechend einer Schwingfrequenz des Taktgenerators von 2 MHz. Besonders kleine Ansprüche in bezug auf die Konstanz der Taktfrequenz erlauben auch die Takterzeugung mit *RC*-Beschaltung an den XTAL-Eingängen. Bild 7.9 zeigt eine geeignete Schaltung.

Ein Messen der Taktfrequenz sollte grundsätzlich nicht direkt an den XTAL-Anschlüssen erfolgen, da das Ergebnis durch die Eingangskapazität des Frequenzmes-

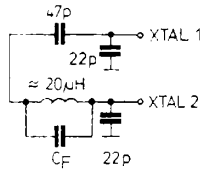


Bild 7.8
Erzeugung des Systemtakts mit LC-Beschaltung

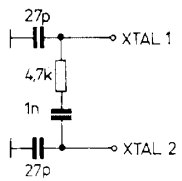
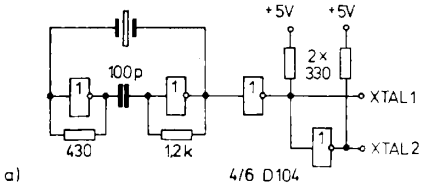
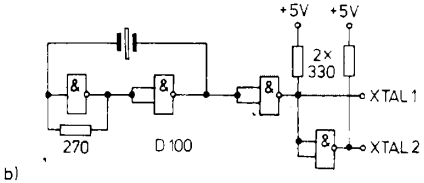


Bild 7.9
Erzeugung des Systemtakts mit RC-Beschaltung



a)



b)

Bild 7.10
Externe Takterzeugung mit Quarzgenerator

sers oder Oszillografen verfälscht wird. Im Extremfall ist auch ein Aussetzen des Generators bei zu großer externer Belastung möglich. Beim *U882* steht der Systemtakt immer am Ausgang SCLK zur Verfügung, während bei den 40poligen Versionen eine Ausgabe über P36 programmierbar ist.

Grundsätzlich kann bei allen Ableittypen der EMR *U881/U882* der Takt auch mit einem externen Generator erzeugt werden. Bei den Schaltkreistypen, die nicht die Möglichkeit des Datenerhalts durch eine externe Stützspannung zulassen, soll die Takteinspeisung an beiden XTAL-Eingängen unter Zwischenschaltung eines Negators erfolgen. Für die power-down-Varianten genügt ein Einphasentakt an XTAL1. Bild 7.10 zeigt 2 Varianten von Quarz-Generatorschaltungen für die externe Erzeugung des Systemtakts.

8. Ausbaumöglichkeiten der seriellen Schnittstelle des *U881/U882*

8.1. Struktur

Eine der wesentlichen Funktionsgruppen, die die EMR *U881/U882* gegenüber ähnlichen Einchiprechnern auszeichnen, ist die integrierte serielle Ein-/Ausgabeschnittstelle. Damit lassen sich bei geringem Hardwareaufwand Kopplungen und Vernetzungen von EMR dieser Familie realisieren.

Als Ein-/Ausgabeleitungen, bezeichnet mit RxD (Empfänger-Eingang) und TxD (Sender-Ausgang), werden die Leitungen P30 bzw. P37 durch Programmierung des P3M-Registers (D6 in Register 247) festgelegt.

Der Datenaustausch erfolgt über das SIO-Register 240, das sowohl ein lesbares Empfangsregister als auch ein beschreibbares Senderegister darstellt. Je nachdem, ob dieses Register in einem Befehl als Quelle oder Ziel auftritt, handelt es sich um einen Sende- oder Empfangsbefehl für den seriellen Kanal.

Die am EMR *U881/U882* vorhandene serielle Schnittstelle ist für Asynchronbetrieb im Duplexverfahren bei wahlweise ungerader Parität ausgelegt. Bild 8.1 zeigt das benutzte Datenformat. Beim Senden werden den Datenbytes immer ein Start(ST)- und 2 Stoppbits (SP) hinzugefügt, so daß ein Sendezeichen aus insgesamt 11 Bits besteht.

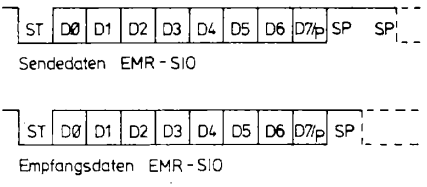


Bild 8.1
Datenformat der EMR-SIO

Im Falle der Übertragung mit ungerader Parität (programmierbar mit Bit D7 des Registers 247) wird das hochwertigste Datenbit D7 im Sendezeichen durch das Paritätsbit ersetzt. Das gilt es zu beachten, wenn z. B. Registerinhalte zu übertragen sind, bei denen das Datenbit D7 einen Teil der Information darstellt. Falls man nicht auf die zusätzliche Paritätskontrolle verzichten will, muß eine Konvertierung der Registerinhalte in einen 7-Bit-Kode (z. B. ASCII) vor dem Senden durchgeführt werden. Aus der Übertragung des Registerinhalts %FF entsteht so beispielsweise die Doppelbyteübertragung %46 %46 (ASCII-Kode für «F» = %46), der Registerinhalt %84 wird gesendet als Doppelbyte %38 %34. Die Verdoppelung der zu übertragenden Byteanzahl stört in den meisten Anwendungsfällen nicht, bringt aber durch die mögliche Paritätskontrolle einen Gewinn an Zuverlässigkeit.

Der serielle Empfänger erwartet einen Bitstrom von 10 Stellen je Zeichen. Das 2. Stopp-Bit wird demnach nicht ausgewertet. Bei programmiertem Empfangsformat

mit ungerader Parität ersetzt die SIO-interne Paritätsprüflogik das ursprünglich gesendete Paritätsbit D7 durch ein Flagbit zur Kennzeichnung von Paritätsfehlern (D7 = 1: Paritätsfehler, D7 = 0: kein Paritätsfehler). Das Empfangsprogramm muß also einen Testbefehl (z. B. TM) auf Bit D7 der empfangenen Daten durchführen, um Paritätsfehler zu erkennen.

Bei Benutzung der seriellen Schnittstelle des EMR ist immer die Reservierung des Zähler-/Zeitgeberkanals T0 als Taktgenerator notwendig.

Die Bitrate des Senders bzw. Empfängers ergibt sich zu

$$BR = f_{XTAL}/(128 \times PRE0 \times T0),$$

BR Bitrate in kBaud = kBit/s,
f_{XTAL} externe Systemtaktfrequenz in kHz,
PRE0 Vorteilerwert des Timers T0, einstellbar über R 245,
T0 Zählumfang des Timers T0, einstellbar über R 244,
128 Konstante, durch die interne SIO-Struktur bedingt.

Somit ergibt sich bei einer externen Systemtaktfrequenz von $f_{XTAL} = 8$ MHz eine maximale Bitrate BR = 62,5 kBaud.

Besonders sei an dieser Stelle auf die erforderliche Stabilität der Systemtaktfrequenz bei serieller Übertragung hingewiesen. Diese Frequenzkonstanz ist immer bei Verwendung von Schwingquarzen gegeben. Wird der Systemtakt dagegen mit *RC*- oder *LC*-Generatoren erzeugt, kann es durch Instabilitäten des Taktgenerators und damit verbundenen Abweichungen der Systemtaktfrequenzen von Sende- und Empfangseinheit gehäuft zu Übertragungsfehlern kommen. Bei größeren Abweichungen scheitert die Datenübertragung gänzlich. Man sollte deshalb schon in der Entwurfsphase einer EMR-Anordnung abwägen, ob eventuell später eine serielle Kopplung in Frage kommt, und in diesem Falle von Anfang an auf eine quarzstabilisierte Takterzeugung orientieren.

Die in der professionellen Rechentechnik genormten Standardbitraten von 110, 150, 300, 600, 1200, ..., 19200 Baud lassen sich exakt nur durch Verwendung einer Systemtaktfrequenz von $f_{XTAL} = 7{,}3728$ MHz realisieren. Da ein entsprechender Quarz den wenigsten Amateuren zur Verfügung steht, müssen mehr oder weniger große Abweichungen in Kauf genommen werden. Besonders bei Kopplungen mit industriell gefertigten Geräten kann es bei Gegenstellen, die sich nur auf genormte Bitraten einstellen lassen (z. B. Drucker, einige K1520-Leiterkarten) zu Problemen kommen.

8.2. Standardisierte Schnittstellen

Für einfache serielle Kopplungen von EMR untereinander sowie eines EMR mit dem seriellen Schaltkreis des *U 880*-Systems (SIO *U 856 D)* reicht es meist aus, die entsprechenden Sende- und Empfangsleitungen direkt miteinander zu verbinden. Die damit überbrückbaren Entfernungen sind jedoch aufgrund der geringen Treibereigenschaften der Sendeausgänge begrenzt. Abhilfe ist durch Verwendung von Leitungstreiber-Schaltkreisen oder diskret aufgebauten Transistorstufen möglich. Für bestimmte Anwendungsfälle ist eine galvanische Trennung der beteiligten Baugruppen notwendig. Sie ist durch Einschaltung von Optokopplern in den Übertragungsweg realisierbar. Für größere Entfernungen oder zur Ansteuerung peripherer Geräte der Kleinrechentechnik

(z. B. Drucker) werden im industriellen Einsatz standardisierte Schnittstellen genutzt, die auch für den Amateur empfohlen werden. Die beiden wichtigsten Serienschnittstellen sollen deshalb nachfolgend näher beschrieben werden.

8.2.1. IFSS-Schnittstelle

Das Interface zum sternförmigen Anschluß von Ein-/Ausgabeleitungen mit serieller Informationsübertragung (IFSS) ist in der Literatur auch unter dem Namen TTY-Schnittstelle oder 20-mA-Stromschleife zu finden. Das IFSS-Interface erlaubt eine asynchrone Übertragung nach dem Start-Stopp-Prinzip über Stromschleifen mit einer 4adrigen geschirmten Duplexverbindung. Der Informationsaustausch geschieht durch Übertragung serieller, binärer Signale mit wählbarer Geschwindigkeit, die für den konkreten Fall per Vereinbarung (Programmierung) festlegbar ist. Als übliche Übertragungsraten werden 9600, 4800, 2400, 1200, 600 und 300 Bit/s gewählt. Der Informationsaustausch ist in Bild 8.2 dargestellt. Die Symbole «+» und «−» kennzeichnen die Stromrichung in der Schleife, wobei die Stromrichtung von «+» nach «−» definiert ist. Dem logischen Zustand «1» in den Schleifen entspricht ein Strom von 15 bis 25 mA, während dem logischen Zustand «0» ein Strom von 0 bis 3 mA zugeordnet ist.

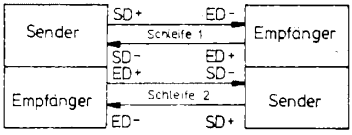


Bild 8.2
Informationsaustausch zwischen 2 Rechnern (IFSS-Prinzip)

Die Festlegungen laut Standard lassen wahlweise eine Stromeinspeisung in Schleife 1 und 2 auf beiden Seiten der Übertragungsanordnung zu. Der Ort der Stromeinspeisung kann dann nach Bild 8.3 über Schalter oder Drahtbrücken festgelegt werden.

Die Schaltung nach Bild 8.3a hat den Vorteil, daß die beteiligten EMR im Sende- und Empfangsteil bezüglich der Stromeinspeisung gleichartig beschaltet sind. Dagegen ist in der Schaltung nach Bild 8.3b die Stromeinspeisung nur auf einer Seite der seriellen Kopplung angeordnet. Für die Gegenstelle läßt sich auf diese Weise der kleinste schaltungstechnische Aufwand erzielen. Bei bestimmten Hardwarelösungen mit begrenzter Leiterkartengröße ist das von Vorteil.

In der Schaltung entsprechend Bild 8.3c sind mehrere Rechner zu einem seriellen Ringsystem verbunden. Die Stromeinspeisung geschieht in diesem Fall über eine gesonderte Baugruppe, so daß alle Rechner im sogenannten Passiv-Modus arbeiten. Zur Verkabelung des Ringes ist eine einadrig geschirmte Leitung (Koaxialkabel) ausreichend. Ein derartiger IFSR-Ring (Interface seriell ringförmig) bedingt einen zusätzlichen Softwareteil in jedem der beteiligten Rechner, welcher die gemeinsame Arbeit auf der Ringleitung ermöglicht. Diese Softwarekomponente dient der Klärung von Prioritätsproblemen und damit der Konfliktlösung bei gleichzeitigem Versuch mehrerer Rechner, Daten zu senden. Weiterhin muß eine Benennung der an der Übertragung beteiligten Einheiten (Kennbyte) vorgenommen werden. Grundsätzlich findet ein ständiges «Abhören» der auf dem Ring vorhandenen Informationen durch jede Einheit

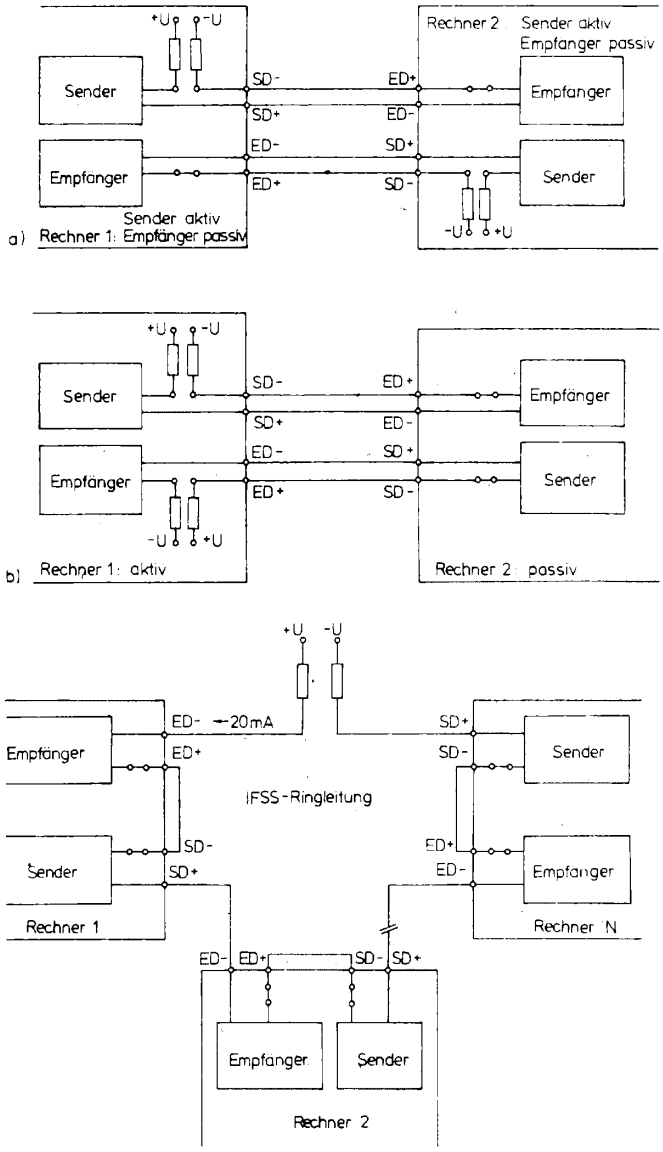


Bild 8.3
IFSS-Kopplungsvarianten

statt. Das Lesen der Daten erfolgt aber nur, wenn die entsprechende Kennung empfangen werden konnte. Die detaillierte Vorstellung eines derartigen Systems geht über den Rahmen dieses Buches hinaus, interessierte Leser seien auf Literatur zu lokalen Netzen (z.B. [12]) verwiesen. Für den Amateur ist die Kopplung von nur 2 Baugruppen mit serieller Schnittstelle die weit häufigere Problemstellung.

Die Bemessung der Signalstromquellen muß so erfolgen, daß bei Leerlauf oder Kurzschluß an den Ausgangsklemmen keine Bauelementeschäden auftreten. In der einfachsten Ausführungsform kann die Stromeinspeisung in die Schleifen über Festwiderstände erfolgen. Höhere Ansprüche (wechselnde Lastbedingungen, große Lei-

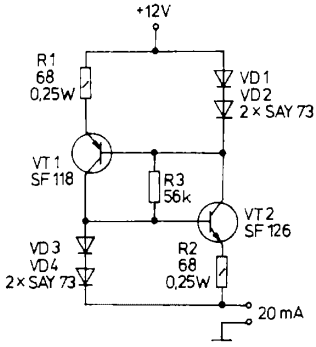


Bild 8.4
Konstantstromquelle für IFSS-Interface

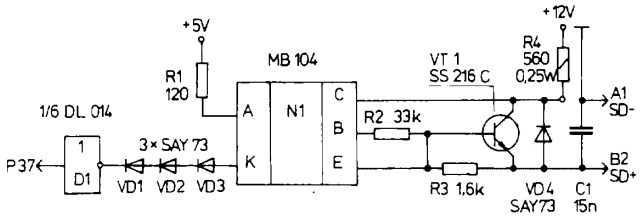


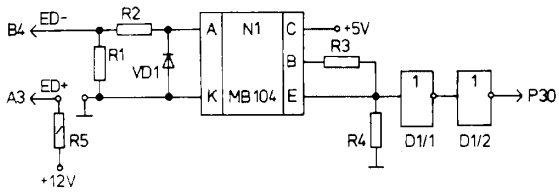
Bild 8.5
Standardschaltung des IFSS-Senders

tungslängen) erfordern die Einspeisung mit Konstantstromquellen. Bild 8.4 zeigt eine mögliche Realisierung für diesen Zweck. Durch die Stabilisierungsschaltung beider Parallelzweige können die geforderten Grenzwerte in bezug auf den Schleifenstrom bei Leitungswiderständen bis zu 200 Ω eingehalten werden. Bei Erhöhung der Teilnehmerzahl (z. B. IFSR-Ring) ist die Dimensionierung der Versorgungsspannung im Bild 8.4 so zu verändern, daß der geforderte Schleifenstrom von 20 mA über allen im Ring auftretenden Spannungsabfällen eingehalten wird.

Die Standardschaltung für den IFSS-Sender ist in Bild 8.5 dargestellt. Zur galvanischen Trennung dient ein Optokoppler. Der Innenwiderstand des Senders wird wesentlich durch den Arbeitswiderstand des Transistors bestimmt. Ausgangsseitig ist ein Kondensator (*C*1) zwischen die Übertragungsleitungen eingeschaltet, der ein Übersprechen in Nachbaradern des Übertragungskabels verhindern soll.

Die Standardschaltung des IFSS-Empfängers ist in Bild 8.6 dargestellt. Widerstände in der Leitung verbessern die Anpassung des Empfängers. Zur galvanischen Trennung dient wiederum ein Optokoppler. Mit dem nachgeschalteten Negator, der möglichst ein Typ mit Triggerverhalten sein sollte, wird ein TTL-gerechter Ausgangspegel zur Ansteuerung des nachfolgenden Eingangs erreicht.

Zur eindeutigen galvanischen Trennung der verbundenen Geräte ist der richtige An-



D1/1, D1/2 : DL 014 N1 : MB 104 VD1 : SAY 73 R1 : 270 Ohm
R2 : 22 Ohm R3 : 560k R4 : 360 Ohm R5 : 560 Ohm/0,25W

Bild 8.6
Standardschaltung des IFSS-Empfängers

Reihe	Kontakt-Nr.	Belegung	Bezeichnung
A	1	Sendedaten	SD-
B	2	Sendedaten	SD+
A	3	Empfangsdaten	ED+
B	4	Empfangsdaten	ED-
A	5	Schirm	

Tabelle 8.1 Steckverbinderbelegung IFSS

schluß der Abschirmung des Übertragungskabels von Bedeutung. Der Schirm darf nur einseitig, und zwar auf der Seite der Stromeinspeisung, angeschlossen sein.

Bei industriellen Geräten mit IFSS-Schnittstelle ist auch die konstruktive Ausführung der Verbindungselemente eindeutig geregelt. Es wird das Steckverbindersystem mit 5poliger Stecker- und Buchsenleiste nach TGL 29331/04 verwendet. Geräteseitig ist die Steckerleiste, kabelseitig die Buchsenleiste einzusetzen. Tabelle 8.1 zeigt die Zuordnung der einzelnen Signalleitungen zu den Kontakten des Steckverbinders.

Die mit dem Interface IFSS möglichen asynchronen Übertragungsformate müssen bei Benutzung des EMR *U 881/U 882* eingeschränkt werden. Das ist besonders bei Kopplungen von EMR mit dem Schaltkreis *U 856* der *U 880*-Peripherie zu beachten.

Es gelten folgende Einschränkungen:

Parameter	Möglichkeiten IFSS allgemein	Einschränkung bei U 881/ U 882
Anzahl Startbits	1	1
Anzahl Datenbits	5, 7, 8	8 (ohne Parität)
Parität	ohne, gerade, ungerade	ohne, ungerade
Anzahl Stoppbits	1, 1,5 , 2	2

8.2.2. V.24-Schnittstelle

Das Interface nach V.24 (auch RS 232 C genannt) ist eine serielle Schnittstelle mit gemeinsamer Masseverbindung. Sie wurde in der TGL 29077 standardisiert. Diese Schnittstelle ist in ihren Möglichkeiten vielseitiger als eine IFSS-Schnittstelle, da sie auch synchrone Datenübertragungen zuläßt. Die logischen Pegel werden durch folgende Spannungsbereiche repräsentiert:

Logikpegel «0»: 3 bis 15 V,

Logikpegel «1»: −3 bis −15 V.

In der praktischen Anwendung wird meist mit einer Schnittstellenspannung von +12 bzw. −12 V gearbeitet. Die Notwendigkeit der Bereitstellung von 2 Betriebsspannungen unterschiedlicher Polarität stellt gleichzeitig auch den größten Nachteil bei EMR-Applikationen dar. Eine V.24-Schnittstelle am EMR *U 881/U 882* sollte deshalb nur in begründeten Fällen, die den zusätzlichen Aufwand rechtfertigen, realisiert werden. Das kann beispielsweise der Fall sein, wenn Kopplungen von EMR-Baugruppen mit industriellen Geräten (Drucker, Büro- oder Personalcomputer als übergeordnete Rechner u. a.) gefordert sind und eine IFSS-Schnittstelle an diesen Gegenstellen nicht zur Verfügung steht.

Tabelle 8.3 gibt einen Überblick zu den definierten Schnittstellenleitungen. Da bei der *U 881/U 882*-Anwendung lediglich asynchrone Übertragungsverfahren zugelassen sind, müssen nicht alle Signale benutzt werden. Grundsätzlich resultiert die Einbezie-

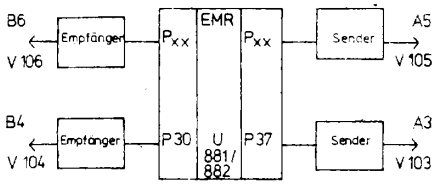
Reihe	Anschluß	Bezeichnung	Bedeutung	Richtung
A	1	V 102	Masse	
B	2	Schirm	Schirm	
A	3	V 103	Sendedaten	Ausgang
B	4	V 104	Empfangsdaten	Eingang
A	5	V 105	Aufforderung zum Senden	Ausgang
B	6	V 106	Bereit zum Senden	Eingang
A	7	V 107	Betriebsbereitschaft	Eingang
B	8	V 108	Anschalten der Datenendstelle	Ausgang
A	9	V 109	Empfangssignalpegel	Eingang
B	10	V 111	Wahl der Übertragungsgeschwindigkeit	Ausgang
A	11	V 113	Sendeschrittakt von Datenendeinrichtung	Ausgang
B	12	V 114	Sendetakt von Datenübertragungseinrichtung	Eingang
A	13	V 115	Empfangsschrittakt von Datenübertragungseinrichtung	Eingang

Tabelle 8.2 Steckverbinderbelegung V.24

hung der verschiedenen Steuerleitungen aus der konkreten schaltungstechnischen Ausführung der Gegenstelle.

Als standardisiertes Verbindungselement kommt bei der Schnittstelle V.24 ein 13poliger Steckverbinder (Bauform *103-13* nach TGL 29331/04) zum Einsatz. Für die notwendige Pegelwandlung zwischen Rechner und Schnittstelle besteht eine optimale Lösung in der Verwendung entsprechender Leitungstreiber *75150 (K170AP2)* bzw. *75154 (K170UP2)*.

Bei einer *U881/U882*-gerechten Modifikation der V.24-Schnittstelle müssen die Freiheitsgrade des entsprechenden Standards eingeschränkt werden. Als Übertragungsmodus kommt nur das asynchrone Verfahren mit wahlweiser ungerader Parität in Betracht. Eine entsprechende Minimalvariante ist in Bild 8.7 dargestellt. Für den Anschluß der Leitungen V105 und V106 müssen 2 freie E/A-Leitungen des EMR als Handshake-Steuerleitungen bereitgestellt werden.



V 103: Sendedaten
V 104: Empfangsdaten
V 105: Aufforderung zum Senden
V 106: Bereitschaft

Empfänger: Umsetzer ±12V/TTL (75154)
Sender: Umsetzer TTL/±12V (75150)

Bild 8.7
V.24-Schaltung

8.2.3. Serielle Kopplung zwischen 2 EMR

Unabhängig von der konkreten Auslegung der seriellen Schnittstelle zwischen den EMR (Direktkopplung, IFSS- oder V.24-Schnittstelle) erfolgt die Programmierung des EMR für serielle Datenübertragung. In diesem Abschnitt werden ein Testprogramm

(Programm 8.1) und ein Beispielprogramm zur Kopplung von 2 EMR über die Serienschnittstellen beschrieben.

Programm 8.1

```
Z8ASM      4.0
LOC     OBJ CODE    STMT SOURCE STATEMENT

                      1 SIOTEST MODULE
                      2
                      3         ! TESTPROGRAMM ZUM SENDEN UND EMPFANGEN UEBER DIE
                      4           SIO EINES EMR
                      5           DAZU KURZSCHLUSS DES SENDERS P37 MIT DEM EMPFAENGER P30
                      6           DAS PROGRAMM BESTEHT AUS:
                      7                 1. INITIALISIERUNG
                      8                 2. VORBEREITUNG EINES DATENMUSTERS IN DEN REGISTERN
                      9                    %20...%2F ZUM SENDEN
                     10                 3. SENDEN DES REGISTERINHALTES %20...%2F UND EMPFANG
                     11                    IN DEN REGISTERN %30...%3F
                     12                 4. ANZEIGE DER REGISTERINHALTE %30...%3F AN P2-LEDS
                     13         !
                     14
                     15 GLOBAL
                     16         MAIN PROCEDURE
                     17         ENTRY
                     18
P 0800               19 $ABS %800                       ! VARIANTE FUER U 883          !
                     20
P 0800 8D  088F      21         JP      DUMMY
P 0803 8D  088F      22         JP      DUMMY
P 0806 8D  088F      23         JP      DUMMY
P 0809 8D  087C      24         JP      SIOIN           ! INTERRUPT SIO-EMPFANG !
P 080C 8D  086D      25         JP      SIOOUT          ! INTERRUPT SIO-SENDER  !
P 080F 8D  088F      26         JP      DUMMY
                     27
                     28 ! ***************   1. INITIALISIERUNG EMR  ***************!
                     29
P 0812 8F            30 BEGIN:  DI
P 0813 E6  FF  80    31         LD      SPL,#%80        ! STACKPOINTER LADEN    !
P 0816 E6  FB  18    32         LD      IMR,#%18        ! IRQ3 UND IRQ4 FREI    !
P 0819 E6  F9  2F    33         LD      IPR,#%2F        ! SIOIN > SIOOUT        !
P 081C E6  F8  D7    34         LD      P01M,#%D7       ! P04...P07 = A12...A15
                     35                                   ZEITSTEUERUNG NORMAL
                     36                                   P10...P17 = AD0...AD7
                     37                                   STACK INTERN
                     38                                   P00...P03 = A8...A11   !
P 081F E6  F7  41    39         LD      P3M,#%41        ! OHNE PARITAET
                     40                                   P30= RxD  P37= TxD
                     41                                   P31...P33 = EINGAENGE
                     42                                   P34...P36 = AUSGAENGE
                     43                                   P2 MIT PULL-UP'S        !
P 0822 E6  F6  00    44         LD      P2M,#%0         ! P2 ALLES AUSGAENGE    !
P 0825 E6  02  00    45         LD      2,#%0           ! P2 RUECKSETZEN        !
P 0828 E6  F5  05    46         LD      PRE0,#%5        ! VORTEILER T0 = 1
                     47                                   AUTOMAT. NEUSTART       !
P 082B E6  F4  08    48         LD      T0,#8           ! ZEITKONSTANTE T0      !
                     49
                     50 ! ***************** 2. LADEN DES MUSTERS *****************!
                     51
P 082E 31  10        52         SRP     #%10            ! REGISTERPOINTER SETZEN!
P 0830 0C  20        53         LD      R0,#%20         ! R0 ALS ZEIGER         !
P 0832 1C  40        54         LD      R1,#%40         ! R1 ALS MUSTER         !
P 0834 F3  01        55 MUSTER: LD      @R0,R1          ! %20...%2F LADEN       !
P 0836 1E            56         INC     R1              ! NAECHSTES MUSTER      !
P 0837 0E            57         INC     R0              ! ZEIGER + 1            !
P 0838 A6  E0  30    58         CP      R0,#%30         ! 16 BYTE GELADEN ?     !
P 083B EB  F7        59         JR      NZ,MUSTER       ! NEIN                  !
                     60
                     61 ! ***************** 3. AUSLOESEN SENDEN/EMPFANG  *********!
                     62
P 083D 1C  20        63         LD      R1,#%20         ! SENDEZEIGER LADEN     !
P 083F 0C  30        64         LD      R0,#%30         ! EMPFANGSZEIGER LADEN  !
P 0841 2C  00        65         LD      R2,#%0          ! FERTIGMELDEBYTE EMPFANG!
P 0843 E6  F1  03    66         LD      TMR,#%3         ! T0 LADEN UND STARTEN  !
P 0846 9F            67         EI
P 0847 A6  E2  AA    68 SIOEND: CP      R2,#%AA         ! WARTEN AUF ENDEMELDUNG!
P 084A EB  FB        69         JR      NZ,SIOEND       ! VOM SIO-EMPFAENGER    !
P 084C D6  0860      70         CALL    WARTEN
                     71
                     72 ! ***************** 4. ANZEIGE DER EMPFANGENEN BYTES AN P2!
                     73
P 084F 0C  30        74 ANZEIG: LD      R0,#%30         ! ZEIGER LADEN          !
P 0851 E5  E0  02    75 ANZEI1: LD      2,@R0           ! BYTE AN P2 AUSGEBEN   !
P 0854 D6  0860      76         CALL    WARTEN
```

```
P 0857 0E              77          INC     R0
P 0858 A6  E0  40      78          CP      R0,#%40         ! ALLE BYTES ANGEZEIGT ?!
P 085B EB  F4          79          JR      NZ,ANZEI1       ! NEIN                  !
P 085D 8D  0812        80          JP      BEGIN           ! NEUSTART              !
                       81
                       82 ! ***************  UNTERPROGRAMM :  WARTEN  ************!
                       83
P 0860 FC  00          84 WARTEN: LD       R15,#%0         ! ZAEHLER1 = ARB.-REG.15!
P 0862 EC  00          85          LD      R14,#%0         ! ZAEHLER2 = ARB.-REG.14!
P 0864 DC  04          86          LD      R13,#4          ! ZAEHLER3 = ARB.-REG.13!
P 0866 FA  FE          87 WARTE1: DJNZ     R15,WARTE1
P 0868 EA  FC          88          DJNZ    R14,WARTE1
P 086A DA  FA          89          DJNZ    R13,WARTE1
P 086C AF              90          RET
                       91
                       92 ! ***************  INTERRUPTBEHANDLUNG SIO-SENDER *******!
                       93
P 086D A6  11  30      94 SIOOUT: CP       %11,#%30        ! ALLE BYTES GESENDET ? !
P 0870 6B  06          95          JR      Z,SIO1          ! JA                    !
P 0872 E5  11  F0      96          LD      SIO,@%11        ! NAECHSTES BYTE SENDEN !
P 0875 20  11          97          INC     %11             ! SENDEZEIGER + 1       !
P 0877 BF              98          IRET
P 0878 E6  FB  08      99 SIO1:   LD       IMR,#%8         ! NUR NOCH IRQ3 FREI    !
P 087B BF             100          IRET
                      101
                      102 ! *************  INTERRUPTBEHANDLUNG SIO-EMPFAENGER *****!
                      103
P 087C F5  F0  10     104 SIOIN:  LD       @%10,SIO        ! BYTE ABSPEICHERN      !
P 087F 20  10         105          INC     %10
P 0881 A6  10  40     106          CP      %10,#%40        ! ALLE BYTES EMPFANGEN ?!
P 0884 6B  01         107          JR      Z,SIO2          ! JA                    !
P 0886 BF             108          IRET
P 0887 E6  FB  00     109 SIO2:   LD       IMR,#%0         ! KEINE INTERRUPTS MEHR !
P 088A E6  F1  00     110          LD      TMR,#%0         ! TO STOPPEN            !
P 088D 2C  AA         111          LD      R2,#%AA         ! ENDEKENNUNG FUER HP   !
P 088F BF             112 DUMMY:  IRET
                      113
P 0890                114          END MAIN
                      115 END SIOTEST

   0 errors
Assembly complete
%
```

Programmbeschreibung SIOTEST

Bevor eine Inbetriebnahme der seriellen Übertragungsstrecke erfolgen kann, ist ein Test der beteiligten Schittstellen sinnvoll. Das Testprogramm SIOTEST ist für diesen Zweck geeignet. Die Leitungen P37 (Sender) und P30 (Empfänger) bzw. die entsprechenden Interfaceleitungen werden während des Tests kurzgeschlossen. Am Programmbeginn steht die Initialisierung der Steuerregister:

- P3M zur seriellen Übertragung,
- IMR zur Freigabe von Sender- und Empfängerinterrupt,
- PRE0 und T0 zur Festlegung der Übertragungsrate.

Im darauffolgenden Programmabschnitt wird in den Registern %20 bis %2F ein Testbyte-Muster generiert, im Beispiel die Bytes %40 bis %4F. Nach Einstellung von Sende- und Empfangszeiger (je ein Universalregister) erfolgt der Start des Zeitgeberkanals T0 zur Bildung des Sende- und Empfangstakts. Alle weiteren Aktivitäten unterliegen dann der Steuerung von Senderinterrupt SIOOUT (Bedingung: Sender leer) bzw. Empfängerinterrupt SIOIN (Bedingung: Zeichen empfangen).

Bei Auftreten des Senderinterrupts SIOOUT wird geprüft, ob bereits alle Bytes gesendet sind. Ist das der Fall, kommt es zur Sperrung aller weiteren Senderinterrupts. Andernfalls wird das durch den Sendezeiger markierte Byte an den Sender ausgegeben und der Sendezeiger nachfolgend inkrementiert.

Das Auftreten eines Empfängerinterrupts SIOIN hat die Eintragung des empfangenen Bytes in das durch den Empfangszeiger markierte Register zur Folge. Nach dem Empfang aller Bytes wird der Zeitgeberkanal T0 gestoppt und dem Hauptprogramm eine Fertigmeldung (im Beispiel der Inhalt %AA in Register R2) übergeben. Das Haupt-

programm setzt daraufhin mit einer visuellen Anzeige aller empfangenen Bytes am Tor 2 fort, wobei das Unterprogramm WARTEN eine Zeitschleife zwischen den Ausgaben realisiert.

Die Kopplung von 2 EMR über eine serielle Schnittstelle soll exemplarisch mit den Programmen 8.2 und 8.3 (SIO-EMR1, SIO-EMR2) demonstriert werden. Das Beispiel sieht die Übertragung eines Datenblocks von 16 Bytes (ASCII-Zeichen), der zusätzlich von einem Startbyte (%AA) sowie einem Stoppbyte (%1E) umrahmt wird, von EMR 1 an den EMR 2 vor. Als Quittung gilt die Rücksendung aller empfangenen Daten zum EMR 1. Die Übertragung von ASCII-Zeichen (7 Bit) ermöglicht die Nutzung der Paritätskontrolle während des Datentransfers. Sollen 8-Bit-Daten genutzt werden, müßte die Initialisierung beider EMR (Register 247) entsprechend geändert werden.

Programmbeschreibung SIO-EMR1

Im Programm SIO-EMR1 (Programm 8.2) sind alle notwendigen Aktivitäten für EMR 1 aufgelistet. Bei praktischen Anwendungen kann dieser Rechner beispielsweise zur Meßwerterfassung dienen. Die vorverdichteten Daten werden als Block an EMR 2 übergeben, die Quittungssignale empfangen und mit den gesendeten Daten verglichen.

Am Anfang des Programms werden wiederum die Steuerregister initialisiert. Das Unterprogramm LADEN erzeugt 16 «Meßwerte» in den Registern %30 bis %3F. In einer konkreten Anwendung müßte dieses Unterprogramm durch ein Umlagern der tatsächlichen Meßwerte in den Registerbereich %30 bis %3F ersetzt werden.

Das Unterprogramm SENDEN initialisiert zunächst die SIO und gibt anschließend das Startbyte an den Sender aus. Mit dem Laden und Starten des Zeitgebers T0 beginnt die serielle Ausgabe an Leitung P37. Alle weiteren Aktivitäten des Senders erledigt das Interruptprogramm SIOOUT. Am Symbol WARTEN wird die Fortsetzung des Hauptprogramms unterbunden, solange noch nicht alle gesendeten Bytes wieder empfangen sind. Im Interruptprogramm SIOOUT (Bedingung: Sender leer) erfolgt die Ausgabe des durch den Sendezeiger (Register %1E) markierten Bytes. Anschließend wird der Sendezeiger inkrementiert und das eventuelle Erreichen des letzten Datenbytes geprüft.

Falls über Leitung P30 ein Zeichen empfangen werden konnte, löst die entsprechende Interruptforderung das Bedienprogramm SIOIN aus. Es überprüft den Senderahmen (Startbyte, Stoppbyte) und trägt das gelesene Byte in das durch den Empfangszeiger markierte Register ein. Durch Rücksetzen des IMR wird dem Hauptprogramm der Empfang aller Bytes mitgeteilt. Der nachfolgende Programmteil COMPAR vergleicht den gesendeten mit dem empfangenen Datenblock. Eine eventuelle Fehlermeldung wird über Tor 2 ausgegeben. Bei einer Nutzung des Programms zur Datenübertragung kann diese Fehlermeldung Aktivitäten in Form einer Wiederholung des fehlerhaften Datenblocks oder der Fortsetzung mit den nächsten anstehenden Daten auslösen.

Programm 8.2

```
Z8ASM      4.0
LOC     OBJ CODE     STMT SOURCE STATEMENT

                       1 SIO_EMR1 MODULE
                       2
                       3         ! Programm zur Kopplung von zwei EMR ueber SIO
                       4
                       5                 E M R 1 :  Messwerterfassung, Uebergabe an EMR 2
                       6                            Empfang der Quittungsmesswerte und
                       7                            Vergleich
                                         Sendeprotokoll : AA ! Byte0...Byte15 ! 1E
                       9
                      10                 Programm wird als Unterprogramm von Rahmen aufgerufen
                      11                                                                    !
                      12
                      13 GLOBAL
                      14         SEND PROCEDURE
                      15         ENTRY
P 0800                16 $ABS %800                              ! VARIANTE FUER U 883   !
                      17
P 0800 8D  08AC       18 INTTAB: JP      DUMMY
P 0803 8D  08AC       19         JP      DUMMY
P 0806 8D  08AC       20         JP      DUMMY
P 0809 8D  088E       21         JP      SIOIN                  ! IRQ3: Empfang der Quittungwerte!
P 080C 8D  087C       22         JP      SIOOUT                 ! IRQ4: Senden der Messdaten    !
P 080F 8D  08AC       23         JP      DUMMY
                      24
P 0812 8F             25 RAHMEN: DI                                                           -
P 0813 31  F0         26         SRP     #%F0                   ! Steuerregistergruppe          !
P 0815 FC  80         27         LD      R15,#%80               ! Stackpointer setzen           !
P 0817 8C  D7         28         LD      R8,#%D7                ! P01M: P04...P07 = A12...A15   !
                      29                                        ! Zeitsteuerung normal          !
                      30                                        ! P10...P17 = AD0...AD7         !
                      31                                        ! Stack intern                  !
                      32                                        ! P00...P03 = A8...A11          !
P 0819 7C  C1         33         LD      R7,#%C1                ! P3M: mit Paritaet             !
                      34                                        ! P30 = RxD   P37 = TxD         !
                      35                                        ! P31..P33 = Eingaenge          !
                      36                                        ! P34..P36 = Ausgaenge          !
                      37                                        ! P2 mit aktiven Pull-up's      !
P 081B 6C  00         38         LD      R6,#0                  ! P2M: alles Ausgaenge          !
                      39
P 081D D6  0828       40         CALL    LADEN                  ! HIER SIMULATION DER MESSWERTE !
P 0820 E6  02  00     41         LD      2,#0                   ! PORT 2 loeschen               !
P 0823 D6  083C       42         CALL    SENDEN                 ! Aufruf UNTERPROGRAMM SENDEN   !
P 0826 8B  FE         43 ENDE:   JR      ENDE
                      44
                      45
P 0828 70  FD         46 LADEN:  PUSH    RP                     ! Registerpointer laden         !
P 082A 31  10         47         SRP     #%10
P 082C 0C  30         48         LD      R0,#%30                ! Register 1. Messwert          !
P 082E 1C  40         49         LD      R1,#%40                ! 1. Messgroesse                !
P 0830 F3  01         50 LADE1:  LD      @R0,R1                 ! laden Register mit Messgroesse!
P 0832 0E             51         INC     R0
P 0833 1E             52         INC     R1                     ! neue Messgroesse              !
P 0834 A6  E0  40     53         CP      R0,#%40                ! letzter Messwert ?            !
P 0837 EB  F7         54         JR      NZ,LADE1               ! nein                          !
P 0839 50  FD         55         POP     RP
P 083B AF             56         RET
                      57
                      58
                      59         ! Unterprogramm Senden !
                      60         ! Input: 16 Messwerte in den Registern %30...%3F
                      61                  P2 = 00
                      62          Output: 16 Messwerte per SIO uebertragen
                      63                  P2 = %11 :  Uebertragung I.O.
                      64                  P2 = %33 :  Uebertragung fehlerhaft           !
                      65
P 083C E6  FB  18     66 SENDEN: LD      IMR,#%18               ! IRQ3(SIOIN) UND IRQ4  !
P 083F E6  FA  00     67         LD      IRQ,#0                 ! ALLE INTERRUPTS ZURUECK!
P 0842 E6  F9  2F     68         LD      IPR,#%2F               ! SIOIN > SIOOUT                !
P 0845 E6  F5  05     69         LD      PRE0,#%5               ! VORTEILER T0=1, AUTOMAT. NEUST.!
P 0848 E6  F4  08     70         LD      T0,#%8                 ! ZEITKONSTANTE T0              !
P 084B E6  1E  30     71         LD      %1E,#%30               ! SENDEZEIGER %1E LADEN         !
P 084E E6  1F  1F     72         LD      %1F,#%1F               ! EMPFANGSZEIGER LADEN          !
P 0851 E6  F0  AA     73         LD      SIO,#%AA               ! STARTBYTE AN SIO-SENDER       !
P 0854 E6  F1  03     74         LD      TMR,#%3                ! T0 LADEN UND STARTEN          !
P 0857 9F             75         EI
P 0858 A6  FB  80     76 WARTEN: CP      IMR,#%80               ! ALLE INTERRUPTS GESPERRT ?    !
P 085B EB  FB         77         JR      NZ,WARTEN              ! NEIN                          !
                      78         ! BEGINN AUSWERTUNG !
P 085D E6  1E  30     79 COMPAR: LD      %1E,#%30               ! MESSWERT-ZEIGER %1E           !
P 0860 E6  1F  20     80         LD      %1F,#%20               ! QUITTUNGSWERT-ZEIGER          !
P 0863 E5  1E  1D     81 COMPA1: LD      %1D,@%1E               ! LESEN MESSWERT                !
P 0866 A5  1F  1D     82         CP      %1D,@%1F               ! VERGLEICH MIT QUITTUNG        !
```

```
P 0869 EB  0D          JR      NZ,FEHLER                ! UNGLEICH                  !
P 086B 20  1E          INC     %1E                      ! ZEIGER +1                 !
P 086D 20  1F          INC     %1F
P 086F A6  1F  30      CP      %1F,#%30                 ! ALLE WERTE VERGLICHEN ?   !
P 0872 EB  EF          JR      NZ,COMPA1                ! NEIN                      !
P 0874 E6  02  11      LD      2,#%11                   ! AN P2 KENNUNG I.O. AUSGEBEN !
P 0877 AF              RET
P 0878 E6  02  33  FEHLER: LD  2,#%33                   ! FEHLERKENNUNG AN PORT2    !
P 087B AF              RET

                       ! INTERRUPTBEHANDLUNG SIO-SENDER                            !
P 087C A6  1E  40  SIOOUT: CP  %1E,#%40                 ! ALLE BYTES GESENDET ?     !
P 087F FB  06          JR      NC,SIOOU3                ! JA                        !
P 0881 E5  1E  F0      LD      SIO,@%1E                 ! NAECHSTES BYTE SENDEN     !
P 0884 20  1E          INC     %1E                      ! SENDEZEIGER + 1           !
P 0886 BF              IRET
P 0887 E6  F0  1E  SIOOU3: LD  SIO,#%1E                 ! ENDEBYTE SENDEN           !
P 088A 56  FB  08      AND     IMR,#%8                  ! NUR NOCH SIOIN-INTERRUPT  !
P 088D BF              IRET


                       ! INTERRUPTBEHANDLUNG SIO-EMPFAENGER                        !
P 088E A6  1F  1F  SIOIN:  CP  %1F,#%1F                 ! EMPFANGSZEIGER NOCH AM ANFANG?!
P 0891 EB  06          JR      NZ,SIOIN1                ! NEIN                      !
P 0893 A6  F0  AA      CP      SIO,#%AA                 ! TEST AUF STARTBYTE        !
P 0896 6B  09          JR      Z,SIOIN3                 ! STARTBYTE I.O.            !
P 0898 BF              IRET                             ! WEITER AUF STARTBYTE WARTEN !
P 0899 A6  1F  30  SIOIN1: CP  %1F,#%30                 ! ALLE BYTES EMPFANGEN ?    !
P 089C FB  06          JR      NC,SIOIN2                ! JA                        !
P 089E F5  F0  1F      LD      @%1F,SIO                 ! EMPFANGENES BYTE SPEICHERN !
P 08A1 20  1F      SIOIN3: INC %1F                      ! EMPFANGSZEIGER +1         !
P 08A3 BF          SIOIN4: IRET
P 08A4 A6  F0  1E  SIOIN2: CP  SIO,#%1E                 ! STOPPBYTE EMPFANGEN ?     !
P 08A7 EB  FA          JR      NZ,SIOIN4                ! NEIN                      !
P 08A9 E6  FB  00      LD      IMR,#0                   ! ALLE INTERRUPTS SPERREN   !
P 08AC BF          DUMMY:  IRET

P 08AD                 END SEND
                   END SIO_EMR1
```

Programmbeschreibung SIO-EMR2

Das Programm SIO-EMR2 (Programm 8.2) enthält alle notwendigen Aktivitäten für die Gegenstelle, also EMR 2. Dieser soll beispielsweise die empfangenen Meßwerte anzeigen (eventuell einer Auswertung unterziehen) und zur Quittung an EMR 1 zurücksenden.

Die Initialisierung der Steuerregister am Anfang des Programms legt in gewohnter Weise das Betriebsverhalten der Baugruppe fest. Im Gegensatz zum EMR 1 wird hier nur mit einer Interruptquelle SIOIN gearbeitet. Das entprechende Interruptprogramm übernimmt gleichzeitig das Quittungssenden der empfangenen Bytes. Bis zur Fertigmeldung für den gesamten Datenblock hält das Hauptprogramm am Symbol WARTEN an. Alle weiteren Aktivitäten werden innerhalb des Interruptprogramms SIOIN erledigt. Es prüft den Rahmen des Datenblocks, trägt das empfangene Byte in das durch den Empfangszeiger markierte Register ein und sendet das Quittungssignal zurück. Nach Empfang des Stoppbytes wird die Fertigmeldung (Registerinhalt %1E = %FF) an das Hauptprogramm abgesetzt, was dessen Weiterarbeit veranlaßt.

Im Beispielprogramm erfolgt diese Programmfortsetzung mit einer Ausgabe der empfangenen Bytes an Tor 2 (AUSGABE). Zwischen den einzelnen Ausgaben wird durch die Zeitschleife ZEIT eine Verzögerung erreicht, die eine visuelle Lesbarkeit der Bitmuster ermöglicht.

Bei nicht zustandekommender Datenübertragung (z. B. durch fehlende Verbindungsleitung oder falsche Taktrate) verharren die vorgestellten Programme in einer Endlosschleife (bei Marke «WARTEN», SIO-EMR1 in Zeile 76, SIO-EMR2 in Zeile 47). Das kann umgangen werden, wenn an diesen Stellen über Zählschleifen eine Zeitbegrenzung eingebaut wird.

```
Z8ASM      4.0
LOC    OBJ CODE    STMT SOURCE STATEMENT

                     1 SIO_EMR2 MODULE
                     2
                     3         ! Programm zur Kopplung von zwei EMR ueber SIO
                     4
                     5                 E M R 2 : Messwertanzeige und Ruecksendung
                     6                           der Daten an EMR 1
                     7
                     8                 Protokoll: Startbyte %AA
                     9                            Byte0...Byte15
                    10                            Stoppbyte %1E
                    11
                    12 GLOBAL
                    13         EMPF PROCEDURE
                    14         ENTRY
P 0800              15 $ABS %800                        ! VARIANTE FUER U 883   !
                    16
P 0800 8D  087E     17 INTTAB: JP      DUMMY
P 0803 8D  087E     18         JP      DUMMY
P 0806 8D  087E     19         JP      DUMMY
P 0809 8D  085D     20         JP      SIOIN            ! IRQ3: Empfang der Messwerte          !
P 080C 8D  087E     21         JP      DUMMY
P 080F 8D  087E     22         JP      DUMMY
P 0812 8F           23 RAHMEN: DI
P 0813 31  F0       24         SRP     #%F0             ! Steuerregistersatz                   !
P 0815 FC  80       25         LD      R15,#%80         ! Stackpointer setzen                  !
P 0817 8C  D7       26         LD      R8,#%D7          ! P01M: P04...P07 = A12...A15          !
                    27                                  ! Zeitsteuerung normal                 !
                    28                                  ! P10...P17 = AD0...AD7                !
                    29                                  ! Stack intern                         !
                    30                                  ! P00...P03 = A8...A11                 !
P 0819 7C  C1       31         LD      R7,#%C1          ! P3M: mit Paritaet                    !
                    32                                  ! P30 = RxD      P37 = TxD             !
                    33                                  ! P31...P33 = Eingaenge                !
                    34                                  ! P34...P36 = Ausgaenge                !
                    35                                  ! P2 mit aktiven Pull-up's             !
P 081B 6C  00       36         LD      R6,#0            ! P2M: alles Ausgaenge                 !
P 081D E6  02  00   37         LD      2,#0             ! Port 2 loeschen                      !
P 0820 BC  08       38         LD      R11,#8           ! IMR: nur IRQ 3 zugelassen            !
P 0822 AC  00       39         LD      R10,#0           ! IRQ: alle Interrupts loeschen        !
P 0824 9C  2F       40         LD      R9,#%2F          ! SIOIN > SIOOUT                       !
P 0826 5C  05       41         LD      R5,#5            ! PRE0: Vorteiler T0=1, automat. Neustart!
P 0828 4C  08       42         LD      R4,#8            ! T0: Zeitkonstante T0                 !
P 082A E6  1F  1F   43         LD      %1F,#%1F         ! Empfangszeiger laden                 !
P 082D E6  1E  00   44         LD      %1E,#0           ! Fertigmeldung ruecksetzen            !
P 0830 1C  03       45         LD      R1,#3            ! T0 laden und starten                 !
P 0832 9F           46         EI
P 0833 A6  1E  00   47 WARTEN: CP      %1E,#0           ! Datenblock empfangen ?               !
P 0836 6B  FB       48         JR      Z,WARTEN         ! nein                                 !
                    49
P 0838 E6  1F  20   50 AUSGAB: LD      %1F,#%20         ! Ausgabezeiger auf erstes Byte setzen !
P 083B E5  1F  02   51 AUSGA1: LD      2,@%1F           ! Byte an Port 2 ausgeben              !
P 083E 20  1F       52         INC     %1F
P 0840 D8  084A     53         CALL    ZEIT             ! Warteschleife                        !
P 0843 A6  1F  30   54         CP      %1F,#%30         ! alle Bytes ausgegeben ?              !
P 0846 7B  F3       55         JR      C,AUSGA1         ! nein                                 !
P 0848 8B  C8       56         JR      RAHMEN           ! fertig fuer neuen Empfang            !
                    57
                    58
                    59         ! Unterprogramm Zeitschleife    !
                    60
P 084A 70  FD       61 ZEIT:   PUSH    RP
P 084C 31  10       62         SRP     #%10             ! Zeitschleife auf Registergruppe 10   !
P 084E 0C  00       63         LD      R0,#0
P 0850 1C  00       64         LD      R1,#0
P 0852 2C  02       65         LD      R2,#2
P 0854 0A  FE       66 ZEIT1:  DJNZ    R0,ZEIT1
P 0856 1A  FC       67         DJNZ    R1,ZEIT1
P 0858 2A  FA       68         DJNZ    R2,ZEIT1
P 085A 50  FD       69         POP     RP
P 085C AF           70         RET
                    71
                    72         ! Interruptbehendlung SIO-Empfaenger    !
                    73
P 085D A6  1F  1F   74 SIOIN:  CP      %1F,#%1F         ! Empfangszeiger noch auf Anfang ?     !
P 0860 EB  0F       75         JR      NZ,SIOIN1        ! nein                                 !
P 0862 E4  F0  1D   76         LD      %1D,SIO          ! empfangenes Byte nach %1D            !
P 0865 A6  1D  AA   77         CP      %1D,#%AA         ! Startbyte %AA ?                      !
P 0868 6B  01       78         JR      Z,SIOIN2         ! ja                                   !
P 086A BF           79         IRET                     ! weiter auf Startbyte warten          !
P 086B E4  1D  F0   80 SIOIN2: LD      SIO,%1D          ! empfangenes Byte zuruecksenden       !
P 086E 20  1F       81         INC     %1F              ! Empfangszeiger + 1                   !
P 0870 BF           82         IRET
```

```
P 0871 A6  1F  30     83 SIOIN1: CP      %1F,#%30        ! alle Bytes empfangen ?                !
P 0874 FB  09         84         JR      NC,SIOIN4       ! ja                                    !
P 0876 F5  F0  1F     85         LD      @%1F,SIO        ! empfangenes Byte abspeichern          !
P 0879 E5  1F  F0     86         LD      SIO,@%1F        ! und zuruecksenden                     !
P 087C 20  1F         87         INC     %1F
P 087E BF             88 DUMMY:  IRET
P 087F E4  F0  1D     89 SIOIN4: LD      %1D,SIO         ! Stoppbyte                             !
P 0882 E4  1D  F0     90         LD      SIO,%1D         ! zuruecksenden                         !
P 0885 A6  1D  1E     91         CP      %1D,#%1E        ! Stoppbyte I.O. ?                      !
P 0888 EB  06         92         JR      NZ,SIOIN5       ! nein                                  !
P 088A E6  FB  00     93         LD      IMR,#%0         ! kein Empfang mehr                     !
P 088D E6  1E  FF     94         LD      %1E,#%FF        ! Fertigmeldung an Rahmenprogramm       !
P 0890 BF             95 SIOIN5: IRET
                      96
P 0891                97         END EMPF
                      98 END SIO_EMR2
                      99
```

9. Ausgewählte Programm- und Anwendungsbeispiele

Unabhängig von der konkreten Anwendung eines EMR und der Komplexität von Hard- und Software werden bestimmte Unterprogramme immer wieder zur Lösung von Teilaufgaben benötigt. Dazu gehören vor allem:

- Arithmetikprogramme, z. B. Multiplikation, Division, Maximal- und Mittelwertberechnung,
- Konvertierungsprogramme, z.B. Hex → Dezimal, Dezimal → Hex, BCD → Hex,
- Messung von Periodendauer und Frequenz einer Impulsfolge,
- Erzeugung von Impulsen definierter Dauer,
- Ausgabeprogramme für Anzeigebaugruppen,
- Analog-Digital-Umsetzung,
- Inbetriebnahme- bzw. Testprogramme.

Die folgenden Programmbeispiele zu den genannten Komplexen sind so gestaltet, daß eine möglichst universelle Verwendbarkeit gegeben ist. Hinsichtlich Programmlaufzeit und Speicherplatzbedarf lassen sich in einigen Fällen noch effektivere Lösungen finden, die aber meist unübersichtlicher und daher schwerer verständlich sind. In allen Programmen wurden die benutzten Register in absoluter Form angegeben. Die an sich universellere, symbolische Schreibweise fand keine Verwendung, da sie bei der Einarbeitung in die Materie zunächst schwerer verständlich ist. Liegen ausreichende Erfahrungen bei der Programmentwicklung vor, ist diese Darstellungsform allerdings vorzuziehen. Mit einem sogenannten Vereinbarungsteil am Anfang des Programms ist die Zuordnung zwischen Symbolen und Absolutregistern gegeben. Eine solche Herangehensweise ist natürlich nur möglich, wenn ein Editor-/Assemblerprogramm zur Verfügung steht.

9.1. Arithmetikprogramme

Multiplikation

Das Programm MUL (Programm 9.1) gestattet die vorzeichenbehaftete Multiplikation von 2 16-Bit-Zahlen, wobei das höchstwertige Bit als Vorzeichen bewertet wird (gesetztes Bit 15 entspricht negativem Vorzeichen). Die restlichen Bit bilden den Betrag. Als Ergebnis entsteht eine 32-Bit-Zahl, wobei das höchstwertige Bit wiederum das Vorzeichen stellt.

Division

Mit dem Programm DIV (Programm 9.2) kann eine vorzeichenlose 32-Bit-Zahl durch eine vorzeichenlose 16-Bit-Zahl dividiert werden. Das Ergebnis ist ebenfalls eine 16-Bit-Zahl.

Die gemeinsame Verwendung der Programme MUL und DIV ist oftmals notwendig, wenn eine größere Rechengenauigkeit gefordert ist. Beispielsweise liefert das Pro-

Programm 9.1

```
Z8ASM      4.0
LOC    OBJ CODE     STMT SOURCE STATEMENT

                      1 MUL MODULE
                      2
                      3 ! multipliziert 2 vorzeichenbehaftete 16-Bit-Zahlen (Bit 15 ist
                      4   Vorzeichenbit, bei negativen Zahlen gesetzt ).
                      5   Das Ergebnis ist eine 32-Bit-Zahl (31 Bit Betrag, hochwertigstes
                      6   Bit liefert Vorzeichen )
                      7
                      8   Multiplikant: R0,R1
                      9   Multiplikator: R2,R3
                     10   Produkt:       R4,R5,R6,R7
                     11
                     12
                     13 GLOBAL
                     14
P 0000               15 MULTIPLY PROCEDURE
                     16
                     17 ENTRY
                     18
P 0000 70  FD        19          PUSH    RP              ! Registerpointer retten        !
P 0002 31  20        20          SRP     #%20            ! Registergruppe einstellen     !
P 0004 8C  11        21          LD      R8,#%11         ! Schleifenzaehler              !
P 0006 98  E2        22          LD      R9,R2
P 0008 B2  90        23          XOR     R9,R0
P 000A 56  E9  80    24          AND     R9,#%80         ! Vorzeichenbit ausblenden      !
P 000D B0  E4        25          CLR     R4
P 000F B0  E5        26          CLR     R5
P 0011 56  E0  7F    27          AND     R0,#%7F         ! Vorzeichenbit ausblenden      !
P 0014 56  E2  7F    28          AND     R2,#%7F
P 0017 CF            29 MU1:     RCF                     ! Cy = 0                        !
P 0018 C0  E4        30          RRC     R4
P 001A C0  E5        31          RRC     R5
P 001C C0  E2        32          RRC     R2
P 001E C0  E3        33          RRC     R3
P 0020 FB  05        34          JR      NC,MU2          ! Sprung, wenn niederwert. Bit=0!
P 0022 CF            35          RCF
P 0023 12  51        36          ADC     R5,R1           ! aufaddieren Ergebnisregister  !
P 0025 12  40        37          ADC     R4,R0
P 0027 8A  EE        38 MU2:     DJNZ    R8,MU1          ! Schleifenzahl abarbeiten      !
P 0029 42  49        39          OR      R4,R9
P 002B 68  E2        40          LD      R6,R2           ! Ergebnis auf Zielregister     !
P 002D 78  E3        41          LD      R7,R3
P 002F 50  FD        42          POP     RP              ! Registerpointer herstellen    !
P 0031 AF            43          RET
                     44
P 0032               45 END MULTIPLY
                     46
                     47 END MUL
```

gramm DIV bei der Aufgabenstellung Y = 100/99 als Ergebnis den Wert 1. Eine wesentlich größere Genauigkeit entsteht, wenn der Zähler des Bruches vorher mit einer Zehnerpotenz multipliziert wird. Die Aufgabe lautet dann beispielsweise: Y = 100 × 1000/99. Das (gedachte) Komma im Ergebnis muß um 3 Stellen nach links verschoben werden, d. h., die Berechnung erfolgt mit einer Genauigkeit von 3 Stellen nach dem Komma. Bei der Wahl des Multiplikationsfaktors ist zu beachten, daß kein Zahlenüberlauf im Ergebnis der Multiplikation auftritt. Da für die Betragsdarstellung des Ergebnisses 31 Bit zur Verfügung stehen, kann dieser Fall in der Praxis meist ausgeschlossen werden.

Natürlich ersetzen die vorgestellten Programme keine vollständige Fest- oder Gleitkommaarithmetik. Für typische EMR-Aufgabenstellungen reichen die damit möglichen Rechenoperationen jedoch aus.

Mittelwertbildung

Mit dem Programm MITTEL (Programm 9.3) kann der Mittelwert von vorzeichenlosen 16-Bit-Zahlen berechnet werden. Die Werte müssen im externen Speicher (RAM) ab einer definierten Anfangsadresse aufeinanderfolgend stehen.

Programm 9.2

```
Z8ASM      4.0
LOC    OBJ CODE    STMT SOURCE STATEMENT

                     1 DIV MODULE
                     2
                     3 ! Division einer vorzeichenlosen 32-Bit-Zahl durch eine 16-Bit-Zahl
                     4   das Ergebnis ist wiederum eine 16-Bit-Zahl
                     5
                     6     A : B = C
                     7
                     8     A = R0,R1,R2,R3
                     9
                    10     B = R4,R5
                    11
                    12     C = R6,R7                                              !
                    13
                    14 GLOBAL
                    15
P 0000              16 MATH PROCEDURE
                    17
                    18 ENTRY
                    19
P 0000 31  20       20 BEGIN:  SRP     #%20
P 0002 8C  11       21         LD      R8,#%11          ! Schleifenzahl           !
P 0004 22  15       22 DIV1:   SUB     R1,R5
P 0006 32  04       23         SBC     R0,R4            ! A - B                   !
P 0008 FB  04       24         JR      NC,DIV2          ! Sprung, wenn A>B        !
P 000A 02  15       25         ADD     R1,R5
P 000C 12  04       26         ADC     R0,R4            ! letzte Subtraktion korrigieren !
P 000E EF           27 DIV2:   CCF                      ! 0 rechts einschieben    !
P 000F 10  E7       28 DIV3:   RLC     R7
P 0011 10  E6       29         RLC     R6
P 0013 90  E3       30         RL      R3
P 0015 10  E2       31         RLC     R2
P 0017 10  E1       32         RLC     R1
P 0019 10  E0       33         RLC     R0
P 001B 7B  03       34         JR      C,DIV4           ! Sprung, wenn hochwertigee Bit = 1 !
P 001D 8A  E5       35         DJNZ    R8,DIV1          ! Schleifenzahl abarbeiten !
P 001F AF           36         RET
P 0020 8A  01       37 DIV4:   DJNZ    R8,DIV5
P 0022 AF           38         RET
P 0023 22  15       39 DIV5:   SUB     R1,R5
P 0025 32  04       40         SBC     R0,R4            ! A - B                   !
P 0027 8B  E6       41         JR      DIV3
                    42
P 0029              43 END MATH
                    44
                    45 END DIV
```

Das Programm läßt sich ohne weiteres so modifizieren, daß die zur Mittelwertbildung verwendeten Zahlen nicht im externen Speicher, sondern im Universalregistersatz stehen. Dazu sind vor allem die LDE-Befehle durch andere, geeignete Ladebefehle zu ersetzen.

Maximalwertberechnung

Das Programm MAX (Programm 9.4) berechnet den Spitzenwert von vorzeichenlosen 16-Bit-Zahlen. Die Werte müssen im externen Speicher (RAM) ab einer definierten Anfangsadresse aufeinanderfolgend stehen. Für die Berechnung des Maximalwerts von Zahlen, die im Universalregistersatz abgelegt sind, müssen wiederum die auf den externen Speicher bezogenen Befehle modifiziert werden.

Programm 9.3

```
Z8ASM      4.0
LOC    OBJ CODE    STMT SOURCE STATEMENT

                     1 MITTEL MODULE
                     2
                     3 EXTERNAL
                     4
                     5 DIV PROCEDURE
                     6
                     7  ! Berechnet den arithmetischen Mittelwert von 16-Bit-Zahlen    !
                     8
                     9 ! R11 = Zahl der zu mittelnden Werte = n                        !
                    10 ! RR8 = Anfangsadresse im RAM                                   !
                    11 ! RR6 = Ergebnis                                                !
                    12
                    13 GLOBAL
                    14
P 0000              15 MESS PROCEDURE
                    16
                    17 ENTRY
                    18
P 0000 31  20       19          SRP     #%20            ! Registergruppe einstellen    !
P 0002 A8  EB       20          LD      R10,R11         ! Zahl d. Werte uebernehmen    !
P 0004 B0  E0       21          CLR     R0
P 0006 B0  E1       22          CLR     R1
P 0008 B0  E2       23          CLR     R2
P 000A B0  E3       24          CLR     R3              ! def. Anfangszustand          !
P 000C C2  58       25 MIT1:    LDC     R5,@RR8
P 000E A0  E8       26          INCW    RR8
P 0010 C2  68       27          LDC     R6,@RR8         ! 16-Bit Messwert holen        !
P 0012 A0  E8       28          INCW    RR8             ! naechster Wert im RAM        !
P 0014 02  36       29          ADD     R3,R6
P 0016 12  25       30          ADC     R2,R5
P 0018 16  E1  00   31          ADC     R1,#0
P 001B 16  E0  00   32          ADC     R0,#0           ! 16-Bit-Werte aufaddieren     !
P 001E AA  EC       33          DJNZ    R10,MIT1
P 0020 4C  00       34          LD      R4,#0           ! H-Teil Divisor = 0           !
P 0022 58  EB       35          LD      R5,R11          ! L-Teil Divisor = n           !
P 0024 D6  0000*    36          CALL    DIV             ! Ergebnis berechnen           !
P 0027 AF           37          RET
                    38
P 0028              39 END MESS
                    40
                    41 END MITTEL
```

Programm 9.4

```
Z8ASM      4.0
LOC    OBJ CODE    STMT SOURCE STATEMENT

                     1 MAX MODULE
                     2
                     3 ! Das Programm MAX sucht den Maximalwert von n 16-Bit-Zahlen , die in einem
                     4 externen Speicher ab einer definierten Anfangsadresse aufeinanderfolgend
                     5 stehen.
                     6 Anfangsadresse: RR0
                     7 Anzahl der zu mittelnden Werte: RR14
                     8 Maximalwert MAX: RR8
                     9 Nummer des Wertes MAX: RR10 !
                    10
                    11 GLOBAL
                    12
P 0000              13 MAIN PROCEDURE
                    14
                    15 ENTRY
P 0000 31  20       16 BEGIN:   SRP     #%20
P 0002 CF           17          RCF
P 0003 10  EF       18          RLC     R15
P 0005 10  EE       19          RLC     R14             ! Verdoppeln von n !
P 0007 B0  EA       20          CLR     R10
P 0009 B0  EB       21          CLR     R11             ! Messpunktzaehler RESET !
P 000B D8  EB       22 MA1:     LD      R13,R11         ! aktuellen Maximalwert merken !
P 000D C8  EA       23          LD      R12,R10
P 000F CF           24          RCF
P 0010 C0  EC       25          RRC     R12             ! Messpunktzaehler :2, fuer Ausgabe !
P 0012 C0  ED       26          RRC     R13             ! R12,R13 :Nr.des Maximalwertes !
P 0014 82  20       27          LDE     R2,@RR0         ! H-Teil 1.Wert !
P 0016 A0  E0       28          INCW    RR0             ! naechste Adresse !
P 0018 A0  EA       29          INCW    RR10            ! MAX-Positionszaehler +1 !
P 001A 82  30       30          LDE     R3,@RR0         ! L-Teil,1.WERT !
P 001C A0  E0       31 MA2:     INCW    RR0             ! Adresse +1 !
P 001E A0  EA       32          INCW    RR10            ! MAX-Position +1 !
P 0020 82  40       33          LDE     R4,@RR0         ! H-Teil, 2.Wert   !
P 0022 A0  E0       34          INCW    RR0
```

```
P 0024 A0  EA          35          INCW    RR10            ! Adresse und Position +1 !
P 0026 82  50          36          LDE     R5,@RR0         ! L-Teil, 2.Wert !
P 0028 A2  EA          37          CP      R14,R10         ! Endwert erreicht?/H-Teil !
P 002A EB  04          38          JR      NZ,MA4          ! wenn nicht, Sprung !
P 002C A2  FB          39          CP      R15,R11         ! Endwert erreicht?/L-Teil? !
P 002E 7B  14          40          JR      C,MA5           ! Ende erreicht !
P 0030 22  53          41 MA4:     SUB     R5,R3           ! neuer wert groesser? !
P 0032 32  42          42          SBC     R4,R2
P 0034 70  FC          43          PUSH    FLAGS           ! Ergebnis merken !
P 0036 80  EA          44          DECW    RR10
P 0038 80  E0          45          DECW    RR0             ! Adresse und Position  korrigieren !
P 003A 50  FC          46          POP     FLAGS           ! Ergebnis wiederholen !
P 003C FB  CD          47          JR      NC,MA1          ! neuer wert groesser,wird neues MAX !
P 003E A0  E0          48          INCW    RR0
P 0040 A0  EA          49          INCW    RR10            ! neuer Wert kleiner/gleich,MAX bleibt !
P 0042 8B  D8          50          JR      MA2
P 0044 88  E2          51 MA5:     LD      R8,R2           ! MAX auf Zielregister laden !
P 0046 98  E3          52          LD      R9,R3
P 0048 AF              53          RET
P 0049                 54 END MAIN
                       55 END MAX
```

9.2. *Konvertierungsprogramme*

Wandlung einer 16-Bit-Hexzahl in Dezimalform

Mit dem Programm HEXDEZ (Programm 9.5) wird eine vorzeichenlose 16-Bit-Hexzahl dezimal gewandelt. Jede der 5 ausgegebenen Dezimalstellen steht in einem Register, wobei die höherwertige Tetrade des Registerinhalts stets Null ist. Durch ein nachträg-

Programm 9.5

```
Z8ASM      4.0
LOC    OBJ CODE    STMT SOURCE STATEMENT

                     1 HEXDEZ MODULE
                     2
                     3 ! Convertiert eine 16-Bit-Hexzahl in das ASCII-Format, wobei
                     4 jede Stelle in einem Universalregister abgelegt wird !
                     5
                     6 ! Input :    Zahl in Registern %20,%21
                     7   Output:    Zahl ab Register %22                    !
                     8 GLOBAL
                     9
P 0000               10 CONVERT1 PROCEDURE
                    11
                    12 ENTRY
                    13
P 0000 70  FD       14 BEGIN:   PUSH    RP              ! Registerpointer retten          !
P 0002 31  20       15          SRP     #%20            ! Registergruppe einstellen       !
P 0004 AC  22       16          LD      R10,#%22        ! Zeiger auf Zielregister         !
P 0006 7C  27       17          LD      R7,#%27         ! Teilungsfaktor 10000            !
P 0008 8C  10       18          LD      R8,#%10
P 000A D6  0029'    19          CALL    STELLE
P 000D 7C  03       20          LD      R7,#3           ! Teilungsfaktor 1000             !
P 000F 8C  E8       21          LD      R8,#%E8
P 0011 D6  0029'    22          CALL    STELLE
P 0014 7C  00       23          LD      R7,#0           ! Teilungsfaktor 100              !
P 0016 8C  64       24          LD      R8,#%64
P 0018 D6  0029'    25          CALL    STELLE
P 001B 8C  0A       26          LD      R8,#%A          ! Teilungsfaktor 10               !
P 001D D6  0029'    27          CALL    STELLE
P 0020 9C  30       28          LD      R9,#%30         ! ASCII vorbereiten               !
P 0022 02  91       29          ADD     R9,R1           ! Einerstelle bilden              !
P 0024 F3  A9       30          LD      @R10,R9         ! auf Zielregister                !
P 0026 50  FD       31          POP     RP              ! Registerpointer herstellen      !
P 0028 AF           32          RET
P 0029 9C  30       33 STELLE:  LD      R9,#%30         ! ASCII-Zeichen vorbereiten       !
P 002B CF           34          RCF
P 002C 22  18       35 ST1:     SUB     R1,R8           ! Divisionsschleife               !
P 002E 32  07       36          SBC     R0,R7
P 0030 9E           37          INC     R9              ! Ergebnisregister +1             !
P 0031 FB  F9       38          JR      NC,ST1
P 0033 02  18       39          ADD     R1,R8           ! letzte Subtraktion anulliert    !
P 0035 12  07       40          ADC     R0,R7
P 0037 00  E9       41          DEC     R9              ! Ergebnisregister korrigieren    !
P 0039 F3  A9       42          LD      @R10,R9         ! Ergebnis auf Zielregister       !
P 003B AE           43          INC     R10             ! naechste Stelle                 !
P 003C AF           44          RET
                    45
P 003D              46  END CONVERT1
                    47
                    48  END HEXDEZ
```

liches Setzen der Bits D4 und D5 in den Ergebnisregistern (z. B. durch den Befehl OR nn,%30) steht das Ergebnis in Form von ASCII-Zeichen zur Verfügung.

Wandlung einer 4stelligen BCD-Zahl in eine 16-Bit-Hexzahl
Das Programm BCDHEX (Programm 9.6) wandelt eine gepackte, 4stellige BCD-Zahl in eine 16-Bit-Hexzahl. Eine Erweiterung der Stellenzahl auf 5 ist möglich, allerdings muß dann der eventuelle Zahlenüberlauf im Ergebnis beachtet werden, d. h., der gewandelte Wert darf nicht größer als %FFFF sein.

9.3. Anzeigeprogramme für Ziffernanzeigebaugruppen

9.3.1. Direkter Anschluß von 7-Segment-Dekodern

Die einfachste Form der Ausgabe eines numerischen oder hexadezimalen Werts besteht darin, an jeweils 4 Leitungen eines Ausgabetores einen BCD-zu-7-Segmentdeko-

Programm 9.6

```
Z8ASM     4.0
LOC     OBJ CODE     STMT SOURCE STATEMENT

                        1 BCDHEX MODULE
                        2
                        3 EXTERNAL
                        4 MUL PROCEDURE
                        5
                        6 ! Wandelt gepackte BCD-Zahl in 16-Bit-Hexzahl                !
                        7 ! RR14 = BCD- Zahl                                           !
                        8 ! RR10 = Hexzahl                                             !
                        9
                       10 GLOBAL
                       11
P 0000                 12 CONVERT PROCEDURE
                       13
                       14 ENTRY
                       15
                       16
P 0000 31   20         17          SRP     #%20          ! Registergruppe entspr. MUL   !
P 0002 B0   E0         18          CLR     R0
P 0004 B0   E2         19          CLR     R2            ! H-Teil Multiplikant = 0      !
P 0006 B8   EF         20          LD      R11,R15
P 0008 56   EB   0F    21          AND     R11,#%0F      ! Einer in R11                 !
P 000B A8   EF         22          LD      R10,R15
P 000D F0   EA         23          SWAP    R10
P 000F 56   EA   0F    24          AND     R10,#%0F      ! Zehner in R10                !
P 0012 C8   EE         25          LD      R12,R14
P 0014 56   EC   0F    26          AND     R12,#%0F      ! Hunderter in R12             !
P 0017 D8   EE         27          LD      R13,R14
P 0019 F0   ED         28          SWAP    R13
P 001B 56   ED   0F    29          AND     R13,#%0F      ! Tausender in R13             !
P 001E 18   EA         30          LD      R1,R10
P 0020 3C   0A         31          LD      R3,#10        ! Multiplikation mit 10        !
P 0022 D6   0000*      32          CALL    MUL
P 0025 02   B7         33          ADD     R11,R7        ! Ergebnis aufaddieren         !
P 0027 18   EC         34          LD      R1,R12
P 0029 3C   64         35          LD      R3,#100       ! Multiplikation mit 100       !
P 002B D6   0000*      36          CALL    MUL
P 002E B0   EA         37          CLR     R10           ! H-Teil Ergebnis zunaechst 0  !
P 0030 02   B7         38          ADD     R11,R7
P 0032 12   A6         39          ADC     R10,R6        ! Ergebnis aufaddieren         !
P 0034 18   ED         40          LD      R1,R13
P 0036 3C   E8         41          LD      R3,#%E8       ! Multiplikation mit 1000      !
P 0038 2C   03         42          LD      R2,#3
P 003A D6   0000*      43          CALL    MUL
P 003D 02   B7         44          ADD     R11,R7
P 003F 12   A6         45          ADC     R10,R6        ! Ergebnis aufaddieren         !
                       46
P 0041 AF              47          RET
                       48
P 0042                 49 END CONVERT
                       50
                       51 END BCDHEX
```

der anzuschließen. In den meisten Fällen dürfte diese Methode aber nicht anwendbar sein, weil bei mehrstelligen Anzeigen zu viele E/A-Leitungen notwendig werden, d.h. anderen Anwendungen verlorengehen. Diese Form der Anzeige ist akzeptabel, wenn entweder besonders wenige Stellen anzuzeigen sind oder Ein-/Ausgabeleitungen für weitere Funktionen nicht benötigt werden. Grundsätzlich kann diese Variante nur in Verbindung mit der 64poligen Schaltkreisversion eine gewisse Bedeutung erlangen, wenn die Minimierung des Bauelementeaufwands bzw. des Platzbedarfs im Vordergrund steht.

9.3.2. Anzeige mit Dekoder *V40511*

Mit dem Dekoderschaltkreis *V40511* steht ein Bauelement mit zusätzlicher Speicherfunktion zur Verfügung. Eine damit aufgebaute Ziffernanzeige ist in bezug auf die zeitliche Belastung des Rechners besonders vorteilhaft. Die Anzeigebaugruppe muß immer nur dann bedient werden, wenn neue Werte zur Ausgabe anstehen. Damit sind die für die Ansteuerung der Anzeige eingesetzten Torleitungen prinzipiell auch für andere Zwecke verwendbar. Bild 9.1 zeigt eine entsprechende Schaltung zur Ansteuerung von maximal 8 Anzeigestellen. Die BCD-Dateneingänge aller Dekoder sind paral-

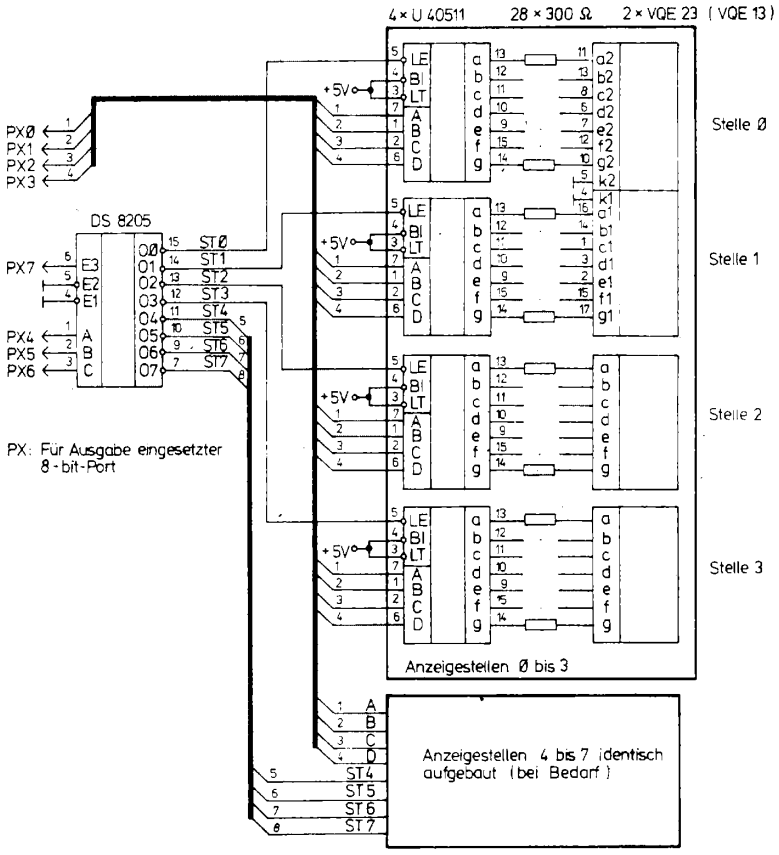


Bild 9.1
8stellige Ziffernanzeige mit V40511

Programm 9.7

```
Z8ASM      4.0
LOC    OBJ CODE     STMT SOURCE STATEMENT

                      1 AUSGABE1 MODULE
                      2
                      3 ! Realisiert die Ansteuerung einer Anzeigeeinheit mit dem
                      4   Dekoderschaltkreis V 40511 .
                      5   Es koennen maximal 8 Stellen bedient werden.
                      6   Unabhaengig von der Stellenzahl wird ein Port des EMR fuer
                      7   die Ausgabe benutzt.
                      8
                      9 GLOBAL
                     10
                     11 OUTPUT PROCEDURE
                     12
                     13 ENTRY
                     14
P 0812               15 $ABS %812
                     16
P 0812 31  10        17          SRP     #%10
P 0814 9C  10        18          LD      R9,#%10          ! Zeiger auf Anzeigepuffer      !
P 0816 AC  08        19          LD      R10,#%8          ! Angabe der Stellenzahl        !
P 0818 E3  B9        20 MA1:     LD      R11,@R9          ! Daten auf Pufferregister      !
P 081A B9  02        21          LD      2,R11            ! Ausgabe                       !
P 081C 56  02  0F    22          AND     2,#%0F           ! obere 4 Bit ausblenden        !
P 081F F0  E9        23          SWAP    R9               ! Stellenzahl vorbereiten       !
P 0821 56  E9  F0    24          AND     R9,#%F0          ! untere 4 Bit ausblenden       !
P 0824 44  E9  02    25          OR      2,R9             ! Anzeigestelle einblenden      !
P 0827 46  E9  01    26          OR      R9,#1            ! Zeiger wiederherstellen       !
P 082A F0  E9        27          SWAP    R9               ! Stellenzahl in L-Tetrade      !
P 082C 46  02  80    28          OR      2,#%80           ! Strobeimpuls ein              !
P 082F FF            29          NOP                      ! Warten                        !
P 0830 56  02  7F    30          AND     2,#%7F           ! Strobeimpuls aus              !
P 0833 9E            31          INC     R9               ! Naechste Stelle               !
P 0834 AA  E2        32          DJNZ    R10,MA1          ! Stellenzahl abarbeiten        !
P 0836 AF            33          RET
                     34
P 0837               35 END OUTPUT
                     36
                     37 END AUSGABE1
                     38
```

lel geschaltet. Über die Leitungen PX0 bis PX3 wird die Dateninformation ausgegeben. Ein zusätzlicher 1-aus-8-Dekoder *DS 8205* adressiert die aktuelle Anzeigestelle, indem der betreffende Ausgang mit dem Anschluß /LE (latch enable) des zugeordneten *V 40511* verbunden ist. Die Bereitstellung der Stellenadresse (0 bis 7) erfolgt über PX4 bis PX6. Eine Aktivierung des *DS 8205* und damit ein Einschreiben von Daten in den Zwischenspeicher der adressierten Steile ist nur möglich, wenn Bit D7 H-Pegel führt, sonst bleiben alle Ausgänge des *DS 8205* inaktiv.

Das zugehörige Programm AUSGABE1 (Programm 9.7) stellt zunächst die Dateninformation sowie die Adresse der Anzeigestelle bereit. Dann wird auf der Torleitung D7 ein kurzer Einschreibimpuls erzeugt. Mit der Rückflanke dieses Impulses stehen die Daten für die gerade adressierte Stelle im Zwischenspeicher zur Verfügung. Das Beispielprogramm AUSGABE1 realisiert die Ausgabe des Anzeigepuffers, bestehend aus 8 aufeinanderfolgenden Universalregistern über Tor 2 des EMR. Es ist zu beachten, daß das hochwertige Nibbel (Tetrade) in jedem Byte des Anzeigepuffers mit 0 belegt sein muß, während die Information selbst im niederwertigen Nibbel steht.

9.3.3. Zeitmultiplexe Anzeige

Der relativ hohe Bauelementeaufwand für eine mehrstellige Ziffernanzeige kann stark reduziert werden, wenn die Ausgabe zeitmultiplex erfolgt. Die Zahl der Anzeigestellen

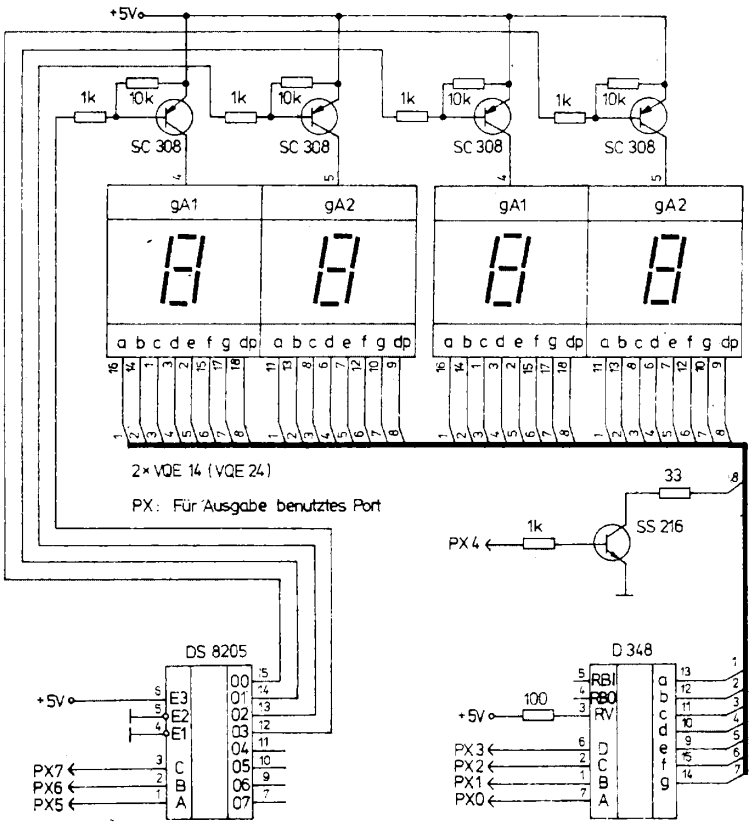


Bild 9.2
4stellige, zeitmultiplex arbeitende Ziffernanzeige-Baugruppe

ist in der Praxis durch die maximale Strombelastbarkeit der Segmentausgänge des eingesetzten Dekoders begrenzt. Naturgemäß ergeben sich günstigere Verhältnisse, wenn Anzeigebauelemente hoher Lichtstärkegruppen zur Verfügung stehen, da der benötigte Strom je Segment entsprechend niedrig gewählt werden kann. Bei Lichtschachtanzeigen *VQE24E* in Verbindung mit Dekodern *D347/348* ist diese Grenze bei 8 Stellen erreicht. In der Praxis werden meist weniger Anzeigestellen benötigt. Die Schaltung einer 4stelligen, zeitmultiplex arbeitenden Anzeigebaugruppe ist in Bild 9.2 dargestellt.

Die Ansteuerung erfolgt wiederum über ein Tor des EMR. Das Ausgabeprogramm AUSGABE2 (Programm 9.8) stellt den Inhalt eines Anzeigepuffers auf den entsprechenden Stellen der Anzeige dar. Die Anzeige wird in einem Interrupt-Behandlungsprogramm aktualisiert, wobei als Interruptquelle der Nulldurchgang des Zeitgebers T0 definiert ist.

Die zeitmultiplexe Ziffernausgabe ist zweifellos eine effektive Methode bezüglich des Bauelementeaufwands. In Abhängigkeit von den konkreten Forderungen können allerdings Echtzeitprobleme aufreten, da das gerade laufende Programm durch die Interruptroutine für die Anzeige zyklisch unterbrochen wird. Ein weiterer Nachteil besteht darin, daß einer der beiden Zähler-/Zeitgeberkanäle nicht mehr für allgemeine Anwendungen zur Verfügung steht. Kompromisse sind in einigen Fällen möglich. Bei-

Programm 9.8

```
Z8ASM     4.0
LOC    OBJ CODE    STMT SOURCE STATEMENT

                     1 AUSGABE2 MODULE
                     2
                     3 ! zeitmultiplexe Ansteuerung einer Ausgabeeinheit nach Bild 9.2          !
                     4
                     5 ! P 20...P23 Daten , P25...P27 Stelle, P24 Dez.pkt. (bei Bedarf)        !
                     6
                     7 GLOBAL
                     8
P 0000               9 ANZEIGE PROCEDURE
                    10
                    11 ENTRY
                    12
                    13                                          ! Variante U 882               !
P 0000 0033'        14 INTAB:   WVAL    DUMMY
P 0002 0033'        15          WVAL    DUMMY
P 0004 0033'        16          WVAL    DUMMY
P 0006 0033'        17          WVAL    DUMMY
P 0008 0033'        18          WVAL    DUMMY
P 000A 0026'        19 INTT1:   WVAL    CYCLE           ! Interruptvektor f. Timer 1           !
P 000C 31  F0       20 INIT:    SRP     #%F0            ! Steuerregistergruppe                 !
P 000E FC  80       21          LD      R15,#%80        ! Stack muss intern definiert sein     !
P 0010 2C  00       22          LD      R2,#0
P 0012 3C  27       23          LD      R3,#%27         ! Auffrischintervall Anzeige einstellen !
                    24                                  ! Veraenderung in Abhaengigkeit der
                    25                                    Systemtaktfrequenz sinnvoll          !
P 0014 BC  20       26          LD      R11,#%20        ! Interrupt Timer 1 erlaubt            !
P 0016 1C  0C       27          LD      R1,#%0C         ! Timer 1 laden und Start              !
P 0018 9F           28          EI
P 0019 31  10       29 MAIN:    SRP     #%10            ! R0...R3 Bilden Anzeigepuffer         !
                    30                                  ! Werte zwischen 0 und %F zulaessig    !
P 001B 46  E1  20   31          OR      R1,#%20         ! Stelleninformation einblenden        !
P 001E 46  E2  40   32          OR      R2,#%40
P 0021 46  E3  60   33          OR      R3,#%60
P 0024 8B  FE       34 HALT:    JR      HALT            ! normalerweise Fortsetzen             !
P 0026 A6  04  14   35 CYCLE:   CP      %4,#%14         ! hoechste Stelle erreicht ?           !
P 0029 7B  03       36          JR      C,ANZ1          ! hoechste Stelle nicht erreicht       !
P 002B E6  04  10   37          LD      %4,#%10         ! mit 1. Stelle neu beginnen           !
P 002E E5  04  02   38 ANZ1:    LD      2,@%4           ! Daten-und Stelleninformation ausgeben !
P 0031 20  04       39          INC     %4              ! naechste Stelle                      !
P 0033 BF           40 DUMMY:   IRET
                    41
P 0034              42 END ANZEIGE
                    43
                    44 END AUSGABE2
```

spielsweise kann die serielle Schnittstelle ebenfalls über den normalerweise für die Ziffernanzeige verwendeten Kanal T0 gesteuert und damit auf die Ziffernanzeige während der Datenübertragung verzichtet werden. Im Interrupt-Bedienprogramm muß in solchen Fällen eine Programmverzweigung in Abhängigkeit von der Betriebsart möglich sein. Allerdings sollte dann auch eine zusätzliche Dunkeltastung der Anzeige erfolgen (Beschaltung des RBI-Eingangs mit einer freien Ausgabeleitung), um eine Überlastung einzelner Segmente der in diesem Falle nicht mehr gemultiplexten Anzeige zu vermeiden.

9.4. Programme in Verbindung mit den Zähler-/Zeitgeberkanälen

Beide Zähler-/Zeitgeber werden vor allem benutzt, um Impulse definierter Dauer und Folgefrequenz zu erzeugen bzw. die genannten Parameter bei extern zugeführten Signalen zu messen. Die Benutzung des Zeitgebers T0 in Zusammenhang mit der seriellen Schnittstelle wurde bereits ausführlich beschrieben.

9.4.1. Erzeugung einer Rechteck-Impulsfolge

Die Ausgabe einer Rechteck-Impulsfolge mit konstanter Folgefrequenz stellt eine der einfachsten Anwendungen der Zähler-/Zeitgeberkanäle, nachfolgend auch Timer genannt, dar. Die Leitung P36 wird für den verwendeten Kanal als Ausgang definiert. Ein Verändern der Folgefrequenz ist durch Modifikation der eingetragenen Werte für Zählumfang und Vorteilerverhältnis in weiten Grenzen möglich. Die im Programm TIMER1 (Programm 9.9) geladenen Werte sind willkürlich festgelegt. Die erzeugte Rechteckspannung hat prinzipiell ein Impuls-Pausen-Verhältnis von 1. Die Ursache dafür ist ein zusätzliches Flipflop zwischen dem internen Ausgang des Rückwärtszählers und P36. Damit ist gleichzeitig eine zusätzliche Frequenzteilung um den Faktor 2 verbunden. Beim Laden der Zeitkonstanten ist das besonders zu berücksichtigen. Im Beispielprogramm TIMER1 arbeitet T1 mit dem internen Takt $f_{XTAL}:8$. Wird beim Laden des Vorteilerregisters PRE1 (%F3) das Bit D1 gesetzt, ist an P31 eine externe Taktquelle anschließbar, d.h., T1 arbeitet dann als Frequenzteiler. Der Gesamtteilungsfaktor ergibt sich aus der Multiplikation von Vorteilerwert und Zählumfang sowie dem Faktor 2, der das Ausgangs-Flipflop berücksichtigt. Demzufolge lassen sich bei dieser Anwendung nur geradzahlige Teilungsverhältnisse realisieren.

Programm 9.9

```
Z8ASM      4.0
LOC    OBJ CODE     STMT SOURCE STATEMENT

                      1 TIMER_1 MODULE
                      2
                      3         ! Programm zur Ausgabe einer Rechteckimpulsfolge an P36 !
                      4
                      5 GLOBAL
                      6         MAIN PROCEDURE
                      7         ENTRY
P 0812                8 $ABS %812
                      9
P 0812 E6  F7  01    10 TIMER1: LD      P3M,#%1          ! P34...P37 = AUSGAENGE
                     11                                    P30...P33 = EINGAENGE
                     12                                    P2 MIT AKTIVEN PULL-UP'S      !
P 0815 E6  F2  00    13         LD      T1,#%0           ! ZEITKONSTANTE T1            !
P 0818 E6  F1  84    14         LD      TMR,#%84         ! P36 = T1-AUSGANG
                     15                                    LADEN ZEITKONSTANTE T1        !
P 081B E6  F3  07    16         LD      PRE1,#%7         ! VORTEILER 1, AUTOM. NEUSTART !
P 081E 46  F1  08    17         OR      TMR,#%8          ! START TIMER 1               !
P 0821 8B  FE        18 ENDE:   JR      ENDE
                     19
P 0823               20         END MAIN
                     21 END TIMER_1
                     22
                     23
```

9.4.2. Messung der Impulsdauer

Eine Impulsdauermessung mit 16-Bit-Genauigkeit ist mit dem Progamm TIMER2 (Programm 9.10) möglich. Dabei wird die Torfunktion für P31 bei Verwendung von T1 ausgenutzt. P31 ist der Eingang der Meßeinrichtung. Solange das Eingangssignal H-Pegel führt, zählt T1 mit dem internen Takt. Die Rückflanke des Meßimpulses bewirkt den Abbruch der Zählung infolge der programmierten Torfunktion. Gleichzeitig wird die Interruptfähigkeit des Eingangs P31 genutzt. Das zugehörige Behandlungsprogramm ENDMESS lädt den erreichten Zählerstand in die Zielregister, setzt den aus Timer und Hilfsregister bestehenden 16-Bit-Zähler zurück und gibt ihn anschließend erneut frei. Außerdem erfolgt eine Zweierkomplementbildung des von T1 gelesenen Zählerstands.

Programm 9.10

```
Z8ASM     4.0
LOC     OBJ CODE     STMT SOURCE STATEMENT

                       1 TIMER_2 MODULE
                       2
                       3         ! Programm zur Messung eines H-Impulses an P31, Zaehlumfang
                       4           16 Bit.
                       5           Zielregister :  %6  -   H-Teil
                       6                           %7  -   L-Teil
                       7           Hilfsregister:  %4,%5
                       8           IRQ 2        :  Ende der Messung
                       9           IRQ 5        :  Uebertrag fuer H-Teil des Messwertes  !
                      10
                      11 GLOBAL
                      12         DAUER PROCEDURE
                      13         ENTRY
P 0800                14 $ABS %800
                      15
P 0800 8D  084F       16 INTTAB: JP       DUMMY
P 0803 8D  084F       17         JP       DUMMY
P 0806 8D  082D       18         JP       ENDMES
P 0809 8D  084F       19         JP       DUMMY
P 080C 8D  084F       20         JP       DUMMY
P 080F 8D  084D       21         JP       HIGH
                      22
P 0812 E6  FF  80     23 BEGIN:  LD       SPL,#%80        ! LADEN STACKPOINTER              !
P 0815 E6  F7  01     24         LD       P3M,#%1         ! P34...P37 = AUSGAENGE           !
                      25                                  ! P30...P33 = EINGAENGE           !
                      26                                  ! P2 mit aktiven Pull-up's        !
P 0818 E6  F2  00     27         LD       T1,#0           ! Zeitkonstante T1                !
P 081B E6  F1  94     28         LD       TMR,#%94        ! P36 = T1-AUSGANG                !
                      29                                  ! P31 = TOR-Eingang               !
                      30                                  ! laden Zeitkonstante T1          !
P 081E E6  F3  29     31         LD       PRE1,#%29       ! Vorteiler 10, autom. Neustart !
P 0821 E6  FB  24     32         LD       IMR,#%24        ! IRQ2 (P31) UND IRQ5 (T1)        !
P 0824 E6  F6  00     33         LD       P2M,#0          ! P2 alles Ausgaenge              !
P 0827 E6  04  00     34         LD       4,#0            ! Hilfsregister 4                 !
P 082A 9F             35         EI
P 082B 8B  FE         36 ENDE:   JR       ENDE
                      37
P 082D 56  F1  F7     38 ENDMES: AND      TMR,#%F7        ! Timer 1 stoppen                 !
P 0830 E4  F2  05     39         LD       5,T1            ! Zaehlwert in Register 5 merken!
P 0833 76  FA  20     40         TM       IRQ,#%20        ! IRQ5 angemeldet ? (Uebertrag) !
P 0836 6B  02         41         JR       Z,ENDME1        ! NEIN                            !
P 0838 20  04         42         INC      4               ! ja, High-Teil +1                !
P 083A E6  FA  00     43 ENDME1: LD       IRQ,#0          ! Interruptanmeldungen loeschen !
P 083D E4  04  06     44         LD       6,4             ! H-Teil auf Zielregister         !
P 0840 B0  04         45         CLR      4
P 0842 46  F1  0C     46         OR       TMR,#%0C        ! Timer laden und starten         !
P 0845 60  05         47         COM      5
P 0847 20  05         48         INC      5               ! Zweierkomplementbildung         !
P 0849 E4  05  07     49         LD       7,5             ! L-Teil auf Zielregister         !
P 084C BF             50         IRET
                      51
P 084D 20  04         52 HIGH:   INC      4               ! H-Teil + 1 bei Uebertrag        !
P 084F BF             53 DUMMY:  IRET
P 0850                54         END DAUER
                      55 END TIMER_2
```

Hinsichtlich der weiteren Auswertung wird so ein scheinbares Vorwärtszählen von T1 erreicht. Damit steht in den Zielregistern ein Wert zur Verfügung, der proportional zur Dauer des am Meßeingang anstehenden Impulses ist.

Der H-Teil des 16-Bit-Zählers wird durch ein Universalregister gebildet. Es wird bei jedem Übertrag bzw. Nulldurchgang des Timers inkrementiert. Dieser Vorgang läuft wiederum als Interruptprogramm ab. Innerhalb des Programmteils ENDMESS muß getestet werden, ob eine Anmeldung des Interrupts HIGH vorliegt. In diesem Falle muß die geforderte Inkrementierung des H-Zählregisters noch ausgeführt werden, sonst ist das Ergebnis fehlerhaft. Das anschließende Löschen des Interrupt-Anmelderegisters IRQ ist notwenig, um ein nochmaliges Inkrementieren nach dem Verlassen von ENDMESS zu verhindern. Es muß hervorgehoben werden, daß die Programmlaufzeit und insbesondere die Interrupt-Annahmezeit auf die Genauigkeit des Meßergebnisses keinen Einfluß haben. Daher ist das Prinzip besonders für die Messung im Mikrosekundenbereich geeignet.

9.4.3. Messung der Periodendauer einer Impulsfolge

Das Programm TIMER3 (Programm 9.11) kann vor allem im Zusammenhang mit der genauen Messung von kleinen Drehzahlen bzw. niedrigen Frequenzen angewendet werden. Es gibt die Periodendauer der an P31 angeschlossenen Rechteckspannung als 16-Bit-Meßwert aus. Durch eine nachfolgende Umrechnung (z. B. mit dem Programm DIV, s. Abschnitt 9.1.) kann die Beziehung $f = 1 / T$ (f – Frequenz, T – Periodendauer) realisiert werden. Für Drehzahlmessungen ist der Zähler des Bruches durch einen entsprechenden Umrechnungsfaktor zu ersetzen.

Gegenüber dem Prinzip des Zählfrequenzmessers ergibt sich ein wesentlicher Vorteil bei Anforderungen an das dynamische Verhalten der Meßeinrichtung. Er besteht in der Tatsache, daß nach jeder Periode (bzw. Umdrehung) ein neuer Meßwert verfügbar ist. Demzufolge ist die minimale Meßzeit identisch mit der Periodendauer. Bei

Programm 9.11

```
Z8ASM      4.0
LOC     OBJ CODE     STMT SOURCE STATEMENT

                       1 TIMER_3 MODULE
                       2
                       3         ! Programm zur Periodendauermessung einer Impulsfolge
                       4           mit Benutzung von T1                                  !
                       5         ! Zielregister :  %6 - Highteil
                       6                           %7 - Lowteil der Periodendauer
                       7           Hilfsregister:  %5                                    !
                       8
                       9 GLOBAL
                      10         PERIODE PROCEDURE
                      11         ENTRY
P 0800                12 $ABS %800
                      13
P 0800 8D  084F       14 INTTAB: JP      DUMMY
P 0803 8D  084F       15         JP      DUMMY
P 0806 8D  0830       16         JP      INT2            ! Stopp Zaehler              !
P 0809 8D  084F       17         JP      DUMMY
P 080C 8D  084F       18         JP      DUMMY
P 080F 8D  084D       19         JP      INT5            ! Uebertrag                  !
                      20
P 0812 E6  FF  80     21 BEGINN: LD      SPL,#%80        ! Stackpointer laden         !
P 0815 E6  F7  01     22         LD      P3M,#%1         ! P34...P37 = Ausgaenge      !
                      23                                 ! P30...P33 = Eingaenge      !
                      24                                 ! P2 mit aktiven Pull-up's   !
P 0818 E6  F6  00     25         LD      P2M,#%0         ! P2 alles Ausgaenge         !
P 081B E6  F2  00     26         LD      T1,#0           ! Zeitkonstante Timer 1      !
P 081E E6  F1  84     27         LD      TMR,#%84        ! P36 = T1-Ausgang           !
                      28                                 ! Laden Zeitkonstante T1     !
P 0821 E6  F3  9F     29         LD      PRE1,#%9F       ! Vorteiler laden, andere Werte !
                      30                                 ! sind moeglich              !
                      31                                 ! T1 intern, automat. Neustart !
P 0824 E6  FB  24     32         LD      IMR,#%24        ! IRQ2 und IRQ5 erlaubt      !
P 0827 B0  05         33         CLR     5
P 0829 B0  06         34         CLR     6
P 082B B0  07         35         CLR     7               ! alle Register loeschen     !
P 082D 9F             36         EI
P 082E 8B  FE         37 ENDE:   JR      ENDE
                      38
P 0830 56  F1  F7     39 INT2:   AND     TMR,#%F7        ! Timer 1 stoppen            !
P 0833 E4  F2  07     40         LD      7,T1            ! Zaehlwert abspeichern      !
P 0836 76  FA  20     41         TM      IRQ,#%20        ! IRQ5 angemeldet ?          !
P 0839 6B  02         42         JR      Z,INT21         ! nein                       !
P 083B 20  05         43         INC     5               ! Highteil + 1               !
P 083D E6  FA  00     44 INT21:  LD      IRQ,#0          ! Interruptanmeldungen loeschen !
P 0840 E4  05  06     45         LD      6,5             ! Highteil in Zielregister   !
P 0843 B0  05         46         CLR     5
P 0845 46  F1  0C     47         OR      TMR,#%0C        ! Timer 1 laden und starten  !
P 0848 60  07         48         COM     7               ! Zweierkomplementbildung    !
P 084A 20  07         49         INC     7
P 084C BF             50         IRET
                      51
P 084D 20  05         52 INT5:   INC     5               ! H-TeIL + 1                 !
P 084F BF             53 DUMMY:  IRET
                      54
P 0850                55         END PERIODE
                      56 END TIMER_3
                      57
```

Zählfrequenzmessern muß die Meßzeit dagegen in Abhängigkeit von der geforderten Genauigkeit um mehrere Zehnerpotenzen größer gewählt werden. Demnach hat dieses Meßprinzip vor allem bei Problemen der Steuerungs- und Regelungstechnik Bedeutung.

Das Programm TIMER3, das nur die eigentliche Periodendauermessung beinhaltet, verwirklicht softwaremäßig wiederum einen 16-Bit-Zähler, bestehend aus dem Zählkanal T1 (L-Teil) und einem Universalregister (H-Teil). Bei jeder HL-Flanke des Meßsignals entsteht eine Interruptforderung. Das zugehörige Behandlungsprogramm liest den aktuellen Zählerstand auf die Zielregister und setzt den Zähler anschließend zurück.

Bei Verwendung des Programms zur Frequenzmessung im Kilohertzbereich kann die Laufzeit des Interrupt-Behandlungsprogramms bereits zu nachweisbaren Fehlern führen. Eine Kompensation dieses systematischen Fehlers ist möglich, indem zum Meßergebnis ein Korrekturwert addiert wird.

9.5. Verarbeitung von Analogsignalen

Bei vielen praktischen Anwendungen liegen die Meßgrößen in analoger Form vor, was den Einsatz von Analog-Digital-Umsetzern (ADU) erforderlich macht. Stehen Umsetzer mit paralleler Datenausgabe, z. B. *C571*, zur Verfügung, ist deren Anschluß an entsprechende Eingänge des Rechnerschaltkreises ohne zusätzlichen Hard- und Softwareaufwand möglich. Oft stehen dem Amateur derartige ADU jedoch nicht zur Verfügung. Als Alternative kann in diesen Fällen auf den besser verfügbaren *C520D* zurückgegriffen werden, wenn die relativ große Umsetzdauer von etwa 10 ms nicht nachteilig ist. Bild 9.3 zeigt die Ankopplung dieses Schaltkreises an ein EMR-Tor. Das zugehörige Übernahmeprogramm legt den zeitmultiplex ausgegebenen Meßwert in einem Puffer (Universalregister) ab. Es ist so gestaltet, daß eine Unterbrechung durch Interrupts generell möglich wird. Allerdings darf diese Unterbrechung nicht länger als 1 ms dauern. Das Übernahmeprogramm seinerseits läßt eine Interruptannahme während der Übertragung des gewandelten Werts auf die Zielregister nicht zu, weil in dieser Phase Interruptverbot besteht. Das nachfolgend vorgestellte Programm ADU (Programm 9.12) legt den gewandelten Meßwert in 2 aufeinanderfolgenden Registern ab, deren Adresse durch einen Zeiger (im Beispiel %23) vor dem Programmaufruf festgelegt werden kann. Da vom *C520D* nur 3 Dezimalstellen zur Verfügung stehen, ist die höherwertige Tetrade des 16-Bit-Zielregisters stets 0. In den verbleibenden 3 Tetraden stehen die BCD-Informationen in der Reihenfolge MSD, NSD, LSD.

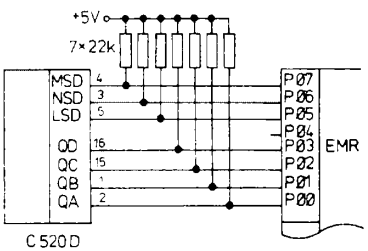


Bild 9.3
Anschluß des ADU C520D an ein EMR-Port

Programm 9.12

```
Z8ASM      4.0
LOC    OBJ CODE     STMT SOURCE STATEMENT

                      1 ADU MODULE
                      2 ! Programm zum Einlesen eines Analogwertes mit ADU C 520 D        !
                      3 ! Der Analogwert wird in 2 Zielregistern (H-Teil,L-Teil) uebergeben.
                      4 Die Adresse des Zielregisters wird durch den Inhalt von %23 definiert !
                      5 GLOBAL
                      6 MAIN PROCEDURE
                      7 ENTRY
P 0812                8 $ABS %812
P 0812 31  20         9  BEGIN: SRP     #%20
P 0814 D6  084F      10          CALL    WAIT            ! Umsetzzeit abwarten          !
P 0817 1C  00        11          LD      R1,#0           ! Zielregister loeschen        !
P 0819 2C  00        12          LD      R2,#0
P 081B D6  0858      13 M1:      CALL    READ            ! ADU lesen                    !
P 081E 76  E0  80    14          TM      R0,#%80         ! MSD ?                        !
P 0821 EB  F8        15          JR      NZ,M1           ! nein, neu versuchen          !
P 0823 18  E0        16          LD      R1,R0           ! Wert uebernehmen             !
P 0825 56  E1  0F    17          AND     R1,#%0F         ! Digits ausblenden            !
P 0828 D6  0858      18 M2:      CALL    READ
P 082B 76  E0  20    19          TM      R0,#%20         ! LSD ?                        !
P 082E EB  F8        20          JR      NZ,M2           ! Nein, neu versuchen          !
P 0830 56  E0  0F    21          AND     R0,#%0F         ! Digits ausblenden            !
P 0833 28  E0        22          LD      R2,R0           ! Wert uebernehmen             !
P 0835 D6  0858      23 M3:      CALL    READ
P 0838 76  E0  40    24          TM      R0,#%40         ! NSD ?                        !
P 083B EB  F8        25          JR      NZ,M3           ! Nein, neu versuchen          !
P 083D F0  E0        26          SWAP    R0              ! NSD auf obere Tetrade von R0 !
P 083F 56  E0  F0    27          AND     R0,#%F0         ! Untere tetrade ausblenden    !
P 0842 42  20        28          OR      R2,R0           ! NSD einblenden               !
P 0844 70  FB        29          PUSH    IMR             ! Interruptzustand retten      !
P 0846 8F            30          DI                      ! Ungestoerte Uebernahme       !
P 0847 F3  31        31          LD      @R3,R1          ! Zielregister,H-Teil laden    !
P 0849 3E            32          INC     R3
P 084A F3  32        33          LD      @R3,R2          ! Zielregister,L-Teil laden    !
P 084C 50  FB        34          POP     IMR             ! alten INT-Zustand herstellen !
P 084E AF            35          RET
                     36
P 084F 4C  02        37 WAIT:    LD      R4,#2           ! Zeitkonstante ca.10ms        !
P 0851 5C  00        38 W0:      LD      R5,#0
P 0853 5A  FE        39 W1:      DJNZ    R5,W1
P 0855 4A  FA        40          DJNZ    R4,W0
P 0857 AF            41          RET
P 0858 08  00        42 READ:    LD      R0,0            ! Port 0 lesen                 !
P 085A A4  00  E0    43          CP      R0,0            ! Gelesene Daten stabil ?      !
P 085D EB  F9        44          JR      NZ,READ         ! Nein, neuer Leseversuch      !
P 085F AF            45          RET
P 0860               46 END MAIN
                     47 END ADU
```

Das Programm ADU berücksichtigt durch die Möglichkeit der indirekten Adressierung des Zielregisters vor dem Aufruf eine weitere häufige Problemstellung, nämlich die Messung mehrerer Analoggrößen mit einem Umsetzer. Die benötigten Kanäle lassen sich mit Analogschaltern bzw. -multiplexern aufbauen. Zur Unterstützung dieser Betriebsweise ist im Übernahmeprogramm eine zusätzliche Zeitschleife WAIT von etwa 10 ms installiert. Sie ist notwendig, um nach dem Einschalten eines Kanals die Umsetzzeit des Wandlers zu berücksichtigen. Bei Verzicht auf diese Maßnahme ist nicht mehr garantiert, daß der übernommene Wert tatsächlich dem Eingangspegel am aktivierten Kanal entspricht.

10. Eingabegerät mit Einchipmikrorechner und Ankopplung an einen übergeordneten Rechner

Mit einem Einchipmikrorechner läßt sich bei geringem Hardwareaufwand ein Eingabegerät für numerische Daten (Ziffern) realisieren. Beim ausgeführten Muster können 4stellige Dezimalzahlen einschließlich Kommastelle eingegeben werden. Als Empfänger bzw. Auswerterechner ist im Beispiel ein Kleincomputer *KC 85/1* oder *KC 87* vorgesehen. Der Einsatz anderer Rechner ist ohne große Schwierigkeiten durch geringfügige Softwareanpassung möglich.

Beim vorgestellten Gerät wird die serielle Schnittstelle in der Grundversion nicht genutzt, es erfolgt im Gegensatz zu den nachfolgenden Beispielen eine parallele Datenübertragung. Die Ein-/Ausgabeleitungen der seriellen Schnittstelle sind jedoch frei gehalten, so daß eine entsprechende Modifikation möglich ist. Das Verfahren der parallelen Übertragung bedingt eine relativ große Zahl an Übertragungsleitungen. Weiterhin besteht normalerweise keine galvanische Trennung zwischen Sender und Empfänger. Diesen Nachteilen stehen im Amateurbereich jedoch auch Vorteile gegenüber. Zum Beispiel sind die Forderungen an die Konstanz bzw. Übereinstimmung der Systemtaktfrequenzen aller beteiligten Einheiten sehr gering. Es kann demnach ohne weiteres auf den Einsatz von Schwingquarzen zugunsten billiger *LC*- oder *RC*-Varianten verzichtet werden. Ein weiterer Vorteil besteht in der übersichtlichen Gestaltung der Software vor allem unter dem Aspekt, daß mehrere Geräte quasi gleichzeitig Daten übertragen sollen. Die nachfolgend beschriebene Hardware erlaubt den Einsatz von maximal 8 Geräten.

Ein weiterer Aspekt, der die parallele Datenübertragung rechtfertigt, ist die Fehlersuche im Störungsfall. Während bei einem seriell arbeitenden Ringsystem nach Möglichkeit ein Logikanalysator oder Speicheroszillograf vorhanden sein sollte, kann die Funktionsfähigkeit der parallelen Datenübertragung mit einem Serviceoszillograf und einer einfachen LED-Anzeige nachgewiesen werden. Letztlich wurde auch der Tatsache Rechnung getragen, daß an den meisten Rechnern (insbesondere Klein- und Heimcomputern) parallele E/A-Möglichkeiten vorhanden sind, eine serielle Schnittstelle jedoch fehlt.

10.1. Hardware

Den Stromlaufplan des Geräts zeigt Bild 10.1. An Tor 2 ist die numerische Tastatur, bestehend aus 13 Halltastern *TSH 19,* angeschlossen. Neben den Zifferntasten 0 bis 9 und der Dezimalpunkteingabe existiert noch eine zusätzliche Taste für den Abschluß der Eingabe (ENTER). Die Löschung fehlerhafter Eingaben ist ebenfalls über eine Sondertaste (CLEAR) möglich.

Tor 1 ist zur Datenübertragung vorgesehen. Die Ausgangstreiber *V 40098* haben Tristate-Ausgänge. Die Sendedaten werden auf den Übertragungsbus durchgeschaltet,

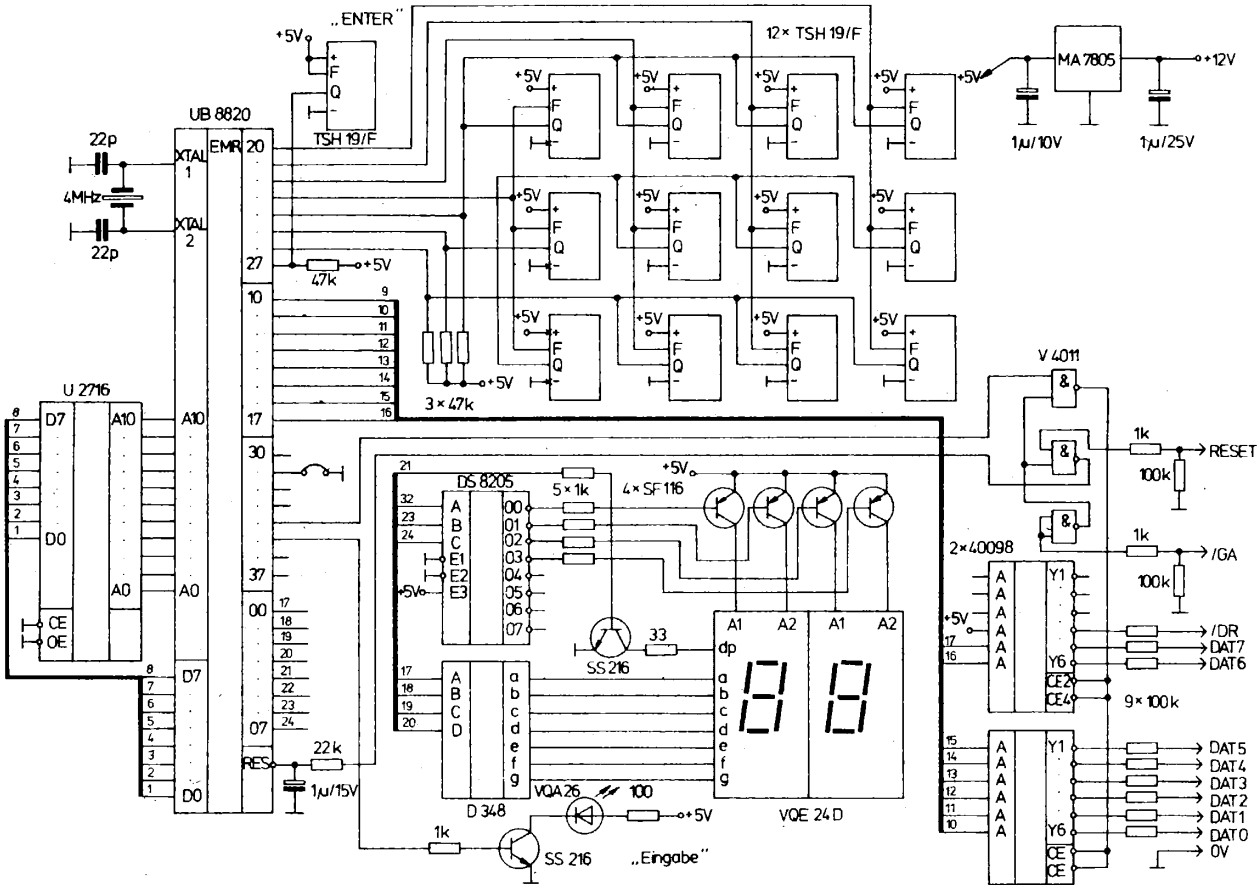


Bild 10.1 Eingabegerät, Stromlaufplan

wenn das Geräte-Anwahlsignal /GA anliegt und die Eingabe durch ENTER abgeschlossen ist. Das Datenwort an Tor 1 hat folgenden Aufbau:

```
D7 D6 D5 D4 D3 D2 D1 D0
 I  I  I--I--I--I--I--I-  Datenbit D0 bis D5
 I  I------------------   Paritätsbit
 I---------------------   Synchronsignal SYN
                          0: Daten ungültig
                          1: Daten gültig
```

Durch Verwendung von Bit D7 für die Ausgabe des Synchronsignals vereinfacht sich die Software auf der Empfängerseite bei gleichzeitig gesteigerter Übertragungssicherheit. Da auch ein Paritätsbit generiert und übertragen wird, stehen für die eigentlichen Daten nur noch 6 Bit zur Verfügung. Für die Übertragung von Ziffern im ASCII-Format ist das ausreichend. Bei einem eventuellen Ausbau des Geräts für alphanumerische Zeicheneingabe ist zu beachten, daß sich mit dem gewählten Datenformat nur Großbuchstaben übertragen lassen.

Über die Geräteanwahlsignale wird das Zusammenspiel zwischen Eingabegeräten und Rechner koordiniert, indem stets nur ein Gerät Daten senden kann. Konflikte auf dem Übertragungsbus sind damit ausgeschlossen. Trotzdem befinden sich an den Datentreiberausgängen zusätzliche Schutzwiderstände, die aber in 1. Linie der Strombegrenzung bei eventuell auftretenden Spannungsspitzen, hervorgerufen durch eingekoppelte Störungen, dienen. Die Ziffernausgabe erfolgt zeitmultiplex über Tor 0 des EMR. An P00 bis P03 werden die BCD-Daten bereitgestellt, P05 bis P07 beinhalten die Stelleninformation. Zur Dekodierung der Ausgabestelle ist ein *DS 8205* eingesetzt, der ausgangsseitig die Digit-Treibertransistoren ansteuert. Als Anzeigebauelemente verwendet man 2× *VQE 24 D*. Über die Leitung P04 kann der Dezimalpunkt eingeschaltet werden. Die Ausgabe erfolgt auch hierbei zeitmultiplex. Damit ergibt sich für das Datenwort an Tor 0 folgender Aufbau:

```
D7 D6 D5 D4 D3 D2 D1 D0
I  I  I   I  I--I--I--I--------  BCD-Ziffer 0 bis 9
I  I  I   I-------------------   0: Kein Dezimalpunkt
I  I  I                         1: Dezimalpunkt
I--I--I-----------------------   Anzeigestelle 0 bis 7
```

Für das Zusammenspiel des Geräts mit dem Rechner sind neben dem bereits erwähnten Geräteanwahlsignal noch 2 weitere Steuerleitungen vorhanden. Durch H-Pegel am Eingang RESET wird das Gerät in den Grundzustand versetzt, d.h., es erfolgt ein Hardware-RESET. Voraussetzung dafür ist die Aktivierung des Geräts mit /GA = 0. Nach RESET erscheint in der Anzeige des angewählten Geräts rechtsbündig eine 0, alle anderen Stellen bleiben dunkel. Gleichzeitig wird dem Bediener der Eingabestatus durch eine zusätzliche LED mitgeteilt. Durch den Aufruf des Geräts über ein Hardware-RESET ist immer gewährleistet, daß eine eventuell fehlerhafte Programmabarbeitung, z. B. hervorgerufen durch Störungen auf den Versorgungsleitungen, mit der Anforderung neuer Eingabedaten beendet wird. Ist die Eingabe mit der ENTER-Taste abgeschlossen, wird das Statussignal /DR (data ready) gesetzt. Durch die Verknüpfung mit RESET ist gewährleistet, daß ein Lesen der gesendeten Daten erst nach ordnungsgemäßem Abschluß der Eingabeoperation erfolgen kann.

Ein Festspannungsregler *MA 7805* übernimmt beim Muster die Stromversorgung des Geräts. Die Rohspannung kann dann zwischen 9 V und 12 V liegen. Als typischer Wert für die Stromaufnahme sind 0,7 A zu veranschlagen.

10.2. Steuerprogramm des Eingabegeräts

Das Steuerprogramm EINGABE (Programm 10.1) hat 3 wesentliche Teile:

INIT: Initialisierung des EMR, Setzen der Anzeige in den Grundzustand, Rücksetzen von /DR.

TAST: Tastaturabfrage, Eintragen der entsprechenden Zeichen in den Anzeigepuffer.

SEND: Datenübertragung durch zyklisches Senden der im Sendepuffer befindlichen Daten, einschließlich Paritätsbildung und Anfügen der Kennbytes.

Das vollständige Listing der für das Eingabegerät entwickelten Software ist im Programm 10.1 dargestellt. Zur Programmentwicklung wurde ursprünglich das in Abschnitt 13 vorgestellte Minimalsystem benutzt, d. h., es stand kein Editor zur Verfügung. Auf eine spätere Überarbeitung mit dem entsprechenden Programmpaket des Personalcomputers *PC1715* wurde weitestgehend verzichtet. Damit soll vor allem gezeigt werden, daß auch komplexe Programme für EMR mit bescheidenen Mitteln erstellbar sind.

Nach der Initialisierung erfolgt ein Sprung in das Tastaturprogramm. Alle Tasten werden über eine 3×4-Matrix abgefragt. Die Spaltenleitungen P20 bis P23 sind mit den entsprechenden Freigabeeingängen der Taster verbunden. Jeweils 4 Schalter *TSH19* bilden eine Gruppe. Als Zeilenleitungen finden P24 bis P26 Verwendung. An jede dieser Leitungen sind die Schalterausgänge einer Gruppe angeschlossen. Die eingelesene Information dient als Offsetwert für die Berechnung einer Registeradresse, deren Inhalt das Zeichen der zugeordneten Taste liefert. Demzufolge steht am Anfang des Tastaturprogramms das Einlesen der Zeichentabelle in die Register %20 bis %2F.

Prinzipiell kann die Tabelle auch direkt im EPROM abgelegt und über LDE- oder LDC-Befehle gelesen werden. Mit Verlassen des Programms steht das erkannte Zeichen in der niederwertigsten Stelle des Anzeigepuffers. Vorher realisiert das Unterprogramm SHIFT eine Verschiebung des Puffers um eine Stelle nach links. Eingabe und Anzeige arbeiten somit in der vom Taschenrechner her gewohnten Weise.

Bei Eingabe des Dezimalpunkts wird ein spezielles Flag gesetzt, so daß weitere Betätigungen der entsprechenden Taste wirkungslos bleiben. Ein Verlassen des Tastaturprogramms ist nur möglich, wenn die ENTER-Taste betätigt oder ein Hardware-RESET durchgeführt wird. Im Normalfall, also dem ordnungsgemäßen Abschluß der Eingabe mit ENTER, erfolgt der Eintritt in das Übertragungsprogramm SEND. Es erfüllt die Teilaufgaben:

- Umwandlung des Anzeigepuffer-Inhalts in ASCII-Zeichen,
- Anfügen von 2 Vorbytes und 1 Endebyte,
- Paritätsbitgenerierung,
- zyklisches Ausgeben der gesamten Zeichenfolge einschließlich Bildung des Synchronsignals SYN.

Die Sendefrequenz, d. h. die Periodendauer des SYN-Signals, ist über P31 für Testzwecke umschaltbar. Wird dieser Eingang auf H-Pegel gelegt, findet das Senden im Testmodus statt. Jedes Zeichen wird dabei etwa 1 s lang ausgegeben. Durch den Anschluß von 8 LEDs unmittelbar an die Datenausgänge des Geräts kann festgestellt werden, ob alle Zeichen einschließlich Paritätsbit richtig gesendet werden. Die zur Aktivierung der Datentreiber notwendigen Signalpegel an den Steuereingängen RESET

Programm 10.1

```
Z8ASM     4.0
LOC    OBJ CODE     STMT SOURCE STATEMENT

                      1 EINGABE MODULE
                      2
                      3 ! Steuerprogramm fuer Eingabegeraet mit paralleler Schnittstelle !
                      4
                      5 GLOBAL
                      6
P 0000                7 KEY PROCEDURE
                      8
                      9 ENTRY
                     10
P 0000 0119'         11 ITAB:   WVAL    DUMMY          ! U 882                          !
P 0002 0119'         12         WVAL    DUMMY
P 0004 0119'         13         WVAL    DUMMY
P 0006 0119'         14         WVAL    DUMMY
P 0008 0119'         15         WVAL    DUMMY
P 000A 00FD'         16 IVE:    WVAL    ANZ
P 000C 31  F0        17         SRP     #%F0
P 000E FC  80        18         LD      R15,#%80       ! Stack einstellen               !
P 0010 7C  01        19         LD      R7,#1          ! P3 Ein/Ausgaenge,P2 pull up    !
P 0012 6C  F0        20         LD      R6,#%F0
P 0014 8C  04        21         LD      R8,#4
P 0016 2C  00        22         LD      R2,#0          ! Zeitkonstante T1               !
P 0018 3C  27        23         LD      R3,#%27        ! Auffrischintervall Anzeige     !
P 001A 1C  00        24         LD      R1,#0
P 001C BC  20        25         LD      R11,#%20       ! INT T1 freigegeben             !
P 001E AC  00        26         LD      R10,#0
P 0020 9C  09        27         LD      R9,#%9
P 0022 B0  00        28         CLR     0
P 0024 B0  01        29         CLR     1
P 0026 B0  02        30         CLR     2
P 0028 B0  03        31         CLR     3              ! P0 bis P3 reset                !
P 002A 46  03  10    32         OR      3,#%10         ! INPUT-LED ein, READY aus       !
P 002D D6  0254'     33         CALL    TIME1          ! ca. 0,5s warten                !
P 0030 56  03  DF    34 CLEAR:  AND     3,#%DF         ! Fertigmeldung geloescht        !
P 0033 E6  16  00    35         LD      %16,#0
P 0036 E6  04  10    36         LD      %4, #%10       ! Zeiger auf Pufferanfang        !
P 0039 E6  10  00    37         LD      %10,#0         ! Anzeige initialisieren         !
P 003C E6  11  2F    38         LD      %11,#%2F       ! Stellen 2 bis 4 dunkel         !
P 003F E6  12  4F    39         LD      %12,#%4F
P 0042 E6  13  6F    40         LD      %13,#%6F
P 0045 46  F1  0C    41         OR      TMR,#%0C       ! Timer 1 laden und starten      !
P 0048 9F            42         EI                     ! Anzeige laeuft                 !
P 0049 E6  14  00    43         LD      %14,#0         ! Merkregister f. Dez.punkt      !
P 004C D6  017E'     44 INPUT:  CALL    TAST           ! Tastaturabfrage                !
P 004F A6  16  0C    45         CP      %16,#%0C       ! ENTER ?                        !
P 0052 6B  1F        46         JR      Z,SEND         ! ja, Sendebeginn                !
P 0054 A6  16  0B    47         CP      %16,#%0B       ! CLEAR betaetigt ?              !
P 0057 6B  D7        48         JR      Z,CLEAR        ! Anzeige loeschen,Neueingabe    !
P 0059 A6  16  0A    49         CP      %16,#%0A       ! Dezimalpunkt ?                 !
P 005C 6B  08        50         JR      Z,POINT
P 005E D6  011A      51         CALL    SHIFT          ! Anzeige eine Stelle n.links    !
P 0061 E4  16  10    52         LD      %10,%16        ! neues Zeichen in 1.Stelle      !
P 0064 8B  E6        53         JR      INPUT
                     54
P 0066 76  14  02    55 POINT:  TM      %14,#2         ! Dezimalpunkt bereits vorh.?    !
P 0069 EB  03        56         JR      NZ,DPIN        ! ja, ignorieren                 !
P 006B 46  10  10    57         OR      %10,#%10       ! nein, eintragen                !
P 006E 46  14  02    58 DPIN:   OR      %14,#2         ! Flag f. Dezimalpkt. setzen     !
P 0071 8B  D9        59         JR      INPUT          ! Eingabe naechstes Zeichen      !
P 0073 56  03  EF    60 SEND:   AND     3,#%EF         ! INPUT-LED aus                  !
P 0076 E6  30  2A    61         LD      %30,#%2A
P 0079 E6  31  3C    62         LD      %31,#%3C       ! 2 Vorbytes                     !
P 007C E6  40  10    63         LD      %40,#%10       ! Zeiger auf 1.Zeichen           !
P 007F E6  41  32    64         LD      %41,#%32       ! Zeiger fuer Sendepuffer        !
P 0082 E5  40  44    65 SEND1:  LD      %44,@%40       ! Zeichen in Pufferregister      !
P 0085 76  44  10    66         TM      %44,#%10       ! Dezimalpunkt gesetzt ?         !
P 0088 6B  05        67         JR      Z,SEND2        ! nein                           !
P 008A E7  41  2E    68         LD      @%41,#%2E      ! ja, Zeichen f. Dez.pkt.        !
P 008D 20  41        69         INC     %41            ! Naechstes Zeichen              !
P 008F E5  40  44    70 SEND2:  LD      %44,@%40       ! ASCII Zeichen herstellen       !
P 0092 56  44  0F    71         AND     %44,#%0F
P 0095 46  44  30    72         OR      %44,#%30
P 0098 F5  44  41    73         LD      @%41,%44       ! In Sendepuffer eintragen       !
P 009B 20  40        74         INC     %40            ! naechstes Zeichen              !
P 009D 20  41        75         INC     %41            ! Sendepuffer inkrementieren     !
P 009F A6  40  14    76         CP      %40,#%14       ! letztes Zeichen ?              !
P 00A2 7B  DE        77         JR      C,SEND1        ! nein, naechstes Zeichen        !
P 00A4 E7  41  00    78         LD      @%41,#0        ! ja, Endbyte in Sendepuffer     !
P 00A7 20  41        79         INC     %41
P 00A9 E6  45  30    80         LD      %45,#%30       ! Zeiger f. Paritaetspruefng.    !
P 00AC E5  45  46    81 PARIN:  LD      %46,@%45
P 00AF 56  46  3F    82         AND     %46,#%3F       ! Synchronsignal ausblenden      !
```

```
P 00B2 A6  46  00     63          CP      %46,#0          ! Endbyte ?                          !
P 00B5 6B  22         64          JR      Z,SEND3         ! ja,keine Paritaetspruefng.         !
P 00B7 E6  47  06     65          LD      %47,#%6         ! Zahl zu pruefender Bytes           !
P 00BA B0  48         66          CLR     %48
P 00BC CF             67 PAR:     RCF
P 00BD E0  46         68          RR      %46             ! ein Bit pruefen                    !
P 00BF FB  02         69          JR      NC,NEXT         ! Bit = 0                            !
P 00C1 20  48         70          INC     %48             ! Bit = 1, Par.Zaehler +1            !
P 00C3 31  40         71 NEXT:    SRP     #%40
P 00C5 7A  F5         72          DJNZ    R7,PAR          ! naechstes Bit pruefen              !
P 00C7 76  48  01     73          TM      %48,#1          ! ungeradzahliges Ergebnis ?         !
P 00CA EB  03         74          JR      NZ,UNGE         ! ja, Paritaetsbit = 0               !
P 00CC 47  45  40     75          OR      @%45,#%40       ! nein, Paritaetsbit = 1             !
P 00CF 20  45         76 UNGE:    INC     %45             ! naechstes Zeichen                  !
P 00D1 8B  D9         77          JR      PARIN
P 00D3 E6  43  30     78          LD      %43,#%30        ! Sendezeiger auf 1. Byte            !
P 00D6 E5  43  42     79 NZEI:    LD      %42,@%43
P 00D9 56  42  7F     80 SEND3:   AND     %42,#%7F        ! Synchronsignal ausblenden          !
P 00DC E4  42  01     81          LD      1,%42           ! Zeichen auf Port                   !
P 00DF 46  03  20     82          OR      3,#%20          ! READY Meldung ausgeben             !
P 00E2 D6  020E'      83          CALL    TIMEL           ! Synchronsignal, L-Phase            !
P 00E5 46  01  80     84          OR      1,#%80
P 00E8 D6  0222'      85          CALL    TIMEH           ! Synchronsignal, H-Phase            !
P 00EB 56  01  7F     86          AND     1,#%7F
P 00EE D6  020E'      87          CALL    TIMEL
P 00F1 20  43         88          INC     %43             ! Sendezeiger auf n. Zeichen         !
P 00F3 A4  41  43     89          CP      %43,%41         ! letztes Zeichen gesendet ?         !
P 00F6 7B  DE         90          JR      C,NZEI          ! nein, weiter                       !
P 00F8 E6  43  30     91          LD      %43,#%30        ! ja, Neubeginn Sendezyklus          !
P 00FB 8B  D9         92          JR      NZEI
P 00FD A6  04  14     93 ANZ:     CP      %4,#%14         ! letzte Stelle ?                    !
P 0100 7B  03         94          JR      C,ANZ1          ! nein, naechste Stelle              !
P 0102 E6  04  10     95          LD      %4,#%10         ! auf 1. Stelle                      !
P 0105 71  04         96 ANZ1:    PUSH    @%4             ! Anzeigestelle retten               !
P 0107 57  04  0F     97          AND     @%4,#%0F        ! Stelleninformation ausblenden      !
P 010A A7  04  0F     98          CP      @%4,#%0F        ! Dunkeltastung ?                    !
P 010D 51  04         99          POP     @%4
P 010F EB  03        100          JR      NZ,ANZ2         ! nein                               !
P 0111 47  04  00    101          OR      @%4,#%0         ! ja, Anzeigestelle dunkel           !
P 0114 E5  04  00    102 ANZ2:    LD      0,@%4           ! Daten auf Anzeigeport              !
P 0117 20  04        103          INC     %4
P 0119 BF            104 DUMMY:   IRET
P 011A E4  12  15    105 SHIFT:   LD      %15,%12         ! Unterprogramm verschiebt den       !
P 011D 56  15  9F    106          AND     %15,#%9F        ! Anzeigepuffer eine Stelle          !
P 0120 46  15  60    107          OR      %15,#%60        ! nach links                         !
P 0123 E4  15  13    108          LD      %13,%15
P 0126 E4  11  15    109          LD      %15,%11
P 0129 56  15  9F    110          AND     %15,#%9F
P 012C 46  15  40    111          OR      %15,#%40
P 012F E4  15  12    112          LD      %12,%15
P 0132 E4  10  15    113          LD      %15,%10
P 0135 56  15  9F    114          AND     %15,#%9F
P 0138 46  15  20    115          OR      %15,#%20
P 013B E4  15  11    116          LD      %11,%15
P 013E 56  16  1F    117          AND     %16,#%1F
P 0141 E4  13  15    118          LD      %15,%13
P 0144 56  15  1F    119          AND     %15,#%1F
P 0147 A6  15  00    120          CP      %15,#0          ! volaufende Null ?                  !
P 014A 6B  06        121          JR      Z,BLANK         ! ja, dunkeltasten                   !
P 014C A6  15  0F    122          CP      %15,#%0F        ! Dunkeltastung gefordert ?          !
P 014F 6B  01        123          JR      Z,BLANK         ! ja, dunkeltasten                   !
P 0151 AF            124          RET
P 0152 46  13  0F    125 BLANK:   OR      %13,#%0F
P 0155 E4  12  15    126          LD      %15,%12
P 0158 56  15  1F    127          AND     %15,#%1F
P 015B A6  15  00    128          CP      %15,#0
P 015E 6B  06        129          JR      Z,BLANK1        ! Dunkeltastung fortsetzen           !
P 0160 A6  15  0F    130          CP      %15,#%0F
P 0163 6B  01        131          JR      Z,BLANK1
P 0165 AF            132          RET
P 0166 46  12  0F    133 BLANK1:  OR      %12,#%0F
P 0169 E4  11  15    134          LD      %15,%11
P 016C 56  15  1F    135          AND     %15,#%1F
P 016F A6  15  00    136          CP      %15,#0
P 0172 6B  06        137          JR      Z,BLANK2
P 0174 A6  15  0F    138          CP      %15,#%0F
P 0177 6B  01        139          JR      Z,BLANK2
P 0179 AF            140          RET
P 017A 46  11  0F    141 BLANK2:  OR      %11,#%0F
P 017D AF            142          RET
P 017E 31  20        143 TAST:    SRP     #%20            ! Tastaturabfrageprogramm            !
P 0180 E6  02  0F    144          LD      2,#%0F          ! P20 bis P23 auf H                  !
P 0183 0C  FF        145          LD      R0,#%FF         ! Zeitkonstante f. Entprellung       !
P 0185 0A  FE        146 LOOP:    DJNZ    R0,LOOP
P 0187 E4  02  09    147          LD      %9,2            ! Portbelegung lesen                 !
P 018A 56  09  F0    148          AND     %9,#%F0
P 018D A6  09  F0    149          CP      %9,#%F0         ! Tastatur frei ?                    !
```

```
P 0190 EB EC       170          JR      NZ,TAST          ! nein, weiter versuchen        !
P 0192 0C 01       171 TAB:     LD      R0,#1            ! Zeichentabelle anlegen        !
P 0194 1C 00       172          LD      R1,#0
P 0196 2C 02       173          LD      R2,#2
P 0198 3C 0C       174          LD      R3,#%0C
P 019A 4C 04       175          LD      R4,#4
P 019C 5C 03       176          LD      R5,#3
P 019E 6C 05       177          LD      R6,#5
P 01A0 7C 0C       178          LD      R7,#%0C
P 01A2 8C 07       179          LD      R8,#%7
P 01A4 9C 06       180          LD      R9,#6
P 01A6 AC 08       181          LD      R10,#8
P 01A8 BC 09       182          LD      R11,#9
P 01AA CC 0A       183          LD      R12,#%0A
P 01AC DC 09       184          LD      R13,#%9
P 01AE EC 0B       185          LD      R14,#%0B
P 01B0 E6 0B 00    186 TASTE:   LD      %0B,#0
P 01B3 E6 0A 01    187          LD      %A,#1            ! 1. Spalte einschalten         !
P 01B6 E4 0A 02    188 TAST0:   LD      2,%0A
P 01B9 70 FD       189          PUSH    RP
P 01BB 31 00       190          SRP     #0
P 01BD FC FF       191          LD      R15,#%FF         ! Zeitschleife f. Entprellung   !
P 01BF FA FE       192 LOOP1:   DJNZ    R15,LOOP1
P 01C1 50 FD       193          POP     RP
P 01C3 E4 02 09    194          LD      %9,2
P 01C6 56 09 F0    195          AND     %9,#%F0          ! Matrix lesen                  !
P 01C9 A6 09 F0    196          CP      %9,#%F0
P 01CC EB 0D       197          JR      NZ,TAST1         ! eine Taste betaetigt          !
P 01CE CF          198          RCF
P 01CF 10 0A       199          RLC     %0A
P 01D1 06 0B 04    200          ADD     %0B,#4           ! naechste Spalte testen        !
P 01D4 A6 0B 10    201          CP      %0B,#%10         ! alle Spalten getestet ?       !
P 01D7 7B DD       202          JR      C,TAST0          ! nein, weiter                  !
P 01D9 8B D5       203          JR      TASTE            ! ja, naechste Eingabe          !
P 01DB E6 0C 00    204 TAST1:   LD      %0C,#0           ! Pufferregister clear          !
P 01DE 76 02 10    205          TM      2,#%10           ! 1. Zeile testen               !
P 01E1 6B 1B       206          JR      Z,ZEIOK          ! Taste in dieser Zeile betaet. !
P 01E3 76 02 20    207          TM      2,#%20           ! 2. Zeile testen               !
P 01E6 E6 0C 01    208          LD      %0C,#1           ! Offsettregister laden         !
P 01E9 6B 13       209          JR      Z,ZEIOK          ! Zeichen gefunden              !
P 01EB 76 02 40    210          TM      2,#%40           ! 3. Zeile testen               !
P 01EE E6 0C 02    211          LD      %0C,#2
P 01F1 6B 0B       212          JR      Z,ZEIOK          ! Zeichen gefunden              !
P 01F3 76 02 80    213          TM      2,#%80           ! 4. Zeile testen               !
P 01F6 6B 02       214          JR      Z,ENTIN          ! ENTER eingegeben              !
P 01F8 8B B6       215          JR      TASTE            ! Abfrage neu beginnen          !
P 01FA E6 16 0C    216 ENTIN:   LD      %16,#%0C
P 01FD AF          217          RET
P 01FE E4 0B 0D    218 ZEIOK:   LD      %0D,%0B
P 0201 04 0C 0D    219          ADD     %0D,%0C          ! Offsettwert addieren          !
P 0204 06 0D 20    220          ADD     %0D,#%20         ! Basisadresse d. Tabelle       !
P 0207 E5 0D 16    221          LD      %16,@%0D         ! Zeichen aus Tabelle lesen     !
P 020A 46 03 50    222          OR      3,#%50           ! INPUT-LED einschalten         !
P 020D AF          223          RET
P 020E 76 03 02    224 TIMEL:   TM      3,#2             ! Testmode ?                    !
P 0211 EB 23       225          JR      NZ,TIMLT
P 0213 31 50       226          SRP     #%50             ! Zeitkonstante L               !
P 0215 0C 01       227          LD      R0,#1
P 0217 1C 01       228 LOOP5:   LD      R1,#1
P 0219 2C FF       229 LOOP4:   LD      R2,#%FF
P 021B 2A FE       230 LOOP3:   DJNZ    R2,LOOP3
P 021D 1A FA       231          DJNZ    R1,LOOP4
P 021F 0A F6       232          DJNZ    R0,LOOP5
P 0221 AF          233          RET
P 0222 76 03 02    234 TIMEH:   TM      3,#2             ! Testmode eongestellt ?        !
P 0225 EB 1E       235          JR      NZ,TIMHT         ! ja                            !
P 0227 31 50       236          SRP     #%50
P 0229 0C 01       237          LD      R0,#1
P 022B 1C 02       238 LOOP8:   LD      R1,#2            ! Zeitkonstante H               !
P 022D 2C FF       239 LOOP7:   LD      R2,#%FF
P 022F 2A FE       240 LOOP6:   DJNZ    R2,LOOP6
P 0231 1A FA       241          DJNZ    R1,LOOP7
P 0233 0A F6       242          DJNZ    R0,LOOP8
P 0235 AF          243          RET
P 0236 31 50       244 TIMLT:   SRP     #%50
P 0238 0C 01       245          LD      R0,#1
P 023A 1C FF       246 LOOP11:  LD      R1,#%FF
P 023C 2C FF       247 LOOP10:  LD      R2,#%FF
P 023E 2A FE       248 LOOP9:   DJNZ    R2,LOOP9
P 0240 1A FA       249          DJNZ    R1,LOOP10
P 0242 0A F6       250          DJNZ    R0,LOOP11
P 0244 AF          251          RET
P 0245 31 50       252 TIMHT:   SRP     #%50
P 0247 0C 02       253          LD      R0,#2
P 0249 1C FF       254 LOOP14:  LD      R1,#%FF
P 024B 2C FF       255 LOOP13:  LD      R2,#%FF
P 024D 2A FE       256 LOOP12:  DJNZ    R2,LOOP12
P 024F 1A FA       257          DJNZ    R1,LOOP13
P 0251 0A F6       258          DJNZ    R0,LOOP14
```

```
P 0253 AF          259          RET
P 0254 31  50      260 TIME1:   SRP      #%50
P 0256 0C  01      261          LD       R0,#1
P 0258 1C  FF      262 TIM2:    LD       R1,#%FF
P 025A 2C  FF      263 TIM1:    LD       R2,#%FF
P 025C 2A  FE      264 TIM0:    DJNZ     R2,TIM0
P 025E 1A  FA      265          DJNZ     R1,TIM1
P 0260 0A  F6      266          DJNZ     R0,TIM2
P 0262 AF          267          RET
                   268
P 0263             269 END KEY
                   270
                   271 END EINGABE
```

und /GA sind durch entsprechende Widerstandsbeschaltung gewährleistet, wenn das Gerät nicht mit dem Auswerterechner verbunden ist. Der Sendemodus kann nur verlassen werden, wenn mit RESET = H neue Daten angefordert werden.

Die Ziffernanzeige wird über ein Anzeigeprogramm gesteuert, das interruptgesteuert aufgerufen wird. Als Interruptquelle dient der Nulldurchgang des Zähler-/Zeitgeberkanals T0. Ausgegeben wird stets der Inhalt des Anzeigepuffers (%10 bis %13). Die Software ist so gestaltet, daß als BCD-zu-7-Segmentdekoder Schaltkreise *D 146/147* oder *D 346/348* gleichermaßen einsetzbar sind. Zur Dunkeltastung einer Anzeigestelle wird H-Pegel an Eingang C des Stellendekoders *DS 8205* ausgegeben. Die Digitausgänge O0 bis O3 sind damit stets inaktiv, d.h., die mit den Eingangspegeln an A und B adressierte Stelle bleibt dunkel. Ebenfalls zeitmultiplex erfolgt die Ausgabe des Dezimalpunkts. Er wird eingeschaltet, wenn bei dem ausgegebenen Zeichen Bit D4 gesetzt ist.

10.3. Ankopplung des übergeordneten Rechners

Hardware

Die Eingabegeräte sind an das *U 880*-System des *KC 85/1* bzw. *KC 87* über einen PIO-Schaltkreis *U 855* angeschlossen. In der Grundvariante ist eine galvanische Trennung nicht vorgesehen. Die Daten- und Steuersignale werden durch *V 4050*-Gatter getrieben. Ein Dekoder *DS 8205* liefert die Geräteanwahlsignale /GA1 bis /GA8. Schutzwiderstände an allen Ein- und Ausgängen sollen wiederum strombegrenzend bei eingekoppelten, energiereichen Störimpulsen wirken. Es gilt folgende Belegung der PIO-Signalleitungen:

```
Port A:      PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0

              D7  D6  D5  D4  D3  D2  D1  D0
-------------------------------------------------------------------

Port B:      PB7 PB6 PB5 PB4 PB3 PB2 PB1 PB0
               I   I   I   I   I---------------- /DR (L-aktiv)
               I   I   I   I-------------------- RESET
               I---I---I---------------------- Geräte-Nr. 0 bis 7
```

Software

Das Hauptprogramm für die Auswertung der Daten ist vom konkreten Anwendungsfall abhängig. In diesem Rahmen wird deshalb nur das Unterprogramm READ (Programm 10.2) zum Empfang eines Datenblocks vorgestellt. Jedem Eingabegerät ist dabei ein Statusbyte zugeordnet, dessen Bitkonfiguration die aktuelle Situation der Programmabarbeitung kennzeichnet. Es hat folgenden Aufbau:

```
D7 D6 D5 D4 D3 D2 D1 D0
I   I  I  I  I  I  I  I----------- 1: Gerät nicht bearbeitet
I   I  I  I  I  I  I               0: Gerät wurde angewählt
I   I  I  I  I  I  I-------------- 1: Vorbytes nicht gefunden
I   I  I  I  I  I                  0: Vorbytes erkannt
I   I  I  I  I  I----------------- 1: Keine Fertigmeldung
I   I  I  I  I                     0: Gerät fertig
I   I  I  I  I-------------------- 1: Paritätsfehler
I   I  I  I                        0: Kein Paritätsfehler
I---I--I--I----------------------  : Eventuell im Hauptprogamm
                                     nutzbar
```

Zunächst wird das Fertigsignal /DR getestet und damit festgestellt, ob die Eingabe beim gerufenen Gerät schon beendet ist. Bei negativem Testergebnis (/DR = H) wird das Programm verlassen. Liegt die Fertigmeldung vor, besteht die nächste Aktivität im Lesen des 1. Vorbytes. Dazu werden 16 aufeinanderfolgende SYN-Zyklen abgetestet. Gelingt die Erkennung nicht, wird der Lesezyklus ebenfalls verlassen. Nach erfolgreichem Lesen des 1. Vorbytes muß das 2. unmittelbar folgen. Tritt dabei ein Fehler auf, wird das Lesen der Daten beim betreffenden Gerät wiederum abgebrochen. Die Vorbytes sind so gewählt, daß ein richtiges Lesen auf die ordnungsgemäße Funktion von Übertragungskabel und zugehöriger Hardware hinweist. Unterbrechungen bzw. Kurzschlüsse lassen dagegen ein fehlerfreies Lesen nicht zu. In Verbindung mit der Paritätsprüfung wird damit eine hohe Übertragungssicherheit gewährleistet. Die Auswertung der Statusbytes im Hauptprogramm erlaubt Rückschlüsse auf die Fehlerursache. Das gewählte Prinzip der Softwaregestaltung unterstützt ein Zusammenspiel mit übergeordneten Programmen, die in einer höheren Programmiersprache geschrieben sind (z. B. BASIC).

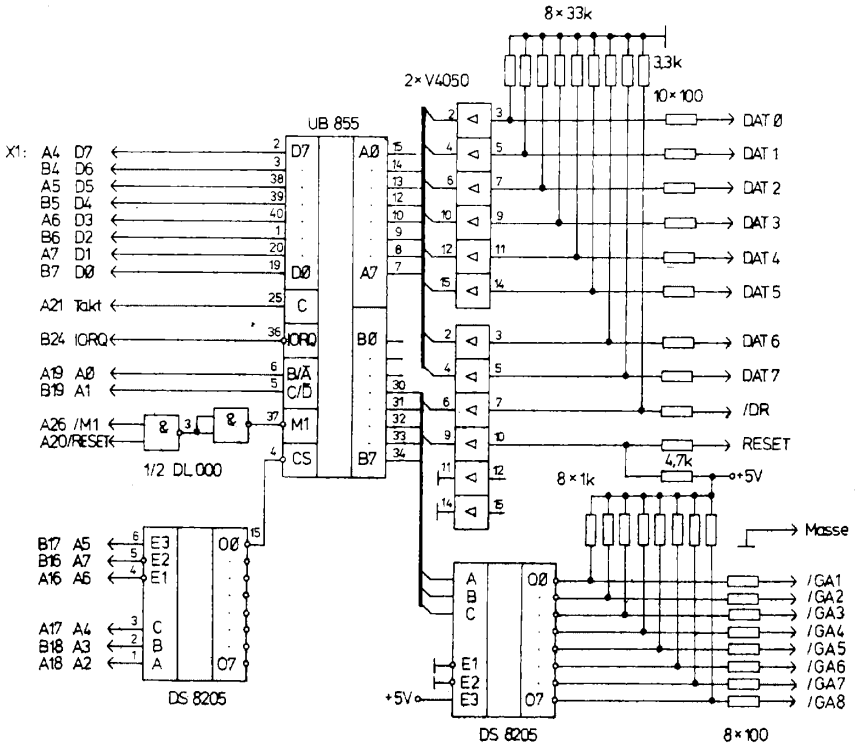


Bild 10.2 Ankopplungsmodul für KC 85/1 bzw. KC 87

Programm 10.2

```
 LOC   OBJ CODE M ZEILE QUELLE

                      1          ; Empfangsprogramm READ fuer Koppelmodul entsprechend Bild 10.2
                      2          ; gelesen werden die Daten eines Geraetes
                      3
                      4
7000                  5          ORG     7000H
                      6  PUANF: EQU     7F00H          ; Speicherbereich fuer Daten aller Geraete
                      7  PADAD: EQU     20H            ; Daten Port A
                      8  PADBD: EQU     PADAD+1        ; Daten Port B
                      9  PADSA: EQU     PADAD+2        ; Steuerwort Port A
                     10  PADSB: EQU     PADAD+3        ; Steuerwort Port B
                     11  PUZEI: EQU     PUANF+80       ; Zeiger auf Datenpuffer
                     12  GERST: EQU     PUZEI+2        ; Zeiger auf Statusbytes
7000  F5             13  LEBLO: PUSH    AF             ; Lesen eines Blockes
7001  C5             14         PUSH    BC
7002  D5             15         PUSH    DE
7003  E5             16         PUSH    HL
7004  2A527F         17         LD      HL,(GERST)     ; Status laden
7007  CB86           18         RES     0,(HL)         ; Bearbeitungsflag setzen
7009  7E             19         LD      A,(HL)
700A  E60F           20         AND     0FH            ; obere 4 Bit = 0
700C  FE00           21         CP      0              ; bereits fertig gelesen ?
700E  2844           22         JR      Z,D6           ; ja, Ende
7010  F5             23         PUSH    AF
7011  C5             24         PUSH    BC
7012  01FF00         25         LD      BC,0FFH        ; Wartezeit
7015  0B             26  W1:    DEC     BC
7016  78             27         LD      A,B
7017  B1             28         OR      C
7018  20FB           29         JR      NZ,W1
701A  C1             30         POP     BC
701B  F1             31         POP     AF
701C  DB21           32         IN      A,(PADBD)      ; Fertigmeldung /DR testen
701E  CB5F           33         BIT     3,A
7020  2032           34         JR      NZ,D6          ; keine Fertigmeldung vom Geraet
7022  CB96           35         RES     2,(HL)         ; Flag fuer Fertigmeldung setzen
7024  0610           36         LD      B,10H          ; 1. Vorbyte suchen
7026  CD5970         37  D1:    CALL    BYTEL          ; Lesen ein Byte
7029  FE2A           38         CP      2AH            ; 1. Vorbyte gelesen ?
702B  2804           39         JR      Z,D2           ; ja, Fortsetzen
702D  10F7           40         DJNZ    D1             ; nein, weitersuchen
702F  1823           41         JR      D6             ; nicht gefunden, Ende
7031  CD5970         42  D2:    CALL    BYTEL
7034  FE3C           43         CP      3CH            ; 2. Vorbyte testen
7036  2802           44         JR      Z,D21          ; gefunden, Fortsetzen
7038  181A           45         JR      D6             ; nicht gefunden, Ende
703A  CB8E           46  D21:   RES     1,(HL)         ; Flag fuer Vorbyteerkennung
703C  0605           47         LD      B,5            ; Zahl der zu lesenden Bytes
703E  ED5B507F       48         LD      DE,(PUZEI)     ; Byte im Anzeigepuffer ablegen
7042  1B             49         DEC     DE             ; rueckwaerts beschreiben
7043  CD5970         50  D3:    CALL    BYTEL          ; Daten lesen
7046  3859           51         JR      C,PAFEH        ; Paritaetsfehler behandeln
7048  FE3F           52  D4:    CP      3FH            ; dunkelgetastete Stelle ?
704A  2002           53         JR      NZ,D5          ; nein, ausgeben
704C  3E20           54         LD      A,20H          ; ja, Leerzeichen eintragen
704E  12             55  D5:    LD      (DE),A         ; im Puffer ablegen
704F  1B             56         DEC     DE             ; naechstes Zeichen
7050  10F1           57         DJNZ    D3             ; alle 5 Zeichen lesen
7052  CB9E           58         RES     3,(HL)         ; Flag setzen in Statusbyte
 LOC   OBJ CODE M ZEILE QUELLE

7054  E1             59  D6:    POP     HL             ; Ende Lesezyklus
7055  D1             60         POP     DE
7056  C1             61         POP     BC
7057  F1             62         POP     AF
7058  C9             63         RET
7059  C5             64  BYTEL: PUSH    BC
705A  D5             65         PUSH    DE
705B  E5             66         PUSH    HL
705C  DB20           67  BY1:   IN      A,(PADAD)      ; Daten lesen
705E  CB7F           68         BIT     7,A            ; SYN = H ?
7060  20FA           69         JR      NZ,BY1         ; ja, warten
7062  DB20           70  BY2:   IN      A,(PADAD)
7064  CB7F           71         BIT     7,A            ; SYN = L ?
7066  28FA           72         JR      Z,BY2          ; ja, warten
7068  0610           73         LD      B,10H
706A  10FE           74  TIM:   DJNZ    TIM            ; Wartezeit
706C  0E20           75         LD      C,PADAD
706E  ED40           76         IN      B,(C)
7070  B8             77         CP      B              ; wenn ungleich: Stoerimpuls
7071  20E9           78         JR      NZ,BY1         ; neu versuchen
7073  2F             79         CPL                    ; Negation der Datentreiber kompensieren
7074  E67F           80         AND     7FH            ; SYN ausblenden
7076  F5             81         PUSH    AF             ; Daten retten
7077  E63F           82         AND     3FH            ; Datenbit isolieren
```

```
7079  57          83          LD      D,A             Paritaetspruefung
707A  B7          84          OR      A
707B  0E00        85          LD      C,0
707D  0606        86          LD      B,6
707F  1F          87  BY3:    RRA
7080  3001        88          JR      NC,BY5
7082  0C          89          INC     C
7083  10FA        90  BY5:    DJNZ    BY3
7085  79          91          LD      A,C
7086  CB47        92          BIT     0,A
7088  7A          93          LD      A,D
7089  2002        94          JR      NZ,BY6
708B  CBF7        95          SET     6,A             ; Paritaetsbit = H
708D  47          96  BY6:    LD      B,A
708E  F1          97          POP     AF              ; gerettete Daten holen
708F  B7          98          OR      A
7090  B8          99          CP      B               ; Gleichheit ?
7091  2807       100          JR      Z,BY7
7093  2A527F     101          LD      HL,(GERST)      ; Statusbyte fuer aktuelles Geraet laden
7096  CBDE       102          SET     3,(HL)          ; Paritaetsfehlerflag setzen
7098  1802       103          JR      BY8             ; verlassen Leseprogramm wegen Paritaetsfehler
709A  E63F       104  BY7:    AND     3FH             ; Datenbit isolieren
709C  E1         105  BY8:    POP     HL
709D  D1         106          POP     DE
709E  D1         107          POP     DE
709F  C1         108          POP     BC
70A0  C9         109          RET
70A1  C9         110  PAFEH:  RET                     ; Auswertung Paritaetsfehler nicht
                 111                                  ; ausgefuehrt
%
```

11. Alphanumerische Anzeigeeinheit mit *VQC 10*, EMR-Steuerung und serieller Kopplung zu einem *U 880*-System

Die alphanumerische Anzeigeeinheit *VQC 10* hat aufgrund ihrer relativ aufwendigen Ansteuerung zur Zeichenaufbereitung bisher wenig Verbreitung gefunden. Dazu hat sicher auch die begrenzte Anzeigekapazität dieser Baugruppe (4 Stellen je *VQC 10*) mit hohem Strombedarf beigetragen. Allerdings wird hier i. allg. der Informationsgehalt einer Nachricht mit der Anzahl der gleichzeitig darstellbaren Zeichen gleichgesetzt. Die *VQC 10* bietet jedoch den Vorteil, durch umlaufende Informationsdarstellung («Laufschrift») auch Nachrichten mit einer längeren Zeichenanzahl darzustellen. Beispielsweise sind visuelle Bedienerinformationen, die auf einer 8stelligen Anzeige (2× *VQC 10*) dargestellt werden und aus einem Satz von 40 Zeichen bestehen, noch problemlos zu verstehen. Damit ist eine Alternative zum oft verwendeten Bildschirm bei relativ geringem Informationsdurchsatz gegeben. Das im folgenden vorgestellte Beispiel soll diese Möglichkeiten erläutern.

Durch Einsatz eines Einchipmikrorechners zur Ansteuerung der eigentlichen Anzeigebaugruppe läßt sich der notwendige Bauelementeaufwand minimieren. Gleichzeitig wird durch den EMR die Kopplungsmöglichkeit zu anderen Rechnern und Geräten über die serielle Schnittstelle bereitgestellt. Die konkrete elektrische Ausführung dieser Schnittstelle richtet sich nach der jeweiligen Rechnergegenstelle (s. Abschnitt 8.); im Beispiel wurde die Schnittstelle nach IFSS-Standard ausgeführt.

Die Gegenstelle der so realisierten Anzeigebaugruppe muß über geeignete Mittel verfügen, den asynchronen seriellen Datentransfer nach *U 881/U 882*-Format zu organisieren. Dafür kommen die in den verschiedenen Rechnertypen vorhandenen Schaltkreise (*U 880-SIO: U 8560; U 881/882*: interne SIO) oder in einfachster Weise eine Eingabeleitung in Frage. Im letzteren Fall muß die Aufbereitung der asynchronen Datenübertragung durch die Software übernommen werden. Im vorgestellten Beispiel wurde eine Ankopplung an ein *U 880*-System über den SIO *U 8560 D* realisiert.

Ein derartiges Anzeigesystem ist beispielsweise als dezentrale Anzeigeeinheit in einer Prozeßüberwachung einsetzbar, bei der einem entfernten Bediener entsprechende Informationen übermittelt werden. Durch den genannten «Laufschrift-Effekt» sind auch Wahlmöglichkeiten in kleinerem Umfang anzeigbar. Ein Dialog läßt sich dann durch entsprechende Tastenbetätigung und Rücksendung über die SIO zum eigentlichen Meßsystem entwickeln. Die IFSS-Schnittstelle gestattet dabei die Übertragung in einem Bereich bis 500 m.

Mögliche Notsituationen lassen sich durch die Verwendung der Laufschrift wesentlich besser aus einem Anzeigesystem herausheben als stehende Wiedergabe mit 7-Segment-Anzeige oder Bildschirm. Durch die Einsparung des Bildschirms einschließlich der zugehörigen Ansteuerung läßt sich in bestimmten Anwendungsfällen ein nicht unerheblicher ökonomischer Nutzen erzielen.

Ein weiterer Anwendungsfall der Anzeigeeinheit ist innerhalb eines UKW-Tuners zur alphanumerischen Stationsanzeige denkbar, wobei vom Einchiprechner noch eine ganze Reihe weiterer Funktionen übernommen werden kann [6]. Bei einer derartigen

Anwendung ist allerdings die vom EMR ausgehende Störstrahlung ein nicht zu unterschätzendes Problem.

11.1. Schaltungstechnik

11.1.1. Anzeigebaugruppe *VQC 10* mit Einchiprechner

Die Schaltung der Anzeigebaugruppe ist in Bild 11.1. dargestellt. Sie wurde weitestgehend aus [7] entnommen. Die Informationsdarstellung erfolgt in der *VQC 10* als 5 × 7-Punktmatrix mit 4 Stellen je Anzeigebauelement. Die Spalteninformation (5 Bit) wird über Tor 2 des EMR ausgegeben. Dazu ist an P23 bis P27 je ein Gatter des Treiberschaltkreises *V 4098* angeschlossen. Die damit verbundene Negation der Spalteninformation ist bei der Gestaltung des Steuerprogramms zu beachten. Soll eine der 5 Spalten einer Anzeigestelle aktiviert werden, muß der entsprechende EMR-Ausgang P23 bis P27 im Moment der Datenübernahme (L-Impuls am cp-Eingang) L-Pegel aufweisen.

Mit den unteren 3 Bit von Tor 2 wird die aktuelle Zeilennummer ausgegeben. Der nachgeschaltete 1-aus-8-Decoder *DS 8205* steuert jeweils einen der 7 Zeilentreiber-Transistoren an. Der angegebene Transistortyp ist bei einer 8stelligen Anzeige noch nicht zwingend erforderlich, es können auch *SF 016* oder ähnliche Typen eingesetzt

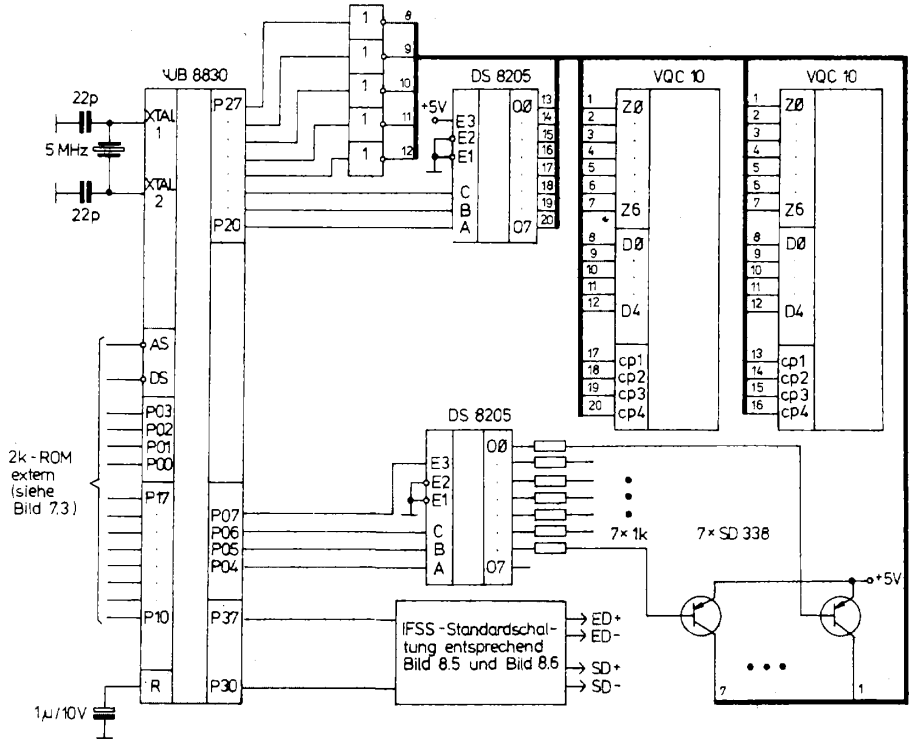


Bild 11.1 Anzeigebaugruppe VQC 10 mit U 883

werden. Beim Ausbau auf noch mehr Anzeigestellen kann dieser Weg jedoch nicht beschritten werden, immerhin ist davon auszugehen, daß im ungünstigsten Falle alle LED einer Zeile leuchten. Die Stromaufnahme beträgt dann bis zu 0,5 A. Durch den Multiplexbetrieb ist der mittlere Strom allerdings wesentlich kleiner, so daß sich die in der Schaltung angegebenen *SD 338* praktisch nicht erwärmen. Für die Transistorauswahl muß dennoch der Spitzenstrom als Kriterium gelten, wenn Bauelementeausfälle sicher vermieden werden sollen.

Ein zweiter *DS 8205* ist für die Stellenauswahl vorgesehen. Die Leitungen P04 bis P06 dienen der Adressierung. Die Aktivierung des Stellendekoders und damit die Datenübernahme in die internen Latches der *VQC 10* erfolgt nur dann, wenn der über P07 ausgegebene Torimpuls H-Pegel aufweist.

Die im Beispiel verwendeten Torleitungen erlauben einen Betrieb des EMR mit externem Speicher, d. h., Tor 1 und die untere Hälfte von Tor 0 sind nicht mit in die Steuerung einbezogen. Eine Erweiterung der Zahl von Anzeigestellen ist durch Kaskadierung mehrerer *DS 8205* als Stellendekoder möglich. Für jeweils 8 Stellen wird dann eine weitere Ausgabeleitung des EMR benötigt.

Die Gestaltung der seriellen Schnittstelle nach IFSS weist keine Besonderheiten auf. Sender und Empfänger werden zur Minimierung des Aufwands an der Anzeigeeinheit im Passiv-Modus betrieben. Auf mögliche andere Auslegungen der Schnittstelle wurde im Abschnitt 8. hingewiesen.

11.1.2. *U 880*-Baugruppe mit SIO und CTC

Um die Kopplung der Anzeigebaugruppe an ein *U 880*-System zu realisieren, wurde die in Bild 11.2 dargestellte Baugruppe genutzt.

Die Dekodierung der E/A-Adressen für die Peripherieschaltkreise erfolgt wie üblich über die Adressen AB0 bis AB7 unter Einbeziehung der Signale /M1, /IORQ und /IODI. Für die im Abschnitt 11.2. beschriebenen Programme sind die Adressen /CSCTC = 70H bis 73H und /CSSIO = 74H bis 77H einzustellen. Gegebenenfalls ist hier eine Modifikation zur Anpassung an andere E/A-Konfigurationen vorzunehmen. Zum Anschluß der Datenleitungen DB0 bis DB7 an den Systembus des *U 880* muß ein entsprechender bidirektionaler Treiberschaltkreis (D1, *DS 8286*) benutzt werden, dessen Richtungssteuerung (Eingang T) in Richtung Systembus bei Lesezugriff zu CTC/SIO oder bei Lesen der Interruptvektoren beider Peripherieschaltkreise umgeschaltet wird. Bei Nutzung von 2 Treiberschaltkreisen *DS 8216* anstelle des *DS 8286* ist der aktive Pegel des Signals zur Richtungssteuerung zu negieren.

Die Interruptkaskade ist über beide interruptfähigen Schaltkreise geführt, im Beispielprogramm werden jedoch nur SIO-Interrupts genutzt. Die CTC-Kanäle 1, 2 und 3 sowie der Kanal A des *U 8560* bleiben frei, sie könnten z. B. zum Aufbau einer weiteren seriellen Schnittstelle dienen. Sollte außerhalb der Baugruppe ein weiterer freier CTC-Kanal vorhanden sein, kann auf den Einsatz des CTC-Schaltkreises ganz verzichtet werden. Der Sende- und Empfangstakt müßte dann dem Eingang RxTxCB des SIO zugeführt werden.

Der CTC-Kanal 0 wird als Zeitgeber ohne Interrupt zur Bereitstellung des Sende- und Empfangstaktes genutzt. Die Stromeinspeisung in Sende- und Empfangsstromschleife erfolgt im Schaltungsbeispiel durch Konstantstromquellen (VT1 bis VT4, VD4

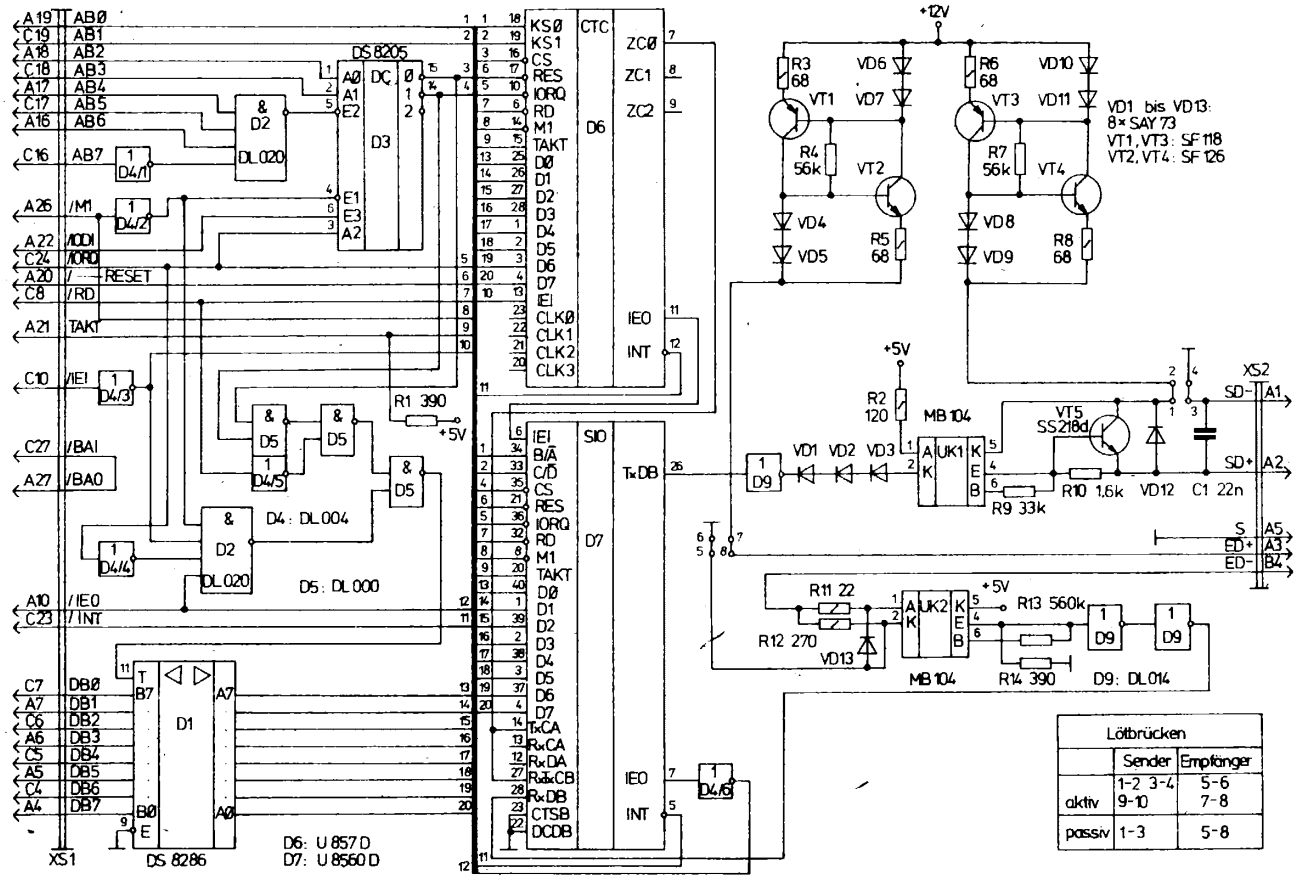


Bild 11.2 U880-Modul mit CTC und SIO

bis VD11, *R3* bis *R8*), um bei großen Übertragungsentfernungen vom konkreten Leitungswiderstand unabhängig zu sein. Die im Bild 11.2 gezeichnete Stellung der Lötbrücken 1 bis 8 entspricht dem Aktiv-Modus in beiden Schleifen. Diese Beschaltung wurde gewählt, um den Schaltungsaufwand an der Anzeigebaugruppe so gering wie möglich zu halten. Sollen die Schleifen im Passiv-Modus arbeiten, so sind anstelle der gezeichneten Lötbrücken die Brücken 1–3 und 5–8 einzusetzen. Die galvanische Trennung der Baugruppen wird über die Optokoppler *MB 104* (UK1, UK2) erreicht. Um diese Trennung aufrechtzuerhalten, darf der Schirm (XS2: A5) der Verbindungsleitung nur an einer der beiden Baugruppen angeschlossen werden. Bei der Verdrahtung der Verbindungsleitungen zur Anzeigebaugruppe ist auf die richtige Kopplung zu achten (SD+ nach ED−, SD− nach ED+).

Sollten die Schaltkreise mit Schmitt-Triggerverhalten (D9), die zur Verbesserung der Schaltflanken des Fototransistors in UK2 eingesetzt werden, nicht zur Verfügung stehen, ist auch die ersatzweise Verwendung normaler Gatter (*DL 004, D 104*) möglich.

11.2. Steuerprogramme

11.2.1. Taktsynchronisation

Für eine Kopplung von Rechnern, die mit unterschiedlichen Taktfrequenzen arbeiten, ist die möglichst ideale Abstimmung der Bitraten der beteiligten seriellen Schaltkreise notwendig. Die im Beispiel genutzten Rechner arbeiten in der asynchronen Datenübertragung mit folgenden Bitraten:

U 881/U 882

$$BR_8 = \frac{f_8}{K8 \cdot VT_8 \cdot ZK_8}$$

mit f_8 – externe Taktfrequenz an XTAL,
VT_8 – Vorteiler T0,
ZK_8 – Zeitkonstante T0,
K8 –128; Konstante.

U 880

$$BR_{80} = \frac{f_{80}}{VT_{80} \cdot ZK_{80} \cdot VT_{SIO}}$$

mit f_{80} – Taktfrequenz *U 880*,
VT_{80} – Vorteiler CTC-Kanal 0,
ZK_{80} – Zeitkonstante CTC0,
VTSIO – Taktratenteiler SIO (Register WR 4).
Aus den Formeln ergibt sich für ideal gleiche Bitraten:

$$\frac{f_8}{f_{80}} = \frac{K_8 \cdot VT_8 \cdot ZK_8}{VT_{80} \cdot ZK_{80} \cdot VT_{SIO}}$$

Im Beispiel wird mit folgenden Daten gearbeitet:

$f_8 = 5$ MHz	$f_{80} = 2{,}45$ MHz
$VT_8 = 1$	$VT_{80} = 16$
$ZK_8 = 49$	$ZK_{80} = 12$
K8 = 128	$VT_{SIO} = 16$.

Damit entstehen die Bitraten:
$BR_8 = 797{,}1$ Baud und $BR_{80} = 797{,}5$ Baud.
Die Bitrate von 797 Bit/s wurde bewußt niedrig gewählt, da im Beispiel nur geringe Datenmengen übertragen werden müssen und die Störanfälligkeit bei niedrigen Übertragungsraten nachweislich geringer ist.

11.2.2. Steuerprogramm der Anzeigebaugruppe

Das Steuerprogramm für die zeitmultiplex arbeitende Anzeige ist im Programm 11.1 aufgelistet. Das Anzeigeprogramm (Zeilen 67 bis 112) wird dabei als zyklisches Hauptprogramm durchlaufen, während neue darzustellende Zeichen über SIO-Interrupt (Zeilen 124 bis 163) in den Anzeigepuffer (Register %10 bis %17) geladen werden.
Die Initialisierung des EMR (Zeilen 42 bis 66) wurde für einen *U 883* vorgenommen, bei anderen Typen müssen gegebenenfalls weitere Befehle zur Torkonfiguration ergänzt werden. Die Programmierung des Zeitgebers T0 ist entsprechend der in Abschnitt 11.2.1. berechneten Baudrate ausgeführt (d. h. VT = 1, ZK = 49). Als serielles Protokoll wird eine Übertragung mit Parität zur Fehlererkennung und 7 Datenbit gewählt. Damit lassen sich die ASCII-Zeichen %00 bis %7F übertragen.

In den Zeilen 56 bis 60 wird der Anzeigepuffer vor der erstmaligen Anzeige gelöscht, um nach RESET eine definierte Anzeige zu erreichen. In den Zeilen 61 bis 63 werden die für den seriellen Empfang notwendigen Bedingungen eingestellt (Empfangsstatus SIOSTA = 0, Empfangszähler SIOZAE = 8, Zeiger SIOZEI auf Zwischenpufferanfang [%18] setzen).

Zu Beginn eines Zyklus des Hauptprogramms wird die erste Zeile der Anzeige beschrieben, d. h., die Bitmuster werden entsprechend den einzelnen Anzeigestellen in aufsteigender Folge geladen. Durch Einschalten der (nicht vorhandenen) 8. Zeile der *VQC 10* bleibt die Anzeige dabei dunkel.

In Zeile 108 wird vor Aufruf einer neuen Zeile das Unterprogramm WAIT eingeschaltet, das 2 Funktionen ausführt. Zunächst besteht es aus einer Warteschleife von etwa 3 ms (Zeilen 115 bis 118). Diese Warteschleife von 21 ms je Zyklus wurde eingeführt, um ein zu häufiges Durchlaufen des Hauptprogramms zu verhindern. Indem die Programmlaufzeit für das Beschreiben aller Anzeigestellen einer Zeile klein gegenüber der restlichen Zeit gehalten wird, kommt eine optimale und gleichmäßige Lichtausbeute aller LED einer Zeile zustande. Die Gesamtwartezeit von 21 ms je Zyklus garantiert dabei ein flimmerfreies Bild der Anzeige. Die 2. Funktion des Unterprogramms WAIT besteht im Einschalten der aktuellen Zeile (Helltastung) am Programmanfang und im Ausschalten der Anzeige nach Ablauf der Wartezeit (Dunkeltastung). Ein Verzicht auf die Dunkeltastung führt dazu, daß zusätzlich zu den tatsächlich angesteuerten LED in der darauffolgenden Zeile ein Punkt mit verminderter Helligkeit leuchtet.

Der ASCII-Kode der darzustellenden Zeichen muß im Anzeigepuffer %10 bis %17 abgelegt sein, wobei der ersten Anzeigestelle das Register %10 zugeordnet ist. Ein Zei-

Programm 11.1

```
Z8ASM      4.0
LOC    OBJ CODE     STMT SOURCE STATEMENT

                      1 VQC10 MODULE
                      2         ! Steuerprogramm fuer EMR U 883 MIT VQC10       !
                      3         ! Version  1.1  vom  14.1.88                    !
                      4
                      5         ! Registerbelegung :
                      6
                      7                 %004 :  ZEILENZAEHLER VQC10
                      8                 %005 :  STELLENZAEHLER VQC10
                      9                 %006 :  ZEIGER AUF ZEICHENPUFFER
                     10                 %007...%009  HILFSZELLEN
                     11                 %0B ,  , %20...%25 HILFSZELLEN
                     12                 %010...%017  ZEICHENPUFFER
                     13                 %018...%01F  ZWISCHENPUFFER BEI SIO-EMPFANG     !
                     14
                     15 CONSTANT
                     16
                     17         STELLE := 8              ! GESAMTZAHL DER ANZEIGESTELLEN (1...8)!
                     18
                     19         TOZEIT := 49             ! ZEITKONSTANTE TIMER 0 !
                     20         ANZPUF := %10            ! BEGINN ANZEIGEPUFFER VQC10            !
                     21         ZWIPUF := %18            ! ZWISCHENPUFFER BEI SIO-EMPFANG        !
                     22         SIOSTA := %0E            ! EMPFANGSSTATUS BEI SIO-EMPFANG        !
                     23         SIOZEI := %0D            ! EMPFANGSZEIGER BEI SIO-EMPFANG        !
                     24         SIOZAE := %0C            ! DATENBYTEZAEHLER BEI SIO-EMPFANG      !
                     25         ZEIZAE := 4              ! ZEILENZAEHLER VQC10                   !
                     26         STEZAE := 5              ! STELLENZAEHLER VQC10                  !
                     27         ANZZEI := 6              ! ANZEIGEZEIGER AUF ZEICHENPUFFER       !
                     28
                     29 GLOBAL
                     30         MAIN PROCEDURE
                     31         ENTRY
P 0800               32 $ABS %800                        ! U 883 - VERSION       !
                     33
P 0800 8D  0912      34 INTTAB: JP      DUMMY            ! INTERRUPTSTARTADRESSEN        !
P 0803 8D  0912      35         JP      DUMMY
P 0806 8D  0912      36         JP      DUMMY
P 0809 8D  08B6      37         JP      SIOIN            ! SERIELLER EMPFANG             !
P 080C 8D  0912      38         JP      DUMMY
P 080F 8D  0912      39         JP      DUMMY
                     40
                     41
P 0812 8F            42 BEGIN:  DI
P 0813 31  F0        43         SRP     #%F0             ! STEUERREGISTER INITIALISIEREN !
P 0815 FC  80        44         LD      R15,#%80         ! STACK AB REGISTER %7F         !
P 0817 8C  37        45         LD      R8,#%(2)00110111! STACK INTERN, P04...P07 AUSGAENGE     !
P 0819 6C  00        46         LD      R6,#%(2)00000000! P2 ALLES AUSGAENGE            !
P 081B 7C  C1        47         LD      R7,#%(2)11000001! UNGERADE PARITAET, SERIELLE E/A!
                     48                                 ! P2 MIT PULL-UP'S           !
P 081D BC  08        49         LD      R11,#%(2)00001000       ! IRQ3 FREI , ABER NOCH DI!
P 081F AC  00        50         LD      R10,#%0
P 0821 5C  05        51         LD      R5,#5            ! T0-VORTEILER = 1 ; MODULU-N   !
P 0823 4C  31        52         LD      R4,#TOZEIT       ! ZEITKONSTANTE T0              !
P 0825 1C  03        53         LD      R1,#%(2)00000011! LADEN UND STARTEN T0          !
                     54
P 0827 31  10        55         SRP     #%10
P 0829 8C  08        56         LD      R8,#STELLE       ! ASCII-ANZEIGEPUFFER LOESCHEN  !
P 082B 9C  10        57         LD      R9,#ANZPUF
P 082D B1  E9        58 LOOP4:  CLR     @R9
P 082F 9E            59         INC     R9
P 0830 8A  FB        60         DJNZ    R8,LOOP4
P 0832 B0  0E        61         CLR     SIOSTA           ! EMPFANGSSTATUS  := 0          !
P 0834 E6  0D  18    62         LD      SIOZEI,#ZWIPUF   ! EMPFANGSZEIGER AUF ANFANG     !
P 0837 E6  0C  08    63         LD      SIOZAE,#STELLE   ! 8 BYTE PRO DATENBLOCK         !
                     64
P 083A 9F            65         EI
P 083B 31  20        66         SRP     #%20
P 083D B0  09        67 LOOP:   CLR     9
P 083F 56  00  0F    68         AND     0,#%0F           ! STELLE 0 EINSTELLEN           !
P 0842 B0  04        69         CLR     ZEIZAE           ! ZEILENZAEHLER RESET           !
P 0844 B0  05        70 MARKE2: CLR     STEZAE           ! STELLENZAEHLER RESET          !
P 0846 E6  02  FF    71         LD      2,#%FF           !  DUNKELTASTUNG ANZEIGE        !
P 0849 E6  06  0F    72         LD      ANZZEI,#ANZPUF-1        ! PUFFERZEIGER AUF ANFANG-1
P 084C 0C  0C        73 MARKE1: LD      R0,#%0C
P 084E 1C  00        74         LD      R1,#0            ! BASISADRESSE ZEICHENGENERATOR !
P 0850 B0  07        75         CLR     7
P 0852 E5  06  08    76         LD      8,@ANZZEI        ! AKTUELLES ZEICHEN AUS PUFFER  !
P 0855 90  08        77         RL      8
P 0857 10  07        78         RLC     7
P 0859 90  08        79         RL      8
P 085B 10  07        80         RLC     7
P 085D 90  08        81         RL      8
P 085F 10  07        82         RLC     7                ! %7 = H-TEIL,  %8 = L-TEIL     !
```

```
P 0861 56  08  F8     83          AND     8,#%F8          ! UNTERE 3 BIT = 000 ?          !
P 0864 04  08  E1     84          ADD     R1,8
P 0867 14  07  E0     85          ADC     R0,7            ! ZEICHENADRESSE BERECHNEN      !
P 086A 44  04  E1     86          OR      R1,ZEIZAE       ! ZEILENNUMMER HINZU            !
P 086D 82  20         87          LDE     R2,@RR0         ! BYTE AUS ZEICHENGENERATOR HOLEN !
P 086F 90  E2         88          RL      R2
P 0871 90  E2         89          RL      R2
P 0873 90  E2         90          RL      R2              ! 3 STELLEN NACH LINKS VERSCHIEBEN !
P 0875 29  09         91          LD      9,R2            ! IN HILFSREGISTER LADEN        !
P 0877 60  09         92          COM     9               ! NEGATION DER TREIBER AUFHEBEN !
P 0879 46  09  07     93          OR      9,#7            ! UNTERE 3 STELLEN = 1          !
P 087C 54  09  02     94          AND     2,9             ! DATEN AUF P2 , UNTERE 3 BIT UNVERAENDERT !
P 087F 46  00  80     95          OR      0,#%(2)10000000 ! SCHREIBIMPULS EIN             !
P 0882 FF             96          NOP
P 0883 FF             97          NOP
P 0884 FF             98          NOP
P 0885 56  00  7F     99          AND     0,#%(2)01111111 ! SCHREIBIMPULLS AUS            !
P 0888 46  02  F8    100          OR      2,#%F8          ! DATENBIT =1                   !
P 088B F0  05        101          SWAP    STEZAE
P 088D E4  05  00    102          LD      0,STEZAE        ! STELLENNUMMER AUF P04...P06   !
P 0890 F0  05        103          SWAP    STEZAE
P 0892 20  05        104          INC     STEZAE
P 0894 20  06        105          INC     ANZZEI          ! NAECHSTES ZEICHEN             ! '
P 0896 A6  05  09    106          CP      STEZAE,#STELLE+1        ! LETZTE STELLE                 !
P 0899 7B  B1        107          JR      C,MARKE1        ! NEIN, FORTSETZEN              !
P 089B D6  08A7      108          CALL    WAIT            ! ZEILE AUSGEBEN UND WARTEN     !
P 089E 20  04        109          INC     ZEIZAE          ! NAECHSTE ZEILE                !
P 08A0 A6  04  07    110          CP      ZEIZAE,#7       ! 7 ZEILEN AUSGEGEBEN ? !
P 08A3 7B  9F        111          JR      C,MARKE2        ! NEIN, FORTSETZEN      !
P 08A5 8B  96        112          JR      LOOP            ! NEUER ANZEIGEZYKLUS           !
                     113
P 08A7 E4  04  02    114 WAIT:    LD      2,4             ! AKTUELLE ZEILE EINSCHALTEN    !
P 08AA 3C  50        115          LD      R3,#%50         ! WARTEZEITEN                   !
P 08AC 4C  03        116 WAIT1:   LD      R4,#3
P 08AE 4A  FE        117 WAIT2:   DJNZ    R4,WAIT2
P 08B0 3A  FA        118          DJNZ    R3,WAIT1
P 08B2 E6  02  FF    119          LD      2,#%FF          ! DUNKELTASTUNG ANZEIGE         !
P 08B5 AF            120          RET
                     121
                     122 ! INTERRUPTBEHANDLUNG SIO-EMPFANG       !
                     123
P 08B6 E4  F0  0B    124 SIOIN:   LD      %0B,SIO         ! BYTE LESEN UND ZWISCHENSPEICHERN      !
P 08B9 76  0B  80    125          TM      %0B,#%80        ! PARITAETSFEHLER ?             !
P 08BC 6B  0D        126          JR      Z,SIOIN2        ! NEIN                          !
P 08BE E6  0E  00    127 SIOERR:  LD      SIOSTA,#0       !STATUS RUECKSETZEN             !
P 08C1 E6  0D  18    128          LD      SIOZEI,#ZWIPUF  ! PUFFERZEIGER RUECKSETZEN      !
P 08C4 E6  0C  08    129          LD      SIOZAE,#STELLE  ! DATENBYTEZAEHLER RUECKSETZEN  !
P 08C7 E6  F0  46    130          LD      SIO,#%46        ! "F" SENDEN FUER "FEHLER"      !
P 08CA BF            131          IRET
                     132
P 08CB A6  0E  00    133 SIOIN2:  CP      SIOSTA,#0       ! STARTBYTE ERWARTET ?          !
P 08CE EB  09        134          JR      NZ,SIOIN4       ! NEIN                          !
P 08D0 A6  0B  1E    135          CP      %0B,#%1E        ! STARTBYTE GELESEN ?           !
P 08D3 EB  03        136          JR      NZ,SIOIN5       ! NEIN                          !
P 08D5 E6  0E  01    137          LD      SIOSTA,#1
P 08D8 BF            138 SIOIN5:  IRET
                     139
P 08D9 A6  0E  01    140 SIOIN4:  CP      SIOSTA,#1       ! DATENBYTES ERWARTET ?         !
P 08DC EB  0D        141          JR      NZ,SIOIN6       ! NEIN                          !
P 08DE F5  0B  0D    142          LD      @SIOZEI,%0B     ! DATENBYTE IN ZWISCHENPUFFER   !
P 08E1 20  0D        143          INC     SIOZEI
P 08E3 00  0C        144          DEC     SIOZAE          ! ZAEHLER - 1                   !
P 08E5 EB  03        145          JR      NZ,SIOIN7       ! NOCH NICHT NULL               !
P 08E7 E6  0E  02    146          LD      SIOSTA,#2
P 08EA BF            147 SIOIN7:  IRET
                     148
P 08EB A6  0B  1F    149 SIOIN6:  CP      %0B,#%1F        ! ENDEBYTE ERWARTET             !
P 08EE EB  CE        150          JR      NZ,SIOERR       ! FEHLER                        !
P 08F0 E6  0C  08    151          LD      SIOZAE,#STELLE  ! 8 BYTE UMLAGERN               !
P 08F3 E6  0D  10    152          LD      SIOZEI,#ANZPUF  ! ZIEL: ANZEIGEPUFFER           !
P 08F6 E6  0E  18    153          LD      SIOSTA,#ZWIPUF  ! QUELLE : ZWISCHWNPUFFER       !
P 08F9 E5  0E  0B    154 SIOIN9:  LD      %0B,@SIOSTA
P 08FC F5  0B  0D    155          LD      @SIOZEI,%0B
P 08FF 20  0D        156          INC     SIOZEI
P 0901 20  0E        157          INC     SIOSTA
P 0903 00  0C        158          DEC     SIOZAE
P 0905 EB  F2        159          JR      NZ,SIOIN9       ! WEITER UMLAGERN               !
P 0907 B0  0E        160          CLR     SIOSTA          ! SIOSTATUS RUECKSETZEN         !
P 0909 E6  0D  18    161          LD      SIOZEI,#ZWIPUF  ! ZEIGER RUECKSETZEN            !
P 090C E6  0C  08    162          LD      SIOZAE,#STELLE  ! ZAEHLER RUECKSETZEN           !
P 090F E6  F0  4F    163          LD      SIO,#%4F        ! "O" SENDEN FUER O.K.          !
P 0912 BF            164 DUMMY:   IRET
                     165
P 0913               166          END MAIN
                     167
                     168 END VQC10
                     169
```

chen besteht aus 7 Zeilen mit je 5 Pixel. Die zugehörigen Bitmuster sind im Zeichengenerator abgelegt, der sich im Beispiel ab Adresse %0C00 im EPROM befindet und im Programm 11.2 als Hexdump wiedergegeben ist. Jeweils 8 aufeinanderfolgende Bytes sind in diesem Zeichengenerator einem Zeichen zugeordnet, wobei in den Bytes die Bits D4 bis D0 genutzt werden. Das 1. Byte stellt die 1. Zeile des Zeichens dar, das 2. Byte die 2. Zeile, usw. Eine leuchtende LED ist im zugeordneten Bit (D4 entspricht linker Spalte, D0 rechter Spalte) mit 1 zu kodieren. Das 8. Byte eines Zeichens wird nicht genutzt.

Die ersten 8 Byte des Zeichengenerators entsprechen dem ASCII-Kode %00, die zweiten 8 Byte dem Kode %01 usw. Da für den gesamten Zeichengenerator nur 1 KByte zur Verfügung steht, sind nur die ASCII-Kodes von %00 bis %7F nutzbar. Dieser Zeichenkode wird auch durch das gewählte serielle Datenübertragungsformat mit 7 Bit und ungerader Parität erforderlich. Im Zeichengenerator nach Programm 11.2 sind Groß- und Kleinbuchstaben sowie Sonderzeichen nach internationalem Zeichensatz (also ohne deutsche Umlaute) enthalten. Die Zuordnung von ASCII-Kode und darstellbaren Zeichen zeigt Bild 11.3. Die nicht benutzten ASCII-Kodes %00 bis %1F können bei Bedarf mit eigenen Symbolen belegt werden. Sollen auch Zeichen mit gesetztem Bit 7 (%80 bis %FF) übertragen werden (z. B. Pseudografik), muß der Zeichengenerator 2 KByte umfassen und die serielle Übertragung mit 8 Bit ohne Parität erfolgen. Ein derartiger Ausbau wird allerdings nur sehr selten sinnvoll sein, zumal die deutlich voneinander abgesetzten Anzeigestellen der *VQC 10* eine zusammenhängende Grafikausgabe nicht zulassen.

Das Einschreiben neuer Daten in den Anzeigepuffer wird über Interruptsteuerung bei seriellem Empfang ausgelöst (Zeilen 124 bis 163 im Programm 11.1). Es wird eine asynchrone Übertragung mit ungerader Parität durchgeführt, wobei ein Sende- und Empfangsrahmen aus einem Startbyte (%1E), 8 Datenbytes und einem Stopbyte (%1F) besteht. Erst nachdem der gesamte Rahmen fehlerfrei empfangen wurde, werden die 8 Datenbytes aus dem Zwischenpuffer (%18 bis %1F) in den eigentlichen Anzeigepuffer umgelagert. Zur Kontrolle des Empfangsrahmens wird ein Statusbyte verwaltet (SIOSTA), das zu Beginn des EMR-Steuerprogramms (Zeile 61) rückgesetzt wurde.

Bei Auftreten eines SIO-Empfänger-Interrupts wird zunächst eine Paritätsprüfung (Zeilen 124 bis 126) durchgeführt. Bei einem erkannten Paritätsfehler werden Empfängerstatus, Empfangszeiger und -zähler rückgesetzt und über den seriellen Ausgang die Fehlermitteilung %46 an den übergeordneten Rechner abgesetzt (Zeilen 127 bis

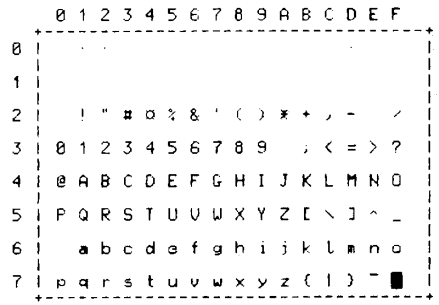


Bild 11.3 ASCII-Zeichensatz für VQC 10

Programm 11.2

```
     1.Record der Datei BUCH.11.2 vom 880127
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0C00 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0C10 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0C20 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0C30 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0C40 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0C50 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0C60 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0C70 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0C80 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0C90 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0CA0 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0CB0 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0CC0 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0CD0 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0CE0 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*
0CF0 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *................*

     2.Record der Datei BUCH.11.2 vom 880127
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0D00 * 00 00 00 00 00 00 00 00 04 04 04 04 04 00 04 00 *................*
0D10 * 0A 0A 0A 00 00 00 00 00 0A 0A 1F 0A 1F 0A 0A 00 *................*
0D20 * 00 11 0E 0A 0E 11 00 00 18 19 02 04 08 13 03 00 *................*
0D30 * 04 0A 0A 0C 15 12 0D 00 04 04 08 00 00 00 00 00 *................*
0D40 * 02 04 08 08 08 04 02 00 08 04 02 02 02 04 08 00 *................*
0D50 * 00 15 0E 1F 0E 15 00 00 00 04 04 1F 04 04 00 00 *................*
0D60 * 00 00 00 0C 0C 08 10 00 00 00 00 1F 00 00 00 00 *................*
0D70 * 00 00 00 00 00 18 18 00 00 01 02 04 08 10 00 00 *................*
0D80 * 0E 11 13 15 19 11 0E 00 04 0C 04 04 04 04 0E 00 *................*
0D90 * 0E 11 01 06 08 10 1F 00 1F 01 02 06 01 11 0E 00 *................*
0DA0 * 02 06 0A 12 1F 02 02 00 1F 10 1E 01 01 11 0E 00 *................*
0DB0 * 07 08 10 1E 11 11 0E 00 1F 01 02 04 08 08 08 00 *................*
0DC0 * 0E 11 11 0E 11 11 0E 00 0E 11 11 0F 01 02 1C 00 *................*
0DD0 * 00 0C 0C 00 0C 0C 00 00 00 00 04 00 04 04 08 00 *................*
0DE0 * 02 04 08 10 08 04 02 00 00 00 1F 00 1F 00 00 00 *................*
0DF0 * 08 04 02 01 02 04 08 00 0E 11 01 02 04 00 04 00 *................*

     3.Record der Datei BUCH.11.2 vom 880127
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0E00 * 0E 11 13 15 17 10 0E 00 04 0A 11 11 1F 11 11 00 *................*
0E10 * 1E 11 11 1E 11 11 1E 00 0E 11 10 10 10 11 0E 00 *................*
0E20 * 1E 09 09 09 09 09 1E 00 1F 10 10 1E 10 10 1F 00 *................*
0E30 * 1F 10 10 1E 10 10 10 00 0E 10 10 17 11 11 0E 00 *................*
0E40 * 11 11 11 1F 11 11 11 00 0E 04 04 04 04 04 0E 00 *................*
0E50 * 01 01 01 01 11 11 0E 00 11 12 14 18 14 12 11 00 *................*
0E60 * 10 10 10 10 10 10 1F 00 11 1B 15 15 11 11 11 00 *................*
0E70 * 11 11 19 15 13 11 11 00 0E 11 11 11 11 11 0E 00 *................*
0E80 * 1E 11 11 1E 10 10 10 00 0E 11 11 11 15 12 0D 00 *................*
0E90 * 1E 11 11 1E 14 12 11 00 0E 11 10 0E 01 11 0E 00 *................*
0EA0 * 1F 04 04 04 04 04 04 00 11 11 11 11 11 11 0E 00 *................*
0EB0 * 11 11 11 11 11 0A 04 00 11 11 11 11 15 15 0A 00 *................*
0EC0 * 11 11 0A 04 0A 11 11 00 11 11 11 0A 04 04 04 00 *................*
0ED0 * 1F 01 02 04 08 10 1F 00 1C 10 10 10 10 10 1C 00 *................*
0EE0 * 00 10 08 04 02 01 00 00 07 01 01 01 01 01 07 00 *................*
0EF0 * 04 0A 00 00 00 00 00 00 00 00 00 00 00 00 1F 00 *................*

     4.Record der Datei BUCH.11.2 vom 880127
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0F00 * 04 04 02 00 00 00 00 00 00 00 0E 12 12 12 0F 00 *................*
0F10 * 10 10 1C 12 12 12 1C 00 00 00 0C 12 10 12 0C 00 *................*
0F20 * 02 02 0E 12 12 12 0F 00 00 00 0E 11 10 10 0E 00 *................*
0F30 * 02 04 04 0E 04 04 04 00 00 01 07 09 07 01 06 00 *................*
0F40 * 10 10 14 1A 12 12 12 00 00 04 00 04 04 04 04 00 *................*
0F50 * 00 04 00 04 04 14 08 00 10 10 12 14 18 14 12 00 *................*
0F60 * 04 04 04 04 04 04 02 00 00 00 0A 15 15 15 15 00 *................*
0F70 * 00 00 14 1A 12 12 12 00 00 00 0C 12 12 12 0C 00 *................*
0F80 * 00 00 1C 12 12 0C 10 00 00 00 0E 12 12 0E 02 00 *................*
0F90 * 00 00 16 18 10 10 10 00 00 00 0E 10 0C 02 1C 00 *................*
0FA0 * 04 04 04 0E 04 04 02 00 00 00 12 12 12 12 0C 00 *................*
0FB0 * 00 00 11 11 11 0A 04 00 00 00 11 11 15 15 0A 00 *................*
0FC0 * 00 00 11 0A 04 0A 11 00 00 00 11 11 0A 04 04 00 *................*
0FD0 * 00 00 1F 02 04 08 1F 00 04 08 08 10 08 08 04 00 *................*
0FE0 * 04 04 04 04 04 04 04 00 04 02 02 01 02 02 04 00 *................*
0FF0 * 1F 00 00 00 00 00 00 00 1F 1F 1F 1F 1F 1F 1F 00 *................*

%
```

131). Bei Eintreffen des Startbytes %1E wird der Empfängerstatus weitergeschaltet (SIOSTA := 1; Zeilen 133 bis 138).

Die nachfolgenden Datenbytes werden im Zwischenpuffer abgelegt. Nachdem alle 8 Datenbytes empfangen wurden, wird der Empfängerstatus weitergeschaltet (SIOSTA := 2; Zeilen 140 bis 145). Nach Erkennen des Stopbytes %1F werden alle empfangenen Datenbytes in den Anzeigepuffer eingetragen (Zeilen 149 bis 164), der Empfängerstatus, der Empfängerzeiger und der Empfängerzähler rückgesetzt und die fehlerfreie Übernahme durch Rücksenden von %4F an den übergeordneten Rechner gemeldet.

11.2.3. *U 880*-Steuerprogramm

Das Steuerprogramm für das Senden des Datenprotokolls entsprechend Abschnitt 11.2.2. durch einen SIO *U 8560* an die *VQC 10*-Anzeigeeinheit ist im Programm 11.3 wiedergegeben.

Das Programm ist als Unterprogramm (Start ab Marke BEGINN) ausgeführt, das einmalig einen Datenblock von 8 Bytes sowie ein Startbyte (1EH) und ein Stopbyte (1FH) sendet. Die zu sendenden Daten müssen vorher im RAM-Bereich ab PUFFER (Zeile 166) abgelegt werden. Das Ergebnis der Datenübertragung wird im Byte QUITT zurückgemeldet.

Die getroffenen Vereinbarungen sind in den Zeilen 11 bis 19 aufgelistet, die Übertragungsrate wird entsprechend Abschnitt 11.2.1. zu 800 Baud eingestellt (CTC-Vorteiler = 16, CTC-Zeitkonstante = 12, SIO-Taktrate × 16). Die Interruptstartadressentabelle liegt im Bereich 0DxxH (I = 0DH), der Interruptvektor des SIO wird mit 50H gewählt.

Zu Beginn des Hauptprogramms werden der CTC-Kanal 0 als Zeitgeber ohne Interrupt entsprechend o.g. Daten initialisiert sowie die Interruptstartadressen eingetragen (Zeilen 28 bis 32, 68 bis 75). Danach wird der SIO-Kanal B zur Duplexübertragung im asynchronen Format mit 2 Stop-Bits und ungerader Parität sowie 7 Zeichen/Bits initialisiert (Zeilen 33, 77 bis 98). Als Interruptquellen werden zugelassen:
Sender- und Empfängerinterrupt, wobei die empfangene Paritätsinformation den Vektor beeinflußt.
Im Anschluß an die SIO-Initialisierung werden Sendezähler (ANZAHL), Sendezeiger (PUFZEI), Senderstatus (MERKER) und Empfängerstatus (QUITT) eingestellt.

Die Datenblockübertragung wird im Unterprogramm durch das Senden des Startbytes 1EH eingeleitet (Zeilen 41 bis 46). Danach verharrt das Unterprogramm so lange in einer Warteschleife, bis der Senderstatus (MERKER) das beendete Senden des gesamten Datenblocks meldet. Das Senden der Datenbytes geschieht interruptgesteuert, indem bei der Bedingung «Sender leer» über die Routine ISEND (Zeilen 99 bis 115) weitere Bytes gesendet werden. Falls alle 8 Datenbytes gesendet wurden, wird die Interruptroutine ISEN1F aktiviert (Zeilen 110, 111). Beim nachfolgenden Senderinterrupt wird das Stopbyte 1FH gesendet (Zeilen 117 bis 123) und die Abschlußroutine ISEEND aktiviert, die dann den SIO-Kanal-B-Sender sperrt und den Senderstatus (MERKER := 1) weiterschaltet (Zeilen 125 bis 142).

Vom Programm, das sich in der Warteschleife befindet, wird der veränderte Sendestatus erkannt und in einer weiteren Warteschleife (Zeilen 52 bis 60) auf das Eintreffen einer Rückmeldung vom EMR (QUITT) gewartet. Das Fehlen dieser Meldung wird

Programm 11.3

```
 LOC   OBJ CODE M ZEILE QUELLE

                      1
                      2           ; Bedienprogramm fuer VQC10 mit U 883
                      3           ; Kopplung ueber IFSS
                      4           ; U 8560 - SIO aktiv ,  U 883 - SIO passiv
                      5
                      6
                      7
                      8           GLOBAL   BEGINN,PUFFER
                      9
                     10
                     11  SIOCTR: EQU      77H       ; U 8560 KANAL B  STEUERWORTADRESSE
                     12  SIODAT: EQU      75H       ; U 8560 KANAL B  DATENWORTADRESSE
                     13  CTCKA0: EQU      70H       ; E/A-ADRESSE CTC-KANAL 0
                     14  CTCZK:  EQU      12        ; CTC-KANAL0 ZEITKONSTANTE
                     15                            ; mit VT=16 UND T x 16 IN SIO :=  800 Boud
                     16  INTTAB: EQU      0D60H     ; INTERRUPTSTARTADRESSENBEREICH SIO
                     17
                     18
C000                 19          ORG      0C000H
                     20
C000   1808          21  BEGINN: JR       START
                     22
                     23                                ; SIO-KANAL B
C002   60C0          24  SIOTAB: DEFW     ISEND         ; SENDEPUFFER LEER
C004   D3C0          25          DEFW     IEXT          ; EXTERN/STATUS-INTERRUPT
C006   C9C0          26          DEFW     IEMPF         ; ZEICHEN EMPFANGEN
C008   D6C0          27          DEFW     ISPEZ         ; BESONDERE EMPFANGSBEDINGUNG
C00A   CD5CC0        28  START:  CALL     CTCINI  ; CTC-INITIALISIERUNG
C00D   2102C0        29          LD       HL,SIOTAB     ; INTERRUPTSTARTADRESSEN
C010   11500D        30          LD       DE,INTTAB     ; EINTRAGEN
C013   010800        31          LD       BC,8
C016   EDB0          32          LDIR                   ; UMLAGERN
C018   CD69C0        33          CALL     SIOINI        ; Z80-SIO INITIALISIEREN
C01B   3E08          34          LD       A,8
C01D   32E4C0        35          LD       (ANZAHL),A    ; 8 DATENBYTES SENDEN
C020   21E7C0        36          LD       HL,PUFFER
C023   22E5C0        37          LD       (PUFZEI),HL   ; PUFFERZEIGER AUF ANFANG STELLEN
C026   3E00          38          LD       A,0
C028   32E2C0        39          LD       (MERKER),A    ; SENDEMERKER LOESCHEN
C02B   32E3C0        40          LD       (QUITT),A     ; EMPFANGSQUITTUNG LOESCHEN
C02E   3E1E          41          LD       A,1EH
C030   D375          42          OUT      (SIODAT),A    ; STARTBYTE SENDEN
C032   3E05          43          LD       A,5
C034   D377          44          OUT      (SIOCTR),A    ; WR5 ADRESSIEREN
C036   3E28          45          LD       A,28H         ; SIO-SENDER FREI
C038   D377          46          OUT      (SIOCTR),A
                     47
C03A   3AE2C0        48  MARKE1: LD       A,(MERKER)
C03D   FE00          49          CP       0             ; SENDEMERKER NOCH NULL ?
C03F   28F9          50          JR       Z,MARKE1      ; JA
                     51
C041   010000        52          LD       BC,0          ; WARTESCHLEIFE AUF EMPFANGSQUITTUNG
C044   3AE3C0        53  MARKE3: LD       A,(QUITT)
C047   FE00          54          CP       0
C049   200A          55          JR       NZ,MARKE2     ; EMPFANGSQUITTUNG ERHALTEN
C04B   10F7          56          DJNZ     MARKE3
C04D   0D            57          DEC      C
C04E   20F4          58          JR       NZ,MARKE3
 LOC   OBJ CODE M ZEILE QUELLE

C050   3E48          59          LD       A,48H         ; EMPFANGSQUITTUNG FEHLT , FC = 48H
C052   32E3C0        60          LD       (QUITT),A
                     61
C055   F3            62  MARKE2: DI
C056   3E18          63          LD       A,18H
C058   D377          64          OUT      (SIOCTR),A    ; SIO-KANAL B RUECKSETZEN
C05A   FB            65          EI
C05B   C9            66          RET
                     67
C05C   F3            68  CTCINI: DI                    ; INITIALISIERUNG CTC-KANAL 0
C05D   0E70          69          LD       C,CTCKA0
C05F   3E07          70          LD       A,7           ; ZEITGEBER,VT=16,OHNE INTERRUPT
C061   ED79          71          OUT      (C),A
C063   3E0C          72          LD       A,CTCZK
C065   ED79          73          OUT      (C),A         ; ZEITKONSTANTE CTC
C067   FB            74          EI
C068   C9            75          RET
                     76
C069   F3            77  SIOINI: DI                    ; SIO-INITIALISIERUNG KANAL B
C06A   0E77          78          LD       C,SIOCTR
C06C   060B          79          LD       B,11
C06E   2175C0        80          LD       HL,SIOITA
C071   EDB3          81          OTIR
C073   FB            82          EI
C074   C9            83          RET
```

```
C075  18          84  SIOITA: DEFB   18H             ; KANAL-RESET
C076  02          85          DEFB   2               ; WR 2
C077  50          86          DEFB   50H             ; INTERRUPTVEKTOR
C078  14          87          DEFB   14H             ; RESET EXTERN/STATUS-INT., WR 4
C079  4D          88          DEFB   4DH             ; TAKT x 16 , 2 STOPP-BIT , UNGERADE PAR.
C07A  03          89          DEFB   3               ; WR 3
C07B  41          90          DEFB   41H             ; 7 BIT/ZEICHEN , EMPFAENGER FREI
C07C  05          91          DEFB   5               ; WR 5
C07D  20          92          DEFB   20H             ; 7 BIT/ZEICHEN , SENDER ZU
C07E  11          93          DEFB   11H             ; RESET EXTERN/STATUS-INT., WR 1
C07F  16          94          DEFB   16H             ; INTERRUPT BEI JEDEM ZEICHEN
                  95                                 ; PARITAET BEEINFLUSST VEKTOR
                  96                                 ; STATUSABHAENGIGER INTERRUPTVEKTOR
                  97                                 ; SENDER-INTERRUPT FREI
                  98
C080  F5          99  ISEND:  PUSH   AF
C081  E5         100          PUSH   HL
C082  2AE5C0     101          LD     HL,(PUFZEI)
C085  7E         102          LD     A,(HL)          ; GEZEIGERTES DATENBYTE AUSGEBEN
C086  D375       103          OUT    (SIODAT),A
C088  23         104          INC    HL              ; ZEIGER + 1
C089  22E5C0     105          LD     (PUFZEI),HL
C08C  3AE4C0     106          LD     A,(ANZAHL)
C08F  3D         107          DEC    A               ; DATENBYTEANZAHL - 1
C090  32E4C0     108          LD     (ANZAHL),A
C093  2006       109          JR     NZ,ISEND1       ; NOCH NICHT ALLE BYTES GESENDET
C095  21A0C0     110          LD     HL,ISEN1F
C098  22500D     111          LD     (INTTAB),HL     ; NEUE INTERRUPTROUTINE SENDER LEER
C09B  E1         112  ISEND1: POP    HL
C09C  F1         113          POP    AF
C09D  FB         114          EI
C09E  ED4D       115          RETI
 LOC  OBJ CODE M ZEILE QUELLE

C0A0  F5         117  ISEN1F: PUSH   AF              ; INTERRUPTBEHANDLUNG STOPPBYTE SENDEN
C0A1  E5         118          PUSH   HL
C0A2  3E1F       119          LD     A,1FH
C0A4  D376       120          OUT    (SIODAT),A      ; 1FH ALS STOPPBYTE SENDEN
C0A6  21AEC0     121          LD     HL,ISEEND       ; NEUE INTERRUPTROUTINE SENDER LEER
C0A9  22500D     122          LD     (INTTAB),HL
C0AC  18ED       123          JR     ISEND1
                 124
C0AE  F5         125  ISEEND: PUSH   AF
C0AF  C5         126          PUSH   BC
C0B0  E5         127          PUSH   HL
C0B1  0E77       128          LD     C,SIOCTR
C0B3  0604       129          LD     B,4
C0B5  21C5C0     130          LD     HL,ENDTAB               ; SIO-SENDER SPERREN
C0B8  EDB3       131          OTIR
C0BA  3E01       132          LD     A,1
C0BC  32E2C0     133          LD     (MERKER),A              ; SENDEMERKER SETZEN
C0BF  E1         134          POP    HL
C0C0  C1         135          POP    BC
C0C1  F1         136          POP    AF
C0C2  FB         137          EI
C0C3  ED4D       138          RETI
C0C5  05         139  ENDTAB: DEFB   5                       ; WR 5
C0C6  20         140          DEFB   20H                     ; SENDER ZU
C0C7  01         141          DEFB   1                       ; WR 1
C0C8  14         142          DEFB   14H                     ; NUR NOCH EMPFAENGER INTERRUPT FREI
                 143
C0C9  F5         144  IEMPF:  PUSH   AF                      ; INTERRUPTBEHANDLUNG ZEICHEN EMPF.
C0CA  DB75       145          IN     A,(SIODAT)
C0CC  32E3C0     146          LD     (QUITT),A               ; GELESENES BYTE ABSPEICHERN
C0CF  F1         147          POP    AF
C0D0  FB         148          EI
C0D1  ED4D       149          RETI
                 150
C0D3  FB         151  IEXT:   EI
C0D4  ED4D       152          RETI
                 153
C0D6  F5         154  ISPEZ:  PUSH   AF              ; INTERRUPT BEI FEHLERHAFTEM EMPFANG
C0D7  DB75       155          IN     A,(SIODAT)
C0D9  3E47       156          LD     A,47H           ; FC=47H FUER GESTOERTEN EMPFANG
C0DB  32E3C0     157          LD     (QUITT),A
C0DE  F1         158          POP    AF
C0DF  FB         159          EI
C0E0  ED4D       160          RETI
                 161
C0E2             162  MERKER: DEFS   1               ; SENDEMERKER
C0E3             163  QUITT:  DEFS   1               ; EMPFANGSQUITTUNG
C0E4             164  ANZAHL: DEFS   1               ; ANZAHL DER ZU SENDENDEN BYTES
C0E5             165  PUFZEI: DEFS   2               ; SENDEPUFFERZEIGER
C0E7             166  PUFFER: DEFS   8               ; 8 BYTE SENDEPUFFER
                 167
                 168  END
                 169
```

nach Ablauf der Wartezeit durch QUITT = 48H angezeigt. Bei eingegangener Quittung wird der gesamte SIO-Kanal B rückgesetzt und die Datenübertragung beendet (Zeilen 62 bis 66).

Die Empfängerinterruptroutinen IEMPF (Zeilen 144 bis 149) und ISPEZ (Zeilen 154 bis 160) melden die von der Anzeigeeinheit eingegangene Information (QUITT = 4FH: Übertragung i. O.; QUITT = 46H: Fehler beim Senden; QUITT = 47H: Fehler im Quittungssignal).

11.2.4. Erzeugung von Laufschrift

Zur Erzeugung einer Laufschrift in der Anzeigeeinheit *VQC 10* mit Einchipmikrorechner gibt es 2 prinzipielle Möglichkeiten. Erstens kann das EMR-Steuerprogramm so gestaltet werden, daß der Inhalt des Anzeigepuffers durch den EMR selbst umlaufend verschoben wird. Diese Variante setzt voraus, daß für eine stehende und laufende Anzeige 2 verschiedene Programmteile benutzt werden müssen. Die Übertragung der darzustellenden Zeichen müßte nur einmal zu Beginn erfolgen. Die 2. Möglichkeit der Laufschriftanzeige besteht darin, das EMR-Steuerprogramm aus Abschnitt 11.2.2. unverändert zu belassen und die Laufschrift vom übergeordneten Rechner aus durch fortwährende Übertragungen von 8 verschobenen Datenbytes auszulösen. Diese Variante bietet den Vorteil, die Laufschrift zu beliebiger Zeit stoppen und fortsetzen zu können. Außerdem kann problemlos eine längere Information dargestellt werden. Für die EMR-Steuerung ist nur ein Programm erforderlich. Das Beispiel im Programm 11.4 benutzt die 2. Form der Laufschrifterzeugung.

Die Länge der darzustellenden Information wird mit LAENGE (Zeile 7) eingestellt. Vor dem Start des Programms LAUF ist der Pufferbereich (PUFFER aus Programm 11.3) mit den gewünschten ASCII-Zeichen und in der gewählten Länge zu füllen. Mit dem Aufruf des Unterprogrammes BEGINN aus Programm 11.3 werden 8 Zeichen linksbündig aus dem RAM-Bereich ab PUFFER zur Anzeigeeinheit übertragen. Anschließend wird das 1.Zeichen aus PUFFER entnommen, im Register E zwischengespeichert und nach Verschieben des restlichen Puffers am Ende des Pufferbereiches (PUFFER + LAENGE) eingetragen. Danach wird der so veränderte Anfang des Pufferbereiches an die Anzeigeeinheit übertragen. Mit der im Abschnitt 11.2.1. gewählten Bitrate von 800 Baud ergibt sich der angestrebte Laufschrifteffekt.

```
 LOC   OBJ CODE M ZEILE QUELLE

                    1                ; MODUL ZUR ERZEUGUNG EINES "LAUFSCHRIFTEFFEKTES"
                    2                ; AN EINER ANZEIGEEINHEIT VQC10 MIT U 883
                    3
                    4
                    5          EXTERNAL        BEGINN,PUFFER
                    6
                    7  LAENGE: EQU     20                      ; LAENGE DES RINGPUFFERS
                    8
C200                9          ORG     0C200H
                   10
C200  CD0000    X   11  LAUF:   CALL    BEGINN                  ; 8 ZEICHEN AB PUFFER AN VQC10 SENDEN
C203  210000    X   12          LD      HL,PUFFER
C206  5E            13          LD      E,(HL)                  ; RINGPUFFER 1 MAL LINKSVERSCHIEBEN
C207  D5            14          PUSH    DE
C208  EB            15          EX      DE,HL
C209  210100    X   16          LD      HL,PUFFER+1
C20C  011300        17          LD      BC,LAENGE-1
C20F  EDB0          18          LDIR
C211  211300    X   19          LD      HL,PUFFER+LAENGE-1
C214  D1            20          POP     DE
C215  73            21          LD      (HL),E
C216  18E8          22          JR      LAUF                    ; NEUEN BLOCK AN VQC10 SENDEN
                   23
                   24          END
```

12. Alphanumerische Tastatur mit serieller und paralleler Schnittstelle

Rechnertastaturen sind üblicherweise in einer Matrixstruktur aufgebaut. Die jeweils betätigte Taste stellt eine elektrische Verbindung zwischen definierten Zeilen- und Spaltenleitungen der Matrix her. Prinzipiell läßt sich dieser Information hard- oder softwaremäßig der entsprechende Zeichenkode zuordnen. Hinsichtlich Bauelementaufwand und erreichbarem Bedienkomfort bietet sich auch bei diesem Problem der Einchipmikrorechner als Lösung an.

Sowohl die vorliegende Schaltung (s. Bild 12.1) als auch das zugehörige Programm wurden speziell für die mehrfach im Einzelhandel angebotene Tastatur *K 7659* entwikkelt. Eine Anpassung an andere Typen erfordert hauptsächlich die Neubelegung der Zeichentabellen. Weiterhin sind die konkreten Testbefehle für die Abfrage der Sondertasten SHIFT, SHIFT LOCK und CTRL entsprechend der tatsächlichen Anordnung in der Matrix zu verändern.

Die vorgestellte Lösung erfüllt folgende Forderungen:

- Abfrage von 12 Spalten- und 8 Zeilenleitungen,
- gleichzeitige serielle und parallele Zeichenausgabe,
- Repeatfunktion für alle Tasten,

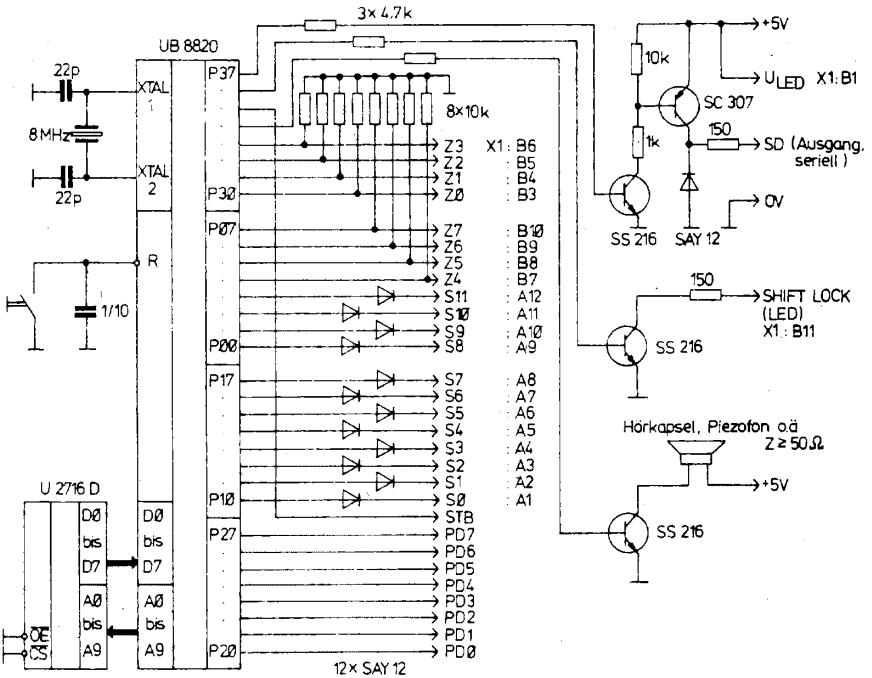


Bild 12.1 Tastaturbaugruppe mit U 882

- Umschaltmöglichkeit zwischen 4 frei wählbaren Zeichentabellen,
- LED-Anzeige für die SHIFT-LOCK-Funktion,
- wahlweise zuschaltbarer Tastatur-BEEP,
- Unterdrückung fehlerhafter Zeichenkodes bei gleichzeitiger Bedienung von 2 Tasten.

12.1. Hardware

Eine gleichzeitige serielle und parallele Zeichenausgabe läßt sich effektiv durch den Einsatz des *U882* realisieren. Tor 1 sowie das untere Halbbyte von Tor 0 bilden die erforderlichen 12 Spaltenleitungen. Entkoppeldioden verhindern Kurzschlüsse bei gleichzeitiger Betätigung mehrerer Tasten. Die 8 auszuwertenden Zeilenleitungen sind an P30 bis P33 bzw. P04 bis P07 angeschlossen. Widerstände gegen Masse sorgen für ein definiertes Potential.

Tor 2 dient der parallelen Ausgabe des erzeugten ASCII-Zeichens. Zusätzlich wird an P35 ein Strobe-Impuls bei jeder Ausgabe erzeugt. Er kann beispielsweise zur Interruptauslösung an der Gegenstelle dienen. Das serielle Ausgangssignal an P37 wird mit VT1 und VT2 verstärkt, so daß ein IFSS-Empfänger ansteuerbar ist.

Eine Rücksendung von Daten an die Tastatur ist nicht erforderlich und bei der vorliegenden Schaltung auch nicht möglich. Im Gegensatz zu den bisherigen Beispielen kommt eine Minimalvaiante der seriellen Datenübertragung zur Anwendung. Es ist weder Paritätsprüfung noch Quittungsbetrieb vorgesehen. Durch Modifikation des P3M-Registers (Setzen von Bit D6) ist die Übertragung mit Parität zwar ohne weiteres möglich, jedoch können dann ohne zusätzlichen Softwareaufwand nur noch Zeichen im 7-Bit-Format übertragen werden.

Zur visuellen Erkennung der SHIFT-LOCK-Funktion dient eine unmittelbar neben der entsprechenden Taste angeordnete LED. Der zur Stromverstärkung eingesetzte Transistor wird von P36 angesteuert.

Als akustische Unterstützung für den Bediener ist die Ausgabe eines sogenannten Tastatur-BEEP möglich (Dauer etwa 20 ms, 4-kHz-Rechteckimpulsfolge an P34). Über den Zeichenkode %80 läßt sich diese Funktion wahlweise zu- oder abschalten.

12.2. Software

Das ausreichend kommentierte Quellprogramm TASTATUR (Programm 12.1) erfordert keine detaillierte Erklärung. Ab Adresse %400 stehen im Abstand von %100 4 Zeichentabellen. In Abhängigkeit vom aktuellen Status (Grundzustand, SHIFT, CTRL, SHIFT & CTRL) wird das auszugebende Zeichen einer dieser Tabellen entnommen.

Ist ein Tabellenplatz mit 00 belegt, erfolgt bei der Betätigung der entsprechenden Taste keine Zeichenausgabe.

Programm 12.1

```
Z8ASM      4.0
LOC    OBJ CODE    STMT SOURCE-STATEMENT

                     1 TAST MODULE
                     2 ! Tastaturabfrage fuer K 7659 mit serieller und paralleler Ausgabe      !
                     3
                     4 GLOBAL
                     5
                     6 KEYBOARD PROCEDURE
                     7 ENTRY
                     8
P 0000               9 $ABS %000                          ! U 882                              !
                    10                                    ! Empfang : CTC 5, SIO /16           !
                    11
P 0000 FF           12 INTAB:   NOP
P 0001 FF           13          NOP
P 0002 FF           14          NOP
P 0003 FF           15          NOP
P 0004 FF           16          NOP
P 0005 FF           17          NOP
P 0006 FF           18          NOP
P 0007 FF           19          NOP
P 0008 FF           20          NOP
P 0009 FF           21          NOP
P 000A 016D         22 INT5:    WVAL TIME
P 000C 8F           23 INIT:    DI
P 000D 31  F0       24          SRP     #%F0
P 000F FC  80       25          LD      R15,#%80         ! Stack                              !
P 0011 6C  00       26          LD      R6,#0            ! Port 2 Ausgaenge                   !
P 0013 7C  41       27          LD      R7,#%41          ! SIO ohne Paritaet, P2 mit Pull-up  !
                    28
P 0015 8C  44       29          LD      R8,#%44          ! P1 und P00...P03 Ausgang, P04...07 !
                    30                                   ! Eingang, Stack intern              !
P 0017 BC  20       31          LD      R11,#%20         ! Interrupt Timer 1 erlaubt          !
P 0019 3C  03       32          LD      R3,#3            ! PRE1: Vorteiler = 1, autom. Neustart !
P 001B 2C  30       33          LD      R2,#%30          ! Zeitkonstante T1 fuer Repeat-Funktion !
P 001D 5C  05       34          LD      R5,#%5           ! PRE0: Vorteiler = 1, autom. Neustart !
P 001F 4C  20       35          LD      R4,#%20          ! Zeitkonstante T0 fuer SIO          !
P 0021 1C  0F       36          LD      R1,#%0F          ! T1 und T0 laden und Starten        !
P 0023 31  10       37          SRP     #%10             ! Arbeitsregistergruppe %10...%1F    !
P 0025 B0  1C       38          CLR     %1C              ! SHIFT aus                          !
P 0027 B0  1B       39          CLR     %1B              ! SHIFT-LOCK aus                     !
P 0029 46  03  80   40          OR      3,#%80           ! P37 = H                            !
P 002C 56  03  8F   41          AND     3,#%8F           ! SHIFT-LOCK-LED aus, BEEP aus       !
P 002F 9F           42          EI                       ! T1-Interrupt fuer Repeat frei      !
P 0030 B0  EC       43          CLR     R12              ! Spaltenzaehler = 0                 !
P 0032 E6  08  F0   44          LD      %8,#%F0          ! lange Pause fuer Repeat            !
P 0035 B0  09       45          CLR     %9               ! Repeat-Zeitzaehler auf Anfang      !
P 0037 B0  00       46          CLR     0
P 0039 B0  01       47          CLR     1
P 003B B0  02       48          CLR     2
P 003D B0  04       49 ZYKL:    CLR     4                ! Kontaktzaehler = 0                 !
P 003F D6  006B     50          CALL    MATRX            ! Matrixabfrage                      !
P 0042 A6  04  00   51          CP      4,#0             ! Kontakt ?                          !
P 0045 6B  13       52          JR      Z,NOKON          ! nein                               !
P 0047 A4  07  06   53          CP      6,7              ! altes Zeichen = neues Zeichen ?    !
P 004A E4  06  07   54          LD      7,6              ! Zeichen uebernehmen                !
P 004D 6B  12       55          JR      Z,KONT           ! Gleichheit                         !
P 004F E6  08  F0   56          LD      8,#%F0           ! neuer Zeichencode, langes Repeat   !
P 0052 E6  09  00   57 AUSG1:   LD      9,#0             !Repeat-Zeitzaehler = 0              !
P 0055 D6  0137     58          CALL    AUSGAB           ! Zeichen ausgeben                   !
P 0058 8B  E3       59          JR      ZYKL
P 005A B0  09       60 NOKON:   CLR     9
P 005C E6  07  00   61          LD      7,#0             ! Tastatur frei, Zeichen = 0         !
P 005F 8B  DC       62          JR      ZYKL
P 0061 A4  08  09   63 KONT:    CP      9,8              ! Repeat-Zeit erreicht ?             !
P 0064 7B  D7       64          JR      C,ZYKL           ! nein, fortsetzen                   !
P 0066 E6  08  30   65          LD      8,#%30           ! ja, kurzes Repeat-Intervall        !
P 0069 8B  E7       66          JR      AUSG1
P 006B B0  0A       67 MATRX:   CLR     %0A              ! Spaltenadresszaehler = 0           !
P 006D B0  0C       68          CLR     %0C              ! Spaltenzaehler = 0                 !
P 006F E6  01  01   69          LD      1,#1             ! 1. Spalte einschalten              !
P 0072 D6  0127     70 INZEI:   CALL    READ             ! Zeileninformation einlesen         !
P 0075 38  0F       71          LD      R3,%0F           ! 1. Lesen                           !
P 0077 FC  FF       72          LD      R15,#%FF
P 0079 FA  FE       73 LO1:     DJNZ    R15,LO1          ! Entprellzeit abwarten              !
P 007B D6  0127     74          CALL    READ
P 007E 48  0F       75          LD      R4,%0F
P 0080 A2  43       76          CP      R4,R3            ! Gleichheit ?                       !
P 0082 EB  EE       77          JR      NZ,INZEI         ! neu versuchen                      !
P 0084 A6  E3  00   78          CP      R3,#0            ! Kontakt ?                          !
P 0087 EB  13       79          JR      NZ,ZEI           ! ja, auswerten                      !
P 0089 06  0A  08   80          ADD     %0A,#%8          ! Spaltenadresse erhoehen            !
P 008C CF           81          RCF
P 008D 10  01       82          RLC     1
```

```
P 008F 10  00       83          RLC   0              ! naechste Spalte einschalten
P 0091 20  0C       84          INC   %0C            ! Spaltenzaehler +1
P 0093 A6  0C  0D   85          CP    %0C,#%0D       ! letzte Spalte erreicht ?
P 0096 7B  DA       86          JR    C,INZEI        ! nein, fortsetzen
P 0098 E4  0F  06   87          LD    6,%0F          ! ja, Zeichen eintragen
P 009B AF           88          RET
P 009C B0  0B       89 ZEI:     CLR   %0B            ! Zeilenzaehler = 0
P 009E B0  EC       90          CLR   R12            ! kein SHIFT
P 00A0 A6  0C  07   91 SHI:     CP    %0C,#7         ! letzte Zeile ?
P 00A3 EB  0E       92          JR    NZ,NOSHI       ! kein SHIFT oder SHIFT LOCK
P 00A5 76  0F  80   93          TM    %0F,#%80       ! SHIFT LOCK ?
P 00A8 6B  02       94          JR    Z,SH1          ! nein
P 00AA 8B  11       95          JR    ENDSHI         ! ja, Ende Sondertasten
P 00AC 76  0F  40   96 SH1:     TM    %0F,#%40       ! SHIFT ?
P 00AF 6B  02       97          JR    Z,NOSHI        ! nein, weiter
P 00B1 8B  0A       98          JR    ENDSHI         ! ja, Ende Sondertasten
P 00B3 A6  0C  06   99 NOSHI:   CP    %0C,#6         ! Spaltennummer fuer CTRL?
P 00B6 EB  08      100          JR    NZ,ZEICOD      ! nein, keine Sondertaste
P 00B8 76  0F  20  101          TM    %0F,#%20       ! CTRL ?
P 00BB 6B  03      102          JR    Z,ZEICOD       ! nein
P 00BD 8D  0CDF    103 ENDSHI: JP    SONDCO         ! Sondertaste auswerten
P 00C0 20  04      104 ZEICOD: INC   4              ! Kontaktzaehler + 1
P 00C2 B0  0B      105          CLR   %0B            ! Zeilenzaehler = 0
P 00C4 FC  07      106          LD    R15,#7         ! 8 Zeilen testen
P 00C6 76  0F  01  107 SH3:     TM    %0F,#1         ! Kontakt in Zeile ?
P 00C9 EB  07      108          JR    NZ,SH2         ! ja, Test abbrechen
P 00CB 20  0B      109          INC   %0B            ! nein, naechste Zeile
P 00CD CF          110          RCF
P 00CE C0  0F      111          RRC   %0F
P 00D0 FA  F4      112          DJNZ  R15,SH3        ! maximal 8 Zeilen
P 00D2 0C  04      113 SH2:     LD    R0,#4          ! Basisadresse f.Tabelle,H- Teil
P 00D4 1C  00      114          LD    R1,#0          ! L- Teil
P 00D6 04  0A  E1  115          ADD   R1,%0A
P 00D9 14  0B  E1  116          ADC   R1,%0B         ! Zeilennummer addieren
P 00DC 16  E0  00  117          ADC   R0,#0          ! Uebertrag beruecksichtigen
P 00DF E6  01  80  118 SONDCO:  LD    1,#%80         ! Spalte f. SHIFT/ SHIFT LOCK ein
P 00E2 56  00  F0  119          AND   0,#%F0
P 00E5 D6  0127    120          CALL  READ           ! Zeileninformation lesen
P 00E8 A6  0F  80  121          CP    %0F,#%80       ! SHIFT LOCK ?
P 00EB EB  04      122          JR    NZ,SH4         ! nein
P 00ED BC  01      123          LD    R11,#1         ! ja, Flag setzen
P 00EF 8B  0D      124          JR    SH5
P 00F1 A6  0F  40  125 SH4:     CP    %0F,#%40       ! SHIFT ?
P 00F4 6B  04      126          JR    Z,SH6          ! ja
P 00F6 CC  00      127          LD    R12,#0         ! nein, Flag zuruecksetzen
P 00F8 8B  04      128          JR    SH5
P 00FA CC  01      129 SH6:     LD    R12,#1         ! Flag fuer SHIFT setzen
P 00FC B0  EB      130          CLR   R11            ! SHIFT LOCK ausschalten
P 00FE 42  CB      131 SH5:     OR    R12,R11        ! SHIFT oder SHIFT LOCK auswerten
P 0100 E6  01  40  132          LD    1,#%40         ! Spalte fuer CTRL einschalten
P 0103 D6  0127    133          CALL  READ
P 0106 A6  0F  20  134          CP    %0F,#%20       ! CTRL ?
P 0109 EB  05      135          JR    NZ,SH7         ! nein
P 010B 46  EC  02  136          OR    R12,#2         ! ja, Flag setzen
P 010E 8B  03      137          JR    SH8
P 0110 56  EC  FD  138 SH7:     AND   R12,#%FD       ! CTRL - Flag ausschalten
P 0113 02  0C      139 SH8:     ADD   R0,R12         ! Offsett addieren
P 0115 82  20      140          LDE   R2,@RR0        ! Zeichen aus Tabelle holen
P 0117 29  06      141          LD    %6,R2          ! in Puffer uebernehmen
P 0119 76  EB  01  142          TM    R11,#1         ! SHIFT LOCK gesetzt ?
P 011C 6B  05      143          JR    Z,SH9          ! nein
P 011E 46  03  40  144          OR    3,#%40         ! ja, P36 auf H setzen
P 0121 8B  03      145          JR    SH10
P 0123 56  03  BF  146 SH9:     AND   3,#%BF         ! nein, P36 auf 0
P 0126 AF          147 SH10:    RET
P 0127 E4  03  0F  148 READ:    LD    %0F,3          ! Zeilen 0...3 lesen
P 012A 56  0F  0F  149          AND   %0F,#%0F       ! obere Tetrade = 0
P 012D E4  00  0E  150          LD    %0E,0          ! Zeilen 4...7 lesen
P 0130 56  0E  F0  151          AND   %0E,#%F0       ! untere Tetrade = 0
P 0133 44  0E  0F  152          OR    %0F,%0E        ! 8 Bit zusammenfassen
P 0136 AF          153          RET
P 0137 E4  06  F0  154 AUSGAB:  LD    SIO,%06        ! serielle Ausgabe
P 013A B0  02      155          CLR   2
P 013C 56  03  EF  156          AND   3,#%0EF        ! P34 = 0
P 013F E4  06  02  157          LD    2,%06          ! parallele Ausgabe
P 0142 EC  02      158          LD    R14,#2
P 0144 FC  80      159 LO5:     LD    R15,#%80
P 0146 DC  28      160 LO4:     LD    R13,#%28       ! Zeitkonstante
P 0148 DA  FE      161 LO2:     DJNZ  R13,LO2
P 014A 76  EA  01  162          TM    R10,#1         ! BEEP gesetzt ?
P 014D 6B  03      163          JR    Z,LO3          ! nein
P 014F B6  03  10  164          XOR   3,#%10         ! ja, 4 Ton ausgeben
P 0152 FA  F2      165 LO3:     DJNZ  R15,LO4
P 0154 EA  EE      166          DJNZ  R14,LO5
P 0156 46  03  20  167          OR    3,#%20         ! Strobe- Impuls f. Parallelausgabe
P 0159 FC  80      168          LD    R15,#%80
P 015B FA  FE      169 LO6:     DJNZ  R15,LO6
P 015D 56  03  DF  170          AND   3,#%DF         ! Strobe aus
```

```
P 0160 FC  80       171          LD      R15,#%80        ! Daten stabil halten
P 0162 FA  FE       172 LO7:     DJNZ    R15,LO7
P 0164 A6  06  80   173           CP      %06,#%80        ! Zeichen f. BEEP ein/aus ?
P 0167 EB  03       174           JR      NZ,ENDAUS       ! nein
P 0169 B6  EA  01   175           XOR     R10,#1          ! ja, aktuellen Status aendern
P 016C AF           176 ENDAUS:  RET
P 016D 20  09       177 TIME:    INC     %09             ! Repeat- Zeitzaehler + 1
P 016F BF           178           IRET
                    179
P 0170              180 END KEYBOARD
                    181
                    182 END TAST
```

13. Minimalsystem zur Programmentwicklung für Einchiprechner mit dem Kleincomputer *KC 85/1* bzw. *KC 87*

Die Entwicklung des Anwenderprogramms für EMR kann nach den Methoden erfolgen, die prinzipiell auch beim *U 880*-System Verwendung finden. Dabei dient ein Editorprogramm zur Erstellung des Quelltextes. Durch einen nachfolgenden Assemblerlauf entsteht ein lauffähiges Maschinenprogramm. Unter professionellen Bedingungen stehen derartige Programmpakete für die meisten Rechner auf *U 880*-Basis zur Verfügung (Bürocomputer *A 5120*, Personalcomputer *PC 1715, MC 80.22*). Im allgemeinen hat der Amateur aber keinen ständigen Zugriff zu diesen Geräten. Es müssen also weitere praktikable Methoden der EMR-Programmerstellung gesucht werden.

Natürlich ist ein ausgereiftes Entwicklungssystem nicht durch Minimalvarianten vollständig ersetzbar. Wenn eine entsprechende Lösung auf Basis des Kleincomputers *KC 85/1* dennoch beschrieben wird, hat das folgende Gründe:

- Das Gesamtprogramm für den EMR belegt häufig nur einen Teil des zur Verfügung stehenden Programmspeicherraums; eine optimale Programmierung in bezug auf die Minimierung der belegten Speicherzellen ist daher meist von untergeordneter Bedeutung.
- In der überwiegenden Zahl der Fälle läuft das Anwenderprogramm innerhalb eines fest vorgegebenen Adreßbereiches (0000 bis %07FF bzw. %0800 bis %0FFF, je nach eingesetztem Schaltkreistyp). Auch die Adressen der Interruptvektoren sowie die E/A-Schnittstellen sind vorgegeben. Die Programmerstellung auf der Basis absoluter Adressen bringt deshalb im Vergleich zu *U 880*-Systemen geringere Nachteile.
- Der Befehlssatz der Einchipmikrorechner *U 881/U 882* ist sehr regelmäßig gestaltet und daher leicht einprägsam. Dazu trägt auch die relativ kleine Zahl von Basisbefehlen bei.

Eine praktische Konsequenz aus den genannten Aspekten besteht in der Möglichkeit, die Programmierung bei vertretbarer Effektivität direkt in Maschinensprache durchzuführen. Voraussetzung dafür ist allerdings ein RAM-Modifikationsprogramm für den Entwicklungsrechner, das über das reine Beschreiben und Lesen des Speichers hinausgeht. Speziell muß eine Relativsprungberechnung implementiert sein. Auch das Einfügen und Streichen von Bytes in Verbindung mit einer Verschiebung bestimmter Adreßbereiche ist in diesem Zusammenhang von Bedeutung. Natürlich werden die so entstandenen Programme etwas weniger «elegant» ausfallen als bei Verwendung eines komfortablen Entwicklungssystems. In vielen, praktisch interessanten Fällen bringt dieser Umstand jedoch keine funktionellen Probleme mit sich. Die mangelnde Dokumentierbarkeit so erstellter Programme muß dagegen als deutlicher Nachteil gesehen werden.

Ein weiterer Aspekt bei der Softwareerstellung ist die Testung von Programmen. Im professionellen Bereich finden für diesen Zweck sogenannte Emulatorsysteme Verwendung, die z. B. einen Schrittbetrieb bei der Programmabarbeitung erlauben. Derartige hard- und softwaremäßig aufwendigen Systeme stehen dem Amateuer jedoch noch seltener zur Verfügung als geeignete Editor- und Assemblerprogramme.

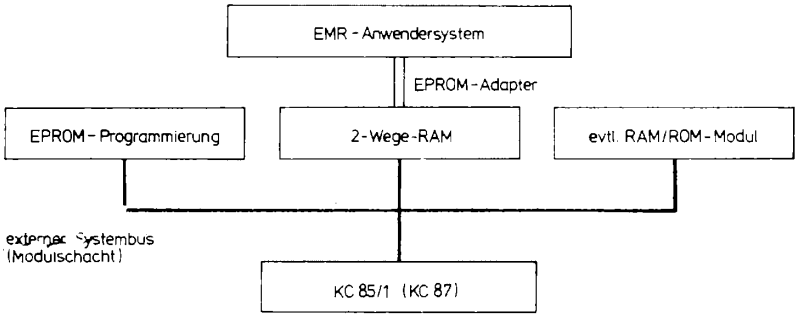


Bild 13.1 Minimalvariante für EMR-Programmentwicklung am KC 85/1, Übersichtsschaltbild

Die vorgestellte Lösung geht davon aus, daß die beste Emulation durch Verwendung des Anwendersystems selbst erreichbar ist. Bild 13.1 zeigt die Minimalvariante mit einem im Prinzip beliebigen *U 880*-System. Die konkrete Auslegung ist allerdings auf den *KC 85/1* zugeschnitten.

13.1. 2-Wege-RAM

Wesentliches Merkmal der Anordnung ist ein sogenannter 2-Wege-RAM. Neben der normalen Schreib-/Lesefunktion vom Bus des Entwicklungsrechners aus kann ein Lesen des Speicherinhalts auch über eine 2. Gruppe von Adreß- und Datentreibern erfolgen. Damit ist ein Zugriff durch den EMR gegeben. Der Zwei-Wege-RAM dient als Programmspeicher des Anwendersystems, d. h., der in der endgültigen Version eingesetzte EPROM wird für die Dauer der Programmentwicklung durch einen vom Entwicklungsrechner aus modifizierbaren RAM-Bereich ersetzt. Über die Umschaltung der beiden Zugriffsmöglichkeiten ist ein schneller Wechsel zwischen den Betriebsarten Programmerstellung und Programmtestung erreichbar. Neben der Freigabe des RAM für den EMR-Speicherzugriff ist zusätzlich ein Hardwarereset zu erzeugen, damit das Programm definiert anlaufen kann. Ist die Softwareentwicklung auf diese Art und Weise erfolgreich abgeschlossen, kann mit der nachfolgend vorgestellten Programmiereinrichtung der Speicherinhalt auf EPROM ausgelagert werden.

Die Verbindung zwischen 2-Wege-RAM und Anwendersystem erfordert einen 24poligen Adapter. Beim aufgebauten Muster kam ein defekter EPROM zum Einsatz. Allerdings ist dafür zu sorgen, daß alle Verbindungen zwischen den Anschlüssen und defektem Chip unterbrochen sind. Ansonsten kommt es zu «unerklärlichen» Fehlfunktionen. Das Verbindungskabel sollte nicht länger als 30 cm sein, um Laufzeitprobleme bei Ausnutzung der maximalen Taktfrequenz des EMR auszuschließen.

Bild 13.2 zeigt den Stromlaufplan des 2-Wege-RAM. Er ist kompatibel zu allen *K 1520*-Systemen. Die Verwendung der Treiberschaltkreise *DS 8212* und *DS 8216* resultiert aus der relativ guten Verfügbarkeit dieser Bauelemente für den Amateur. Moderner und platzsparender ist natürlich die Verwendung von Schaltkreisen *DS 8282* bzw. *8286* bei prinzipiell gleicher Schaltung. Während die Datenbustreiber auf der *U 880*-Seite bidirektional arbeiten müssen, genügt für den EMR-Zugriff ein Treiber ohne Datenrichtungsumschaltung. Die Umschaltung zwischen beiden Zugriffsarten

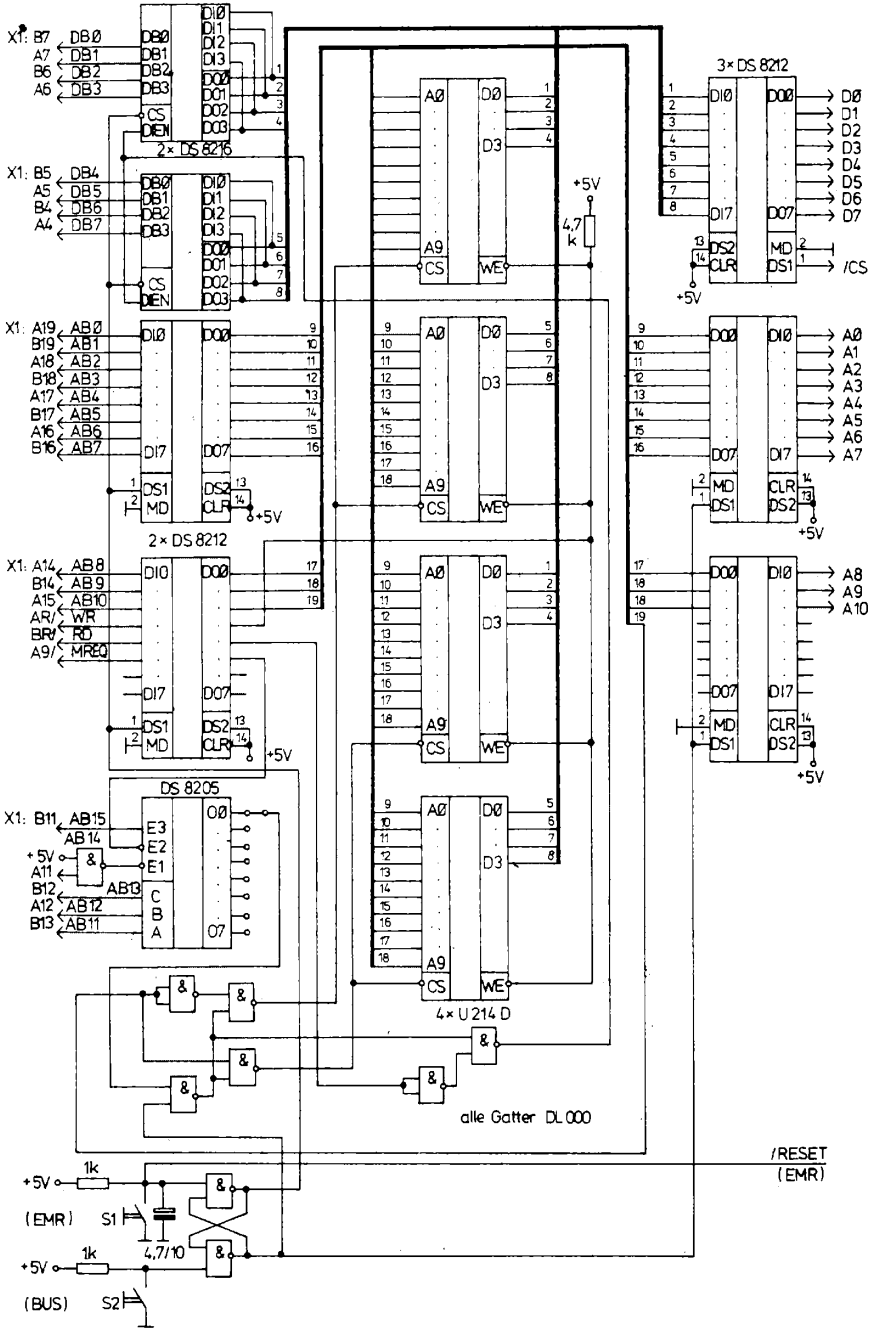


Bild 13.2 2-Wege-RAM

geschieht mit einem RS-Flipflop in Verbindung mit den Tastern S1 und S2. LEDs an den Flipflop-Ausgängen zeigen den aktuellen Status an. Prinzipiell ist auch eine softwaremäßige Umschaltung der Betriebsart, z. B. über OUT-Befehle, möglich. Der dazu notwendige Hardwareaufwand wird vor allem dann beträchtlich, wenn eine freie Zuordnung der Ausgabeadressen im Sinne universeller Verwendbarkeit gegeben sein soll.

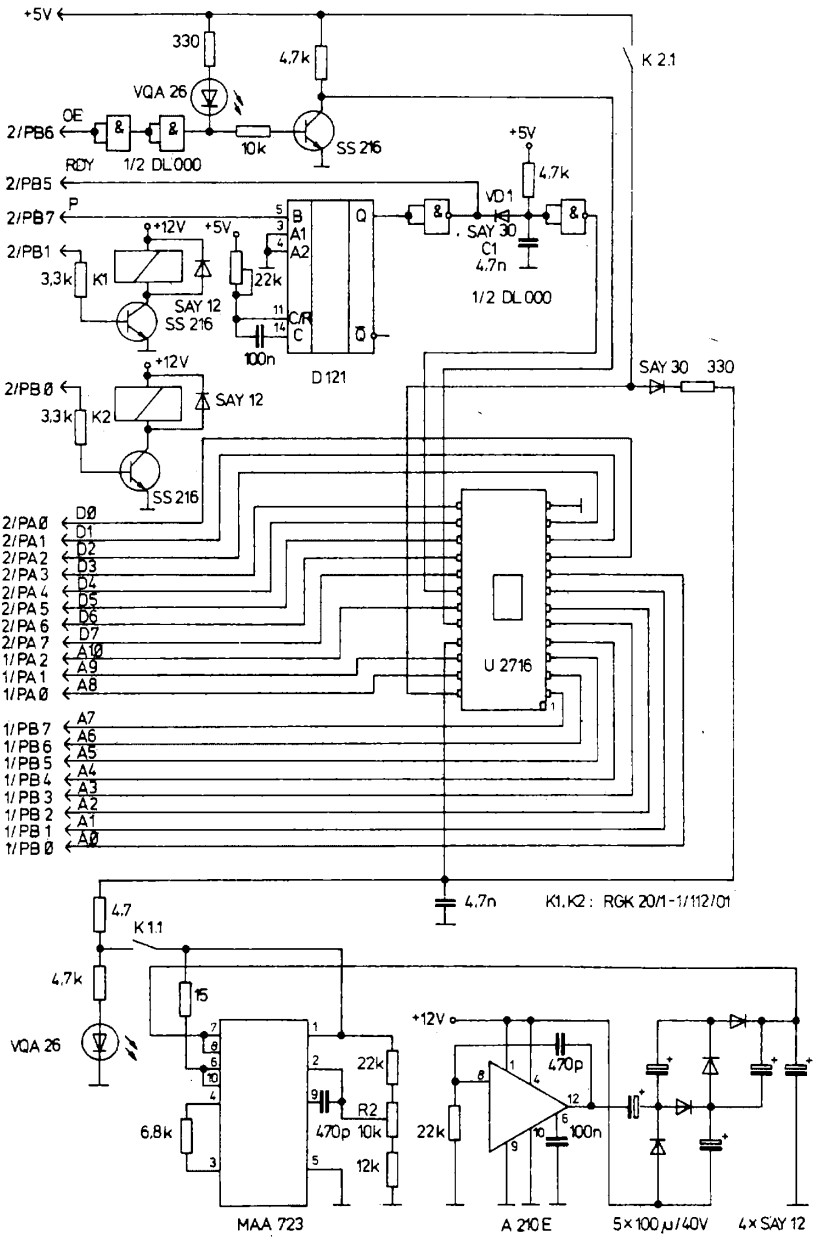


Bild 13.3 EPROM-Programmiereinrichtung für KC

13.2. EPROM-Programmiereinrichtung

Nachdem das Programm mit Hilfe des 2-Wege-RAM erstellt und getestet wurde, kann eine Abspeicherung auf Kassette erfolgen. Der letzte Schritt zur Fertigstellung des Anwendersystems ist die Programmierung eines EPROM.

Bild 13.3 zeigt eine entsprechende Schaltung. Sie basiert auf der in [4] vorgestellten Baugruppe. Alle Adreß-, Daten- und Steuersignale werden über 2 PIO-IS *UB 855* bereitgestellt. Dieser Teil der Schaltung ist nicht dargestellt, da er keinerlei Besonderheiten aufweist. Um die Lauffähigkeit des nachfolgend beschriebenen Programms zu gewährleisten, sind folgende I/O-Adressen einzuhalten:

```
PIO 1       Port A
Daten:       28H        Adresse, H-Teil A8 bis A10
Steuerwort: 29H

PIO 1       Port B
Daten:       2AH        Adresse, L-Teil A0 bis A7
Steuerwort: 2BH

PIO 2       Port A
Daten:       38H        Daten D0 bis D7
Steuerwort: 39H

PIO 2       Port B
Daten:       3AH        Steuerleitungen
Steuerwort: 3BH
```

Im Hinblick auf den konkreten Einsatzfall wurde die Baugruppe ausschließlich zur Programmierung des EPROM *U 2716 D* bzw. äquivalenter Typen ausgelegt. Ein Monoflop *D 121* erzeugt 1-ms-Ausgangsimpulse. Durch mehrfaches, softwaregesteuertes Anstoßen läßt sich die Dauer des Programmierimpulses festlegen. Das Halteglied VD1, *R*1, *C*1 garantiert, daß die Programmierimpulse keine Unterbrechungen im 1-ms-Takt aufweisen. Sie sind sonst prinzipiell vorhanden, da stets erst die Rückflanke des Monoflop-Ausgangsimpulses abgewartet wird, bevor ein erneutes Anstoßen erfolgt.

Die Programmiereinrichtung beinhaltet auch eine Transverterschaltung zur Programmierspannungserzeugung mit dem Schaltkreis *A 210 E*. Die Rohspannung am Eingang des nachfolgenden Spannungsreglers *MAA 723* beträgt etwa 30 V. Mit *R*2 läßt sich die Programmierspannung von 25 V genau einstellen.

13.3. Software

Die speziell für den *KC 85/1* entwickelte Software gestattet in Verbindung mit den beschriebenen Zusatzbaugruppen die Entwicklung lauffähiger Programme für Einchipmikrorechner. Das Programm besteht aus den Teilen RAM und EPROG. Im RAM-Modus werden prinzipiell 64 Bytes auf dem Bildschirm dargestellt. Ein Kursorsymbol weist auf das modifizierbare Byte. Im einzelnen sind folgende Operationen möglich:

Programmteil RAM

Kommando	Operation
0 bis F	Beschreiben der durch die Kursorposition angewählten RAM-Zelle mit einer Hexzahl (Op-Code bzw. Daten)
→	Kursorposition 1 Byte vorwärts
←	Kursorposition 1 Byte rückwärts
+	Kursorposition 64 Bytes vorwärts (Seitenschaltung vorwärts)
−	Kursorposition 64 Bytes rückwärts (Seitenschaltung rückwärts)
J	Bei nachfolgender Eingabe einer Absolutadresse wird die relative Sprungweite zur Kursorposition berechnet und angezeigt. Nach Betätigung von ENTER erfolgt die Eintragung in die durch den Kursor spezifizierte RAM-Zelle.
W	Nach der Eingabe wird eine Adresse angefordert, die den Wirkungsbereich der Kommandos «Q» und «S» nach oben begrenzt. Oberhalb der angezeigten Adresse bleiben die Verschiebekommandos wirkungslos.
Q	Verschieben des RAM-Inhalts um ein Byte nach oben und Einfügen eines NOP-Befehls auf der aktuellen Kursorposition. Anschließend kann der NOP-Befehl beliebig überschrieben werden.
S	Streichen des durch die Kursorposition spezifizierten Bytes. Der RAM-Inhalt wird um ein Byte nach unten verschoben. Der Verschiebebereich wird, ebenso wie bei «Q», durch die angegebene Endadresse begrenzt.
P	Eintritt in den Programmteil EPROG.

Programmteil EPROG

L	Lesen des EPROM-Inhalts ab eingegebener Anfangsadresse auf den durch die eingegebene Anfangs- und Endadresse definierten RAM-Bereich.
V	Vergleich von EPROM- und RAM-Inhalt innerhalb der vorgegebenen Adressen. Bei Fehlern wird die betreffende RAM- und EPROM-Adresse angezeigt. Über ENTER kann bis zur nächsten fehlerhaften Adresse bzw. zur Endadresse fortgesetzt werden. Mit «L» ist der Rücksprung ins Menü möglich.
T	Test des EPROM auf %FF im definierten Adreßbereich. Fehlerhafte Adressen werden angezeigt. Fortsetzung mit ENTER, Rücksprung ins Menü mit «L».
P	Programmierung aller Zellen des EPROM im spezifizierten Adreßbereich. Ein Test auf vollständige Löschung erfolgt nicht, d. h., eine Mehrfachprogrammierung ist möglich und in Ausnahmefällen auch notwendig.
N	Neueingabe des Adreßbereiches für EPROM und RAM.
R	Eintritt in Programmteil RAM.
O	Rücksprung ins Betriebssystem (OS) des Rechners.

Das Programm EOS (Programm 13.1) läuft ab Adresse 3000H und beansprucht etwa 2,5 KByte im RAM. Auf Anfrage kann das Programm auch auf andere Adressen gebunden werden. Die Implementierung auf anderen Rechnern ist ohne größere Probleme

möglich, da die speziellen Systemrufe des *KC 85/1*-Betriebssystems nicht zur Anwendung kommen. Lediglich das physische Tastaturprogramm wird direkt gerufen. Es ist durch ein beliebiges anderes Programm ersetzbar, sofern bei Austritt das ASCII-Zeichen der betätigten Taste im Akkumulator steht.

Programm 13.1

```
     1.Record der Datei BUCH.13.1 vom 880205
       0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F
3000 * C3 0D 30 45  4F 53 20 20  20 20 20 00  00 CD 68 33   *..0EOS      ...h3*
3010 * 21 FF 00 22  26 38 21 28  EC 22 02 38  21 B4 34 06   *!.."&8!(.".8!.4.*
3020 * 03 CD 03 31  CD 1A 31 CD  1A 31 21 B7  34 06 11 CD   *...1..1..1!.4...*
3030 * 03 31 CD 1A  31 CD 7F F7  FE 52 CA BA  30 FE 50 CA   *.1..1....R..0.P.*
3040 * 44 30 18 F1  21 AB EE 22  04 38 CD 68  33 21 28 EC   *D0..!..".8.h3!(.*
3050 * 22 02 36 21  C8 34 06 0F  CD 03 31 21  01 50 22 12   *".8!.4....1!.P".*
3060 * 38 CD 1A 31  21 43 34 06  08 CD 03 31  CD DB 30 21   *8..1!C4....1..0!*
3070 * 00 38 CD 28  36 21 4B 34  06 06 CD 03  31 CD DB 30   *.8.(6!K4....1..0*
3080 * 21 06 38 CD  28 36 2A 06  38 ED 5B 00  38 B7 ED 52   *!.8.(6*.8.[.8..R*
3090 * 22 0E 38 CD  1A 31 21 94  34 06 0A CD  03 31 CD DB   *".8..1!.4....1..*
30A0 * 30 21 0A 38  CD 28 36 CD  1A 31 C3 E0  34 C3 BA 30   *0!.8.(6..1..4..0*
30B0 * 4D 4F 4E 49  20 20 20 20  00 00 CD 68  33 22 02 02   *MONI    ...h3"..*
30C0 * 22 39 00 21  38 00 36 C3  CD 68 33 C3  E1 31 1A FE   *"9.!8.6..h3..1..*
30D0 * 40 38 04 E6  0F C6 09 ED  6F 13 C9 ED  5B 02 38 D6   *@8......o...[.8.*
30E0 * CD 7F F7 D1  FE 0D 28 09  FE 08 28 09  CD 0B 31 18   *......(...(...1.*
30F0 * EE 2A 02 38  C9 2A 02 38  36 20 2B 36  00 22 02 38   *.*.8.*.86 +6.".8*

     2.Record der Datei BUCH.13.1 vom 880205
       0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F
3100 * 18 DD 00 7E  CD 0B 31 23  10 F9 C9 E5  2A 02 38 77   *...~..1#....*.8w*
3110 * 23 36 00 22  02 38 E1 C9  10 E9 E5 D5  F5 11 00 EC   *#6.".8..........*
3120 * 2A 02 38 36  20 B7 ED 52  38 07 21 28  00 19 EB 18   **.86 ..R8.!(....*
3130 * EF ED 53 02  38 EB 36 00  F1 D1 E1 C9  36 20 7A B3   *..S.8.6.....6 z.*
3140 * 28 0A 1B 23  18 F6 22 2B  00 CD 83 F7  D1 E1 F1 C9   *(..#.."+........*
3150 * 2A 20 38 D5  ED 5B 22 38  0E 08 E5 21  FA EC 22 02   ** 8..["8...!..".*
3160 * 38 E1 06 08  E5 2A 02 38  2B 36 20 E1  2A 20 38 B7   *8....*.8+6 .* 8.*
3170 * ED 52 20 06  2A 02 38 2B  36 3E 2A 20  38 CD A7 31   *.R .*.8+6>* 8..1*
3180 * 23 22 20 38  E5 2A 02 38  23 22 02 38  E1 10 D6 E5   *#" 8.*.8#".8....*
3190 * 2A 02 38 0D  D5 11 10 00  19 D1 22 02  38 E1 20 C2   **.8.......".8. .*
31A0 * D1 C3 B3 32  CD 1A 31 7E  E5 CB 3F CB  3F CB 3F CB   *...2..1~..?.?.?.*
31B0 * 3F B7 FE 0A  38 06 D6 09  F6 40 18 02  F6 30 2A 02   *?...8....@...0*.*
31C0 * 38 77 23 22  02 38 E1 7E  E6 0F FE 0A  38 06 D6 09   *8w#".8.~....8...*
31D0 * F6 40 18 02  F6 30 E5 2A  02 38 77 23  22 02 38 E1   *.@...0.*.8w#".8.*
31E0 * C9 21 01 EC  22 02 38 06  0E 21 F3 32  CD 03 31 CD   *.!..".8..!.2..1.*
31F0 * DB 30 21 20  38 CD 28 36  CD 1A 31 21  03 33 06 0F   *.0! 8.(6..1!.3..*

     3.Record der Datei BUCH.13.1 vom 880205
       0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F
3200 * CD 03 31 2A  20 38 22 22  38 00 21 22  38 CD 37 36   *..1* 8""8.!"8.76*
3210 * CD 1A 31 21  13 33 06 04  CD 03 31 06  0B 3E 20 CD   *..1!.3....1..> .*
3220 * 0B 31 23 10  FA 2A 22 38  CD A7 31 C3  50 31 D5 CD   *.1#..*"8..1.P1..*
3230 * 7F F7 D1 00  FE 47 28 A9  FE 57 CA 92  37 FE 4A CA   *.....G(...W..7.J.*
3240 * 33 37 FE 2B  CA 70 32 FE  2D CA 7C 32  FE 09 CA 91   *37.+.p2.-.|2....*
3250 * 32 FE 08 CA  AD 32 FE 0D  FE 50 CA 44  30 FE 4F CA   *2....2...P.D0.O.*
3260 * 00 00 28 4F  FE 53 CA FE  36 FE 51 CA  16 37 18 BE   *..(O.S..6.Q..7..*
3270 * 21 37 EC 22  02 38 C3 03  32 C3 50 31  21 37 EC 22   *!7.".8..2.P1!7."*
3280 * 02 38 2A 20  38 11 80 00  B7 ED 52 22  20 38 C3 03   *.8* 8.....R" 8..*
3290 * 32 2A 22 38  23 22 22 38  2A 20 38 11  40 00 B7 ED   *2*"8#""8* 8.@...*
32A0 * 52 22 20 38  C3 1A 33 22  02 38 C3 0A  32 2A 22 38   *R" 8..3".8..2*"8*
32B0 * 2B 18 E2 21  5E EC 36 00  23 22 02 38  ED 5B 02 38   *+..!^.6.#".8.[.8*
32C0 * D5 CD 7F F7  D1 FE 47 D2  33 32 FE 30  DA 33 32 CD   *......G.32.0.32.*
32D0 * 0B 31 D5 CD  7F F7 D1 CD  0B 31 2A 02  38 36 20 D5   *.1.......1*.86 .*
32E0 * CD 7F F7 FE  0D 20 F9 2A  22 38 CD CE  30 CD CE 30   *..... .*"8..0..0*
32F0 * C3 91 32 63  50 45 49 43  48 45 52 2D  20 41 44 52   *..2SPEICHER- ADR*

     4.Record der Datei BUCH.13.1 vom 880205
       0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F
3300 * 2E 3A 20 4B  55 52 53 4F  52 2D 41 44  52 45 53 53   *.: KURSOR-ADRESS*
3310 * 45 3A 20 42  59 54 45 11  80 00 2A 22  38 ED 5B 20   *E: BYTE...*"8.[ *
3320 * 38 B7 ED 52  DA 43 33 3E  40 BD E5 21  37 EC 22 02   *8..R.C3>@..!7.".*
3330 * 38 E1 C2 0A  32 11 40 00  2A 20 38 B7  19 22 20 38   *8...2.@.* 8..." 8*
3340 * C3 70 32 11  40 00 2A 20  38 B7 19 22  20 38 11 40   *.p2.@.* 8..." 8.@*
3350 * 00 B7 19 21  37 EC 22 02  38 2A 20 38  11 80 00 B7   *...!7.".8* 8.....*
3360 * ED 52 22 20  38 C3 09 32  E5 D5 C5 F5  21 00 EC 11   *.R" 8..2....!...*
3370 * 01 EC 01 B0  03 36 20 ED  B0 F1 C1 D1  E1 C9 21 A2   *.....6 .......!.*
3380 * EF 22 02 38  21 B4 33 06  06 CD 03 31  CD DB 30 21   *.".8!.3....1..0!*
3390 * 24 38 CD 28  36 21 BA 33  06 06 CD 03  31 CD DB 30   *$8.(6!.3....1..0*
33A0 * 1A FE 59 CA  AD 33 FE 4E  CA EE 33 18  D1 CD 68 33   *..Y..3.N..3...h3*
33B0 * 2A 24 38 E9  47 4F 54 4F  3A 20 20 52  55 4E 3F 20   **$8.GOTO:  RUN? *
33C0 * 4A 55 4D 50  2D 41 44 52  3A 20 20 44  49 53 54 41   *JUMP-ADR:  DISTA*
33D0 * 4E 43 45 3A  20 45 4E 44  41 44 52 2E  53 48 49 46   *NCE: ENDADR.SHIF*
33E0 * 54 3A 20 53  48 49 46 54  20 42 49 53  3A 20 F3 CD   *T: SHIFT BIS: ..*
33F0 * 68 33 3E EF  D3 81 3E 01  D3 81 2A 24  38 FB E9 53   *h3>...>...*$8..S*
```

```
      5.Record der Datei BUCH.13.1 vom 880205
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
3400 * 50 20 50 43 20 41 46 20 42 43 20 44 45 20 48 4C   *P PC AF BC DE HL*
3410 * 20 49 58 20 49 59 20 5A 20 43 20 53 20 50 2F 56   * IX IY Z C S P/V*
3420 * 20 28 50 43 29 20 28 48 4C 29 20 53 59 53 54 45   * (PC) (HL) SYSTE*
3430 * 4D 2D 43 4F 4E 54 52 4F 4C ED 4D 45 50 52 4F 47   *M-CONTROL.MEPROG*
3440 * 20 20 20 52 41 4D 20 41 42 3A 20 20 42 49 53 3A   *   RAM AB:  BIS:*
3450 * 20 45 50 52 4F 4D 20 53 54 45 43 4B 45 4E 21 54   * EPROM STECKEN!T*
3460 * 45 53 54 20 20 28 54 29 20 56 45 52 47 4C 20 28   *EST  (T) VERGL (*
3470 * 56 29 20 4C 45 53 45 4E 20 28 4C 29 20 50 52 4F   *V) LESEN (L) PRO*
3480 * 47 52 20 28 50 29 20 52 41 4D 3A 20 20 45 50 52   *GR (P) RAM:  EPR*
3490 * 4F 4D 3A 20 45 50 52 4F 4D 20 41 42 3A 20 43 59   *OM: EPROM AB: CY*
34A0 * 43 4C 3A 20 20 45 50 52 4F 4D 20 4C 4F 45 53 43   *CL:  EPROM LOESC*
34B0 * 48 45 4E 21 45 4F 53 52 41 4D 20 28 52 29 2C 45   *HEN!EOSRAM (R),E*
34C0 * 50 52 4F 47 20 28 50 29 45 50 52 4F 4D 2D 54 59   *PROG (P)EPROM-TY*
34D0 * 50 3A 20 32 37 31 36 20 FF 00 FF 00 FF FF FF 20   *P: 2716 ....... *
34E0 * 21 D8 34 F3 01 29 02 ED B3 01 2B 02 ED B3 01 39   *!.4..)....+....9*
34F0 * 02 ED B3 01 3B 02 ED B3 3E 40 D3 3A FB 21 51 34   *....;...>@.:.!Q4*

      6.Record der Datei BUCH.13.1 vom 880205
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
3500 * 06 0E CD 1A 31 CD 03 31 CD DB 30 CD 1A 31 CD 1A   *....1..1..0..1..*
3510 * 31 3E 42 D3 3A 06 27 21 5F 34 CD 03 31 CD 1A 31   *1>B.:.'!_4..1..1*
3520 * CD 1A 31 CD DB 30 1A FE 56 CA 91 35 FE 4C 28 1B   *..1..0..V..5.L(..*
3530 * FE 50 CA 40 36 FE 54 CA 00 36 FE 52 CA BA 30 FE   *.P.@6.T..6.R..0.*
3540 * 4F CA 00 00 FE 4E CA 44 30 18 D8 ED 5B 0A 38 2A   *O....N.D0...[.8**
3550 * 00 38 CD 75 35 DB 38 C5 4F 06 32 10 FE DB 38 B9   *.8.u5.8.O.2...8.*
3560 * C1 20 F2 77 CD 80 35 20 E9 CD 68 33 21 28 EC 22   *. .w..5 ..h3!(."*
3570 * 02 38 C3 E0 34 C5 0E 2A ED 59 0E 28 ED 51 C1 C9   *.8..4..*.Y.(.Q..*
3580 * 23 13 E5 D5 ED 5B 06 38 13 B7 ED 52 7C B5 D1 E1   *#....[.8...R|...*
3590 * C9 ED 5B 0A 38 2A 00 38 CD 75 35 DB 38 C5 4F 06   *..[.8*.8.u5.8.O.*
35A0 * 32 10 FE DB 38 B9 C1 20 F2 BE BE C4 BF 35 CD 80   *2...8.. .....5..*
35B0 * 35 20 E5 CD 68 33 21 28 EC 22 02 38 C3 E0 34 D5   *5 ..h3!(."..8..4.*
35C0 * E5 2A 04 38 22 02 38 21 67 34 06 05 CD 03 31 E1   *.*.8".8!.4....1.*
35D0 * D1 22 08 38 E5 D5 21 08 38 CD 37 36 21 8C 34 06   *."..8..!.8.76!.4.*
35E0 * 08 CD 03 31 D1 ED 53 08 38 D5 21 08 38 CD 37 36   *...1..S.8.!.8.76*
35F0 * CD 7F F7 FE 4C 20 06 D1 E1 2A 08 38 C9 D1 E1 C9   *....L ...*.8....*

      7.Record der Datei BUCH.13.1 vom 880205
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
3600 * ED 5B 0A 38 2A 00 38 CD 75 35 DB 38 06 20 10 FE   *.[.8*.8.u5.8. ..*
3610 * DB 38 FE FF C4 BF 35 CD 80 35 20 EB 21 28 EC 22   *.8....5..5 .!(."*
3620 * 02 38 CD 68 33 C3 E0 34 23 CD CE 30 CD CE 30 2B   *.8.h3..4#..0..0+*
3630 * CD CE 30 CD CE 30 C9 23 CD A7 31 2B CD A7 31 C9   *..0..0.#..1+..1.*
3640 * 2A 12 38 22 10 38 2A 00 38 ED 5B 0A 38 CD 75 35   **.8".8*.8.[.8.u5*
3650 * CD 80 35 20 F8 3E 12 D3 3A 3E FF D3 39 3E 00 D3   *..5 .>..:>..9>..*
3660 * 39 21 64 00 22 0C 38 CD 1A 31 21 9E 34 06 06 CD   *9!d."..8..1!.4...*
3670 * 03 31 2A 02 38 22 14 38 23 23 22 02 38 2A 02 38   *.1*.8".8##".8*.8*
3680 * 22 16 38 ED 5B 0A 38 2A 00 38 CD 75 35 ED 4B 10   *".8.[.8*.8.u5.K.*
3690 * 38 48 7E D3 38 3E 93 D3 3A 3E 12 D3 3A DB 3A CB   *8H~.8>..:>..:.:.*
36A0 * 6F 28 FA 06 20 3E 00 0D B9 20 EA CD 80 35 20 DA   *o(.. >... ...5 .*
36B0 * E5 2A 14 38 22 02 38 21 10 38 CD A7 31 3E 00 BE   *.*.8".8!.8..1>..*
36C0 * 35 E1 20 BF 3E 40 D3 3A CD 68 33 21 00 EC 22 02   *5. .>@.:.h3!..".*
36D0 * 38 C3 E0 34 CD 68 33 21 28 EC 22 02 38 06 0F 21   *8..4.h3!(."..8..!*
36E0 * A5 34 CD 03 31 CD DB 30 C3 E0 34 E5 B7 ED 52 E1   *.4..1..0..4...R.*
36F0 * 38 03 ED B0 C9 0B 09 EB 09 EB 03 ED B8 C9 2A 22   *8.............*"*

      8.Record der Datei BUCH.13.1 vom 880205
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
3700 * 38 23 ED 5B 22 38 CD D5 37 ED 4B 26 38 CD EB 36   *8#.["8..7.K&8..6*
3710 * 2A 22 38 C3 95 32 ED 5B 22 38 13 CD D5 37 ED 4B   **"8..2.["8...7.K*
3720 * 26 38 2A 22 38 CD EB 36 2A 22 38 36 00 22 22 38   *&8*"8..6*"86.""8*
3730 * C3 95 32 21 73 EF 22 02 38 21 C0 33 06 0A CD 03   *..2!s."..8!.3....*
3740 * 31 CD DB 30 21 28 38 CD 28 36 21 CA 33 06 0B CD   *1..0!(8.(6!.3...*
3750 * 03 31 2A 26 38 ED 5B 22 38 B7 ED 52 2B 22 2A 38   *.1*(8.["8..R+"*8*
3760 * 21 2A 38 CD A7 31 D5 CD 7F F7 D1 FE 4A 28 C4 FE   *!*8..1.....J(..*
3770 * 0D 28 02 18 F1 2A 22 38 11 2A 38 1A 77 CD 83 37   *.(...*"8.*8.w..7*
3780 * C3 91 32 E5 C5 21 73 EF 06 28 36 20 23 10 FB C1   *..2..!s..(6 #...*
3790 * E1 C9 21 73 EF 22 02 38 21 D5 33 06 0E CD 03 31   *..!s."..8!.3....1*
37A0 * CD DB 30 21 2C 38 CD 28 36 CD 83 37 21 1A EC 22   *..0!,8.(6..7!..."*
37B0 * 02 38 21 2C 38 CD 37 38 C3 2E 32 21 40 EC 22 02   *.8!,8.76..2!@."..*
37C0 * 38 36 5A D3 20 C3 2E 32 21 40 EC 22 02 38 36 20   *86Z. ..2!@."..86 *
37D0 * D3 21 C9 2E 32 E5 D5 2A 2C 38 ED 5B 22 38 B7 ED   *.!..2..*,8.["8..*
37E0 * 52 22 26 38 D1 E1 C9 C3 0A 05 00 00 00 00 00 00   *R"&8............*
37F0 * 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   *................*

%
```

Sachwortverzeichnis

Adressenausgabe 23
Adressierung
–, direkt 37
–, relativ 38
–, unmittelbar 37
Adressierungsarten 37
Adreßtor 12
alphanumerische Anzeigeeinheit 100
alphanumerische Tastatur 114
Analog-Digital-Umsetzer 87
Analogsignale 87
Ankopplungsmodul für KC 85/1 bzw. 87 97
Anschlußbelegung 18
Anzeigeprogramme 79
Arithmetik- und Logikbefehle 41
Arithmetik-Logik-Einheit 6, 10
Arithmetikprogramme 74
ASCII-Zeichensatz 108

BASIC-Interpreter 7
Befehlsbeschreibung 39
Befehlsdekoder 10
Befehlsgruppen 37
Befehlslesemechanismus 10
Befehlssatz 37
Befehlszyklus-Zeitverläufe 22
Betriebsartenregister 15
Bondversion 7
Bootstrap-Ladeprogramm 7

continous mode 15
Controller für direkten Speicherzugriff 5
CPU-Steuerbefehle 47
CPU-Steuerung 10

Dekoderschaltkreis V 40511 80
DIL-Gehäuse 6
Divisionsprogramm 76
Drehzahlmessung 86

Ein-/Ausgabeleitungen 12
Eingabegerät 89
–, Steuerprogramm 92
Einkartenrechner 5
Entwicklungsversion 6
EPROM-Programmiereinrichtung 123

Festspannungsregler 91
Flagregister 33
Frequenzverdopplung 15

Handshake-Steuerung 13

IFSS-Schnittstelle 62
–, Steckverbinderbelegung 65
Impulsdauer-Messung 84
Interrupt 25
Interrupt-Anmelderegister 17, 32
Interrupt-Maskierungsregister 31
Interrupt-Prioritätsregister 32
Interruptstartadressen 16
Interrupt-Steuerregister 35
Interruptsteuerung 16
Interruptverarbeitung 6

Konstantstromquelle 64
Konvertierungsprogramme 78

Ladebefehle 40
Laufschrift 113
Leitungstreiber, integriert 66
lokale Netze 63

Maximalwertberechnung 77
Mehrrechnersysteme 5
Mikrorechnerfamilie 5
Mittelwertberechnung 77
Multiplikationsprogramm 75

on-chip-Taktgenerator 10
Operationskode-Lesezyklus 19

Paritätsprüflogik 61
Periodendauer-Messung 86
Pipelining 10
Pipelining-Befehlsverarbeitung 22
Pollingbetrieb 14
power-down-Version 7
Power-on-Resetschaltung 56
Programmentwicklung 119
programmierbarer paralleler Ein-/Ausgabeschaltkreis 5
programmierbarer serieller Ein-/Ausgabeschaltkreis 5
programmierbarer Zähler/Zeitgeber 5
Programmlauf-Überwachungsschaltung 57
Programmspeicher 7
Programmspeicherleitungen 6
Programmsteuerbefehle 45
Programmzähler 10

Quarzgenerator 59
QUIL-Gehäuse 6

Rechnertastatur 114
Registeradressierung 38
Registerbeschreibung 27
Registerpointer 33
Registersatz 10

ROM-Maske 7
Rotations- und Verschiebebefehle 43
Rücksetzeingang 19

Schaltungstechnik 51
serielle Schnittstelle 15
serielles Ein-/Ausgaberegister 34
Serien-Parallel-Wandlung 15
single pass mode 15
Softwareerstellung 119
Spannungsüberhöhung 19
Spannungsüberhöhung am RESET-Eingang 58
Speicher
–, adressenaktiviert 24
–, chipaktiviert 24
–, extern 23
–, intern 25
Speicherbereiche 11
Speicherlesezyklus 23
Speicherschreibzyklus 23
Speicherzeitverhalten 23
Stack 12
Stackpointer 34
standardisierte Schnittstellen 61
Statusregister 10
Statussignale 6
Steckfassungen 7
Steuerprogramme 104
Steuerregister 10
Stromeinspeisung 62

T0-Vorteilerregister 30
T1-Vorteilerregister 31

Taktfrequenz 7
Taktratenteiler 104
Taktsynchronisation 104
Tastatur 89
Technologie 9
Test-ROM 7
Testmodus 19
Tor-0- und Tor-1-Betriebsart 27
Tor-2-Betriebsart 28
Tor-3-Betriebsart 28
Transverterschaltung 123
Typenspektrum 8

Übersichtsschaltbild 9
Übertragsflag 33
Universalregister 10
U 880-Mikrorechnerfamilie 5

Verarbeitungsbreite 5
Verarbeitungsgeschwindigkeit 5
V.24-Schnittstelle 65
–, Steckverbinderbelegung 66

Zähler/Zeitgeber 14
Zähler/Zeitgeberbetriebsart 29
Zähler/Zeitgeberregister 31
Zählfrequenzmesser 86
Zeichengenerator 108
zeitmultiplexe Ziffernanzeige 81
Zentrale Verarbeitungseinheit 5
Zugriffsverfahren 24

2-Wege-RAM 120

Literatur

[1] *M. Roth/K. Geidel, PAS 881* Einchip-Mikroprozessor Z8-Beschreibung, Ilmenau 1984
[2] *W. Bennewitz/H. Podszuweit*, Programmierung von Einchipmikrorechnern, Berlin 1987
[3] *M. Bankel*, Einchipmikrorechner *U 881*, *U 882* und *U 883*. In: radio fernsehen elektronik 34 (1985) H. 2, S. 81 bis 84
[4] *B. Hübler/K. Evert*, Ausbaufähiger Mikrocomputer mit dem *U 880*, Berlin 1985
[5] *M. Lohöfner*, Tabellarischer Befehlssatz der Einchip-Mikrorechner *U 881 D* und *U 882 M*. In: radio fernsehen elektronik 35 (1986) H. 7, S. 422
[6] *H. Kieser/M. Bankel*, Einchipmikrorechner, Berlin 1987
[7] *P. Weber* u. a., Einchipmikrorechner *U 882* mit Display und Dezimaltastatur. In: radio fernsehen elektronik 35 (1986) H. 11, S. 693 bis 695
[8] *L. Claßen/U. Oefler*, Wissensspeicher Mikroprozessorprogrammierung, Berlin 1986
[9] TGL 37360 Unipolarer Einchipmikrorechner UB 881 D
[10] TGL 42639 Unipolarer Einchipmikrorechner UB 882 M
[11] TGL 38607 Unipolarer Einchipmikrorechner UB 883 D
[12] *B. Lindemann*, Lokale Computernetze, Berlin 1988

Einführend wird mit der inneren Struktur der Einchip-Mikrorechner aus dem Kombinat Mikroelektronik vertraut gemacht. Dabei werden gleichzeitig Besonderheiten und Unterschiede gegenüber anderen zentralen Rechnereinheiten herausgearbeitet. Aufbauend auf der Grundschaltung, werden Hardwarelösungen mit unterschiedlichem Komfort und Schwierigkeitsgrad vorgestellt. Die Programmentwicklung ist direkt auf die Nutzung von Kleincomputern zugeschnitten und für die Lösung verschiedener Teilprobleme ausgelegt. Die Erläuterung des Befehlssatzes und Informationen zu dessen Anwendung runden den Inhalt ab.

ISBN 3-327-00776-4
00650