CIP-Kurztitelaufnahme der Deutschen Bibliothek

Schulé, Roland:
Einführung in die Mikroprozessor-Anwendung /
Roland Schulé; Axel Gruppe. Unter Mitarb.
von Michael Zillgitt. Jean Pütz (Hrsg.). –
1. Aufl. – Köln: vgs, 1987.
(Experimente)

ISBN 3-8025-1239-1

NE: Gruppe, Axel:

Bildnachweis

Bilder 1.3, 1.4, 1.6: IBM, Stuttgart
Bild 1.5: Siemens, München
Bild 1.9: VDI-Verlag, Düsseldorf
Bild 7.26: Michael Zillgitt, Köln
Bild 11.4: Fischer-Werke, Waldachtal 3
Alle anderen Fotos:
Wolfgang Arntz, Wermelskirchen 2

1. Auflage 1987

Umschlaggestaltung: Fred Papen Grafik Design,
Overath-Steinenbrück
Zeichnungen: Peter Kohlöffel, Nonnenhorn
Gesamtherstellung:
Universitätsdruckerei H. Stürtz AG, Würzburg
Printed in Germany
ISBN 3-8025-1239-1

Vorwort

Verehrte Leser!

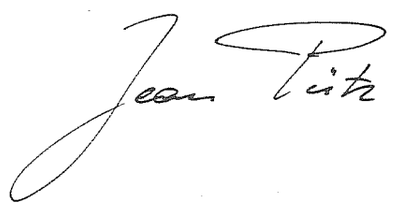
Rund 16 Jahre ist es nun her, daß ich im WDR meinen ersten Fernsehkurs „Einführung in die Elektronik" realisieren konnte. Schon damals erschien, ebenfalls in der vgs, ein Begleitbuch dazu. Bald wurde die Notwendigkeit offensichtlich, das zu Lernende im wörtlichen Sinne auch „begreifbar" zu machen: Der Leser bzw. der Lernende soll durch eigene Experimente an den Stoff herangeführt werden. In Ergänzung zum Begleitbuch wurden also – ebenso wie zur Digitaltechnik-Serie – Experimentier-Systeme entwickelt. Somit bildete sich ein Medienverbund mit mehreren Bestandteilen heraus: Sendungen, Bücher und Hardware – alles eng aufeinander bezogen und mit vielen praxisnahen Übungen (mit Lösungen) in den Büchern.
Dieses Konzept resultierte nicht zuletzt aus meinen Erfahrungen, die ich während meiner eigenen Ausbildung (über den zweiten Bildungsweg) sammeln konnte, sowie später als Lehrer und schließlich als Wissenschafts-Journalist. Gerade beim Erarbeiten der Mikroelektronik kommt es besonders auf eigenes Experimentieren und auf didaktisch gut durchdachtes Material an.
Nach den geschilderten Gesichtspunkten wurde auch dieses Buch über die Mikroprozessor-Anwendung konzipiert. Es ist Teil eines ganz besonderen Medienpaketes: Die zugehörige Hardware – die Interfaceplatine BERT – ist nicht nur ein intelligentes Interface, sondern ebenso ein eigenständiger kleiner Computer. Dazu weiter unten mehr.
Zuvor ein paar Worte über die Aspekte, auf die es uns beim Entwickeln von Buch und Hardware entscheidend ankam.
Viele von uns sind im beruflichen oder im Hobby-Bereich während der letzten Jahre mehr und mehr mit elektronischen, digitalelektronischen und schließlich mikroprozessor-gesteuerten Anlagen in Berührung gekommen. Der Gang der technischen Entwicklung kann hier nicht im einzelnen beschrieben werden. Festzuhalten ist jedoch, daß das Wissen hier immer rascher veraltet und sich der Einzelne immer schneller auf Neues einstellen muß. Zudem wird der Anwendungsbereich des Mikroprozessors sozusagen von Tag zu Tag größer; heute kommt fast kein technisches Gerät gehobener Leistung mehr ohne ihn aus, ob im Haushalt die Waschmaschine, im Auto die Kraftstoffanlage, in der Unterhaltungselektronik die Video- und die Stereo-Anlage. Die Aufzählung ließe sich beinahe beliebig verlängern. Um so wichtiger ist es angesichts der rasanten Entwicklung, die Grundlagen zu beherrschen. Und diese sind – wie schon erwähnt – am effektivsten mit Hilfe selbst nachvollziehbarer Experimente zu erarbeiten.
Für die Einführung in die Mikroprozessor-Anwendung war also vor allem ein geeignetes Experimentiersystem zu entwickeln. Computer der unterschiedlichsten Preis- und Leistungsklassen sind ja inzwischen in großen Stückzahlen verbreitet. Wie aber kann man damit ein Kurspaket realisieren? Denken Sie an die vielen verschiedenen Computersysteme mit all den daraus entstehenden Problemen der Kompatibilität. Als ersten Schritt erwogen wir die Schaffung eines von der deutschen Industrie neu zu entwickelnden Standard-Computers, der sowohl möglichst preiswert als auch BTX-fähig sein sollte. Dabei war das BTX-System zur Rückkopplung mit den Kursteilnehmern vorgesehen.
Dieses ehrgeizige Projekt ließ sich leider nicht verwirklichen – unter anderem, weil die vielen Besitzer der anderen Computer (vom VC20 bis hin zum IBM-PC) nicht oder nur unter großen Schwierigkeiten hätten teilnehmen können. Schließlich sollte natürlich niemand schon aus rein technischen Gründen ausgeschlossen sein; unser Ziel war und ist es doch, diese heutzutage extrem wichtig gewordene neue Technologie möglichst vielen zugänglich und verständlich zu machen.
Besonderes Augenmerk richteten wir auf die Benutzung des Mikroprozessors für Steuerungs- und Regelungs-Aufgaben: Der Computer sollte „Fühler und Augen" sowie „Arme und Beine" erhalten, also mit der Umwelt Signale austauschen können. Zuerst mußten wir uns daher um die Schnittstellen kümmern, d. h. um die Verbindungen des Computers mit den externen Geräten. Hier ließ sich die notwendige Vereinheitlichung ohne weiteres erzie-

len. Denn sehr viele der gebräuchlichen Home- und Personal Computer besitzen eine der sogen. V24-Norm entsprechende Schnittstelle; hier wird unsere völlig neu entwickelte Interface-Platine BERT angeschlossen. Auf ihr haben alle Benutzer dieselben Experiment-Anschlüsse.
Die Forderung nach Einheitlichkeit ist damit zwar erfüllt, aber zur technischen Realisierung muß BERT über die V24-Schnittstelle Signale mit dem Computer austauschen. Dazu müßte jedoch jeder Benutzer seinen Computer selbst entsprechend programmieren. Das erfordert allerdings bereits recht gute Kenntnisse, die in einer Einführung naturgemäß nicht vorausgesetzt werden dürfen. Deswegen hat die Interface-Platine BERT ganz besondere Eigenschaften bekommen: Sie ist so „intelligent", daß die gewünschten Aktivitäten der Peripherie mit einfachen Kommandos auszulösen sind; diese werden vom Computer her – beispielsweise aus einem BASIC-Programm heraus – an BERT gesendet (über die V24-Schnittstelle). Der Benutzer kann sich daher voll auf die logische Aufbereitung der Steuerungs- oder Regelungs-Aufgabe konzentrieren und muß sich nicht im einzelnen um das Erzeugen oder Verarbeiten der Signale auf den Experimentleitungen kümmern.
Damit BERT dies alles leisten kann, ist er als eigenständiger kleiner Computer ausgelegt; er wird mit Hilfe eines (in diesem Buch auch abgedruckten) Terminalprogramms programmiert. Und schließlich ist BERT auch autonom einsetzbar; aber das ist schon eher etwas für die Fortgeschrittenen. Auch sie finden in den späteren Kapiteln dieses Buches viele Anregungen.
Unser Buch ist insgesamt als Einführung angelegt, auch wenn es zum Ende hin ein relativ hohes Niveau erreicht. Es beginnt mit einfachsten Steuerungen, und neue Inhalte sowie komplexere Abläufe werden Schritt für Schritt anhand von Experimenten und Übungen erarbeitet. Wir wählten die Programmiersprache BASIC, weil sie bei den in Frage kommenden Computern weit verbreitet und außerdem leicht zu erlernen ist. Dabei verkennen wir keineswegs die Vorteile der strukturierten Sprachen wie etwa Pascal; die Benutzung unseres Interfaces BERT ist dementsprechend auch nicht auf BASIC beschränkt. Will man in einer anderen Sprache programmieren, dann muß lediglich sichergestellt werden, daß die V24-Schnittstelle für Ein- und Ausgabe anzusprechen ist. Die durchgehende Benutzung von BASIC in diesem Buch ist also durchaus nicht als eine prinzipielle Einschränkung anzusehen.

Betonen möchte ich noch: Wie mit den anderen von mir herausgegebenen Büchern auch, wenden wir uns hier an alle, die die Mikroprozessor-Technik vom Praxisbezug her betrachten wollen, und keineswegs nur an den mathematisch oder technisch Vorgebildeten. Ob interessierte Laien, Facharbeiter, Ingenieure oder Akademiker: jeder benötigt heutzutage Kenntnisse in der Mikroprozessortechnik; und keiner von ihnen hat von vornherein Vorteile gegenüber dem anderen, sie zu erlernen und das wichtige Prinzip zu verstehen und anzuwenden: Ersatz von Hardware durch Software.
Zum Schluß möchte ich mich bei den Autoren Roland Schulé und Axel Gruppe bedanken, die mit großem Einsatz sehr viel Arbeit und Zeit investiert haben: in die Konzeption des Experimentiermaterials, die Erarbeitung der Versuchsaufbauten und Programme und schließlich auch in die didaktische Feinarbeit. Auch dem Verlag und seinem Lektor Michael Zillgitt sei an dieser Stelle gedankt. Wer sich unter unseren Lesern bereits selbst mit Dingen beschäftigt hat, wie sie dieses Buch bietet, wird vielleicht einschätzen können, welcher ungeheure Aufwand dahintersteckt.
Schließlich danke ich auch Herrn B. Thomsen, der unseren BERT aus dem Schaltplan „erstehen" ließ und ein praxisgerechtes Gerät daraus machte. Den Löwenanteil an Arbeit und Ideen bei BERT hat Herr R. Schulé geleistet: Er hat BERTs Kommando-Interpreter erdacht und bis zur Serienreife entwickelt sowie ausgetestet. Dieser im System-EPROM von BERT angesiedelte Interpreter ist ein wahres Bonbon für alle, die die Mikrocomputertechnik erarbeiten wollen. Etwas Vergleichbares ist bisher in keinem anderen Einplatinencomputer zu finden.
Das Medienpaket zur Einführung in die Mikroprozessor-Anwendung wird noch vervollständigt durch 10 Fernsehfilme, die in Kooperation des WDR mit dem FWU (Institut für Film und Bild in Wissenschaft und Unterricht, München) derzeit vorbereitet werden; die Ausstrahlung ist für 1987/88 vorgesehen.

Und nun wünsche ich Ihnen viel Erfolg!

Ihr



Inhalt

5. Analog und Digital

6. Anzeige und Tastatur

7. Mechanische Antriebe und Positionierung

8. BERT ist ein eigenständiger Computer

9. Programm-Entwicklung mit BERT

10. Datenübertragung

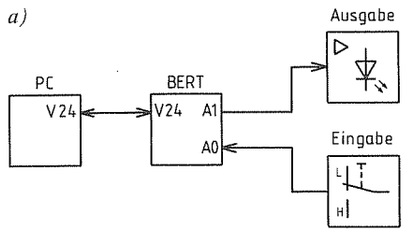
11. Zwei Anwendungsbeispiele für BERT

12. Anhang

Die Vorbereitung des ersten Experiments

In *Kapitel 1* wurde schon die Interface-Platine BERT vorgestellt, so daß an dieser Stelle nur die wenigen Anschlüsse zu erwähnen sind, die für den ersten Versuch mit ihr benötigt werden. Zunächst gehören hierzu die Versorgungsspannungen, und zwar die Wechsel- oder Gleichspannung von ca. 8 V für BERT und für Eingabe- und Ausgabe-Platine; hier kann, weil die Stromaufnahme nicht zu groß ist, direkt die an BERT abzugreifende 5-V-Gleichspannung verwendet werden.
An die 5polige DIN-Buchse von BERT wird die V24-Leitung (bzw. RS232-Leitung) vom PC angeschlossen. Insbesondere ist hier auf den Spannungspegel zu achten, da viele Computer mit den „echten" V24-Pegeln (+12 V und −12 V) arbeiten und nicht mit TTL-Pegeln (0 V und +5 V) wie z. B. der Commodore C64. Bei einigen Rechnern ist daher eine zusätzliche Pegelanpassung vorzusehen. Zu diesen Fragen werden im Anhang ausführliche Erläuterungen gegeben; dort wird auch die Anschluß-Belegung für das anzufertigende Verbindungs-Kabel zwischen BERT und PC gezeigt.
Von den vier Ports der experimentseitigen Schnittstelle unserer Interfaceplatine benutzen wir für den ersten Versuch nur die Stifte A0 und A1 des Ports A, und zwar wird der Signalausgang der Eingabeplatine (also der Schalter) mit A0 und der Signaleingang der Ausgabeplatine (also die Leuchtdiode bzw. Lampe) mit A1 verbunden. Die komplette Verschaltung von PC, BERT und zwei Experimentierplatinen ist in *Bild 2.5* zu sehen.

a)



b)

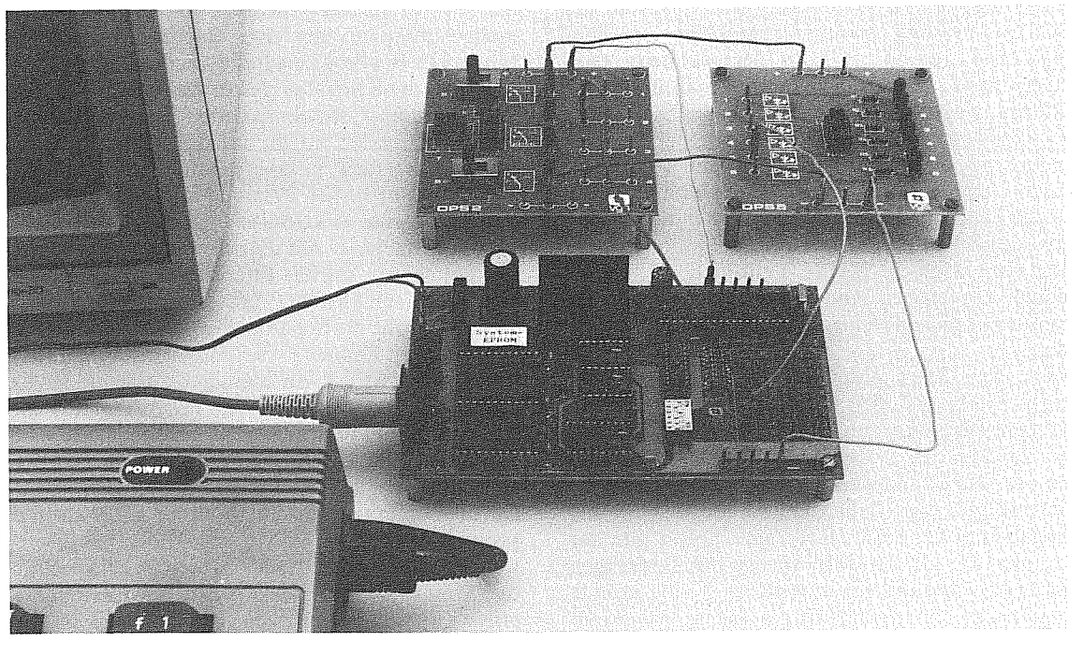


Bild 2.5: Verschaltung des Computers (PC), der Interfaceplatine BERT sowie der Eingabe- und der Ausgabe-Platine für das erste Experiment (EINAUS). Alle Experiment-Platinen und BERT sowie der PC müssen natürlich eine gemeinsame elektrische Masse haben. Diese Verbindungen sind hier nicht gezeichnet. Zwischen BERT und PC ist die Masseleitung im V24-Kabel (vgl. Anhang) realisiert.

Die Programmierung der Experiment-Schnittstelle

Erinnern wir uns daran, daß das Steuerungsprogramm EINAUS *(Bild 2.4)* die Eingabe des Schalterzustands von der Tastatur erwartet und den Lampenzustand auf dem Bildschirm ausgibt. In dem soeben beschriebenen Experiment-Aufbau sind aber praxisgerecht wirklich ein Schalter und eine Lampe (Leuchtdiode) vorgesehen. Das bedeutet jedoch: Der Datenaustausch des PCs mit der „Außenwelt" (Peripherie) erfolgt jetzt über die V24-Leitung und die daran angeschlossene Interfaceplatine BERT – und nicht mehr über Tastatur und Monitor! Also müssen wir für die in *Bild 2.5* dargestellte Verschaltung des Rechners mit der Peripherie die Eingabe- und Ausgabe-Anweisungen im Programm EINAUS ändern, weil jetzt die V24-Schnittstelle aktiviert werden muß.

Die Initialisierung

Im Normalbetrieb des PCs wird die V24-Schnittstelle nicht angesprochen. Daher ist der Rechner vor dem Beginn des eigentlichen Steuerprogramms auf die Zusammenarbeit mit BERT vorzubereiten. Diese sogenannte *Initialisierung* benötigt nur wenige Programmzeilen, für die ja vor der Zeile 100 unseres Programms EINAUS noch Platz ist (vgl. *Bild 2.4*). Dieser Vorspann sieht zwar für die unterschiedlichen Rechnertypen jeweils anders aus (d. h. er ist rechnerspezifisch), doch können die wesentlichen Punkte der Initialisierung am Beispiel des Commodore C64, des IBM-PCs sowie des Apple II und der jeweils Kompatiblen verdeutlicht werden. Alle diese Programme sind in der Struktur ähnlich aufgebaut. Um ihr Verständnis zu erleichtern, werden soweit möglich die gleichen Zeilennummern verwendet *(Bild 2.6)*.
Die Initialisierung beginnt mit dem Aktivieren der V24-Schnittstelle des PCs.
Beim C64 ist dafür ein besonderer „Datenkanal" vorgesehen, der eröffnet wird mit der BASIC-Anweisung

```
OPEN 2,2,0,CHR$(8)+CHR$(1)
```

Beim IBM-PC wird der Datenkanal mit einer ähnlichen OPEN-Anweisung eröffnet. Die Schreibweise lautet hier:

```
OPEN "COM1:1200,N,8,2" AS #2
```

Beim Apple II wird die V24-Schnittstelle durch einige PRINT-Befehle mit Steuerzeichen (Zeilen 22 bis 34) an den Schnittstellen-Baustein eingestellt.
Nach der Datenkanal-Eröffnung kann bei den Eingabe- und Ausgabe-Anweisungen im weiteren Programm diese Schnittstelle unter Bezug auf ihre Nummer (2) angewählt werden (s. u.).
Mit diesen Befehlen im Programm INIT ist von seiten des PCs die Art und Weise der Datenübermittlung auf der V24-Leitung festgelegt. Dazu gehören die Übertragungs-Geschwindigkeit (1200 Baud, d. h. 1200 Bits pro Sekunde) und weitere Einzelheiten der Betriebsweise (Vollduplex, X-Draht-Handshake) sowie Vereinbarungen über das Format der gesendeten Daten (8 Bits Wortlänge, 1 Stopbit, keine Paritäts-Erkennung). Auf die Bedeutung dieser sogen. *Übertragungs-Parameter* werden wir in *Kapitel 10* zurückkommen. Zur Benutzung anderer, insbesondere schnellerer Übertragungs-Geschwindigkeiten finden Sie im Anhang nähere Informationen.
Selbstverständlich muß BERT mit den gleichen Parametern („Kenndaten") für die Übertragung arbeiten, damit die Kommunikation funktioniert. Die Übertragungs-Geschwindigkeit kann am 6fach-DIL-Schalter (Port E) auf der BERT-Platine eingestellt werden: Für 1200 Baud müssen die Schalter 0 und 2 auf 1 (ON) und Schalter 1 auf 0 (OFF) stehen; die im Programm INIT mit X bezeichneten Bits sind für die Festlegung der Baud-Rate ohne Bedeutung. Die anderen Übertragungs-Parameter sind an BERT so voreingestellt, daß die Kommunikation zwischen PC und BERT jetzt auf Anhieb funktionieren sollte.
Auf das Einrichten der V24-Schnittstelle folgen im Initialisierungs-Programm drei PRINT-Anweisungen, die Meldungen für den Benutzer auf dem Bildschirm ausgeben.
Wichtig für den weiteren Verlauf der Initialisierung ist die Aufforderung, die Reset-Taste an BERT zu drücken. BERT wird damit, wie schon erwähnt, in seinen Ausgangszustand versetzt. Er bestätigt dies durch Senden des Zeichens : (Doppelpunkt) über die V24-Leitung. Auf diesen Doppelpunkt wartet der PC (Zeile 50, bzw. 50 ff. beim Apple II). Empfängt er ihn, so bedeutet dies, daß die Kommunikation zwischen BERT und PC funktioniert. Der PC sendet daraufhin seinerseits das Kommando

```
GO@%1018
```

über die V24-Leitung an BERT (Zeile 60 bzw. 60 ff.).

Bild 2.6a:

```
10 REM INIT   (C64, VC20)
11 REM ====
12 REM INITIALISIERUNG VON BERT
15 REM ALS INTERFACE
17 REM
20 OPEN 2,2,0,CHR$(8)+CHR$(1)
30 PRINT CHR$(147);CHR$(18);"         BERT-INTERFACE-BETRIEB          "
35 PRINT"    1200 BAUD: PORT E AUF 5 = XXX101    ";
40 PRINT "   BITTE RESET-TASTE AN BERT DRUECKEN!  ";
50 GET#2,DU$:IF DU$<>":" THEN 50
60 PRINT#2,"G0@%1018"
70 INPUT#2,DU$:INPUT#2,DU$
80 PRINT CHR$(147);CHR$(18);"         BERT-INTERFACE-BETRIEB          ";
90 PRINT "          START BASIC-PROGRAMM          "
```

Bild 2.6b:

```
10 REM Programm INIT (IBM-PC)
12 REM Initialisierung von BERT
15 REM als Interface
17 REM
20 OPEN "COM1:1200,N,8,2" AS #2
25 WIDTH 40 : KEY OFF : CLS : COLOR 0,7
30 PRINT"          BERT-INTERFACE-BETRIEB          ";
35 PRINT"    1200 BAUD: PORT E AUF 5 = XXX101    "; : COLOR 7,0
40 PRINT"   BITTE RESET-TASTE AN BERT DRUECKEN!  "
45 IF EOF(2) THEN 45
50 DU$=INPUT$(LOC(2),#2) : IF DU$<>":" THEN 45
60 PRINT#2,"G0@%1018"
65 FOR ZEIT=1 TO 1000 : NEXT
70 DU$=INPUT$(LOC(2),#2)
80 CLS : COLOR 0,7 : PRINT"         BERT-INTERFACE-BETRIEB          ";
90 PRINT"          START BASIC-PROGRAMM          "; : COLOR 7,0
```

Bild 2.6c:

```
10  REM INIT (APPLE II+/E MIT SSC)
11  REM ====
12  REM INITIALISIERUNG VON BERT
15  REM ALS INTERFACE
17  REM
20 D$ =  CHR$ (4): REM DOS CONTROL ZEICHEN
21 A$ =  CHR$ (1): REM SSC CONTROL ZEICHEN
22  PRINT D$;"PR#2": PRINT D$;"IN#2"
23  PRINT A$;"8 BAUD": REM 1200 BAUD
24  PRINT A$;"0 DATA": REM 8 BIT + 1 STOPBIT
25  PRINT A$;"0 PARITY": REM KEIN PARITAETSBIT
26  PRINT A$;"1 TRANSLATE": REM GROSS/KLEIN
27  PRINT A$;"ECHO DISABLE": REM ECHO AUS
28  PRINT A$;"FIND DISABLE": REM KEINE ZEICHEN VON DER TASTATUR
29  PRINT A$;"0 CR": REM KEINE CR VERZOEGERUNG
30  PRINT A$;"0 LF": REM KEINE LF VERZOEGERUNG
```



```
31  PRINT A$;"0 FF": REM KEINE FF VERZOEGERUNG
32  PRINT A$;"MASK ENABLE": REM KEIN LF (INPUT)
33  PRINT A$;"LF DISABLE": REM KEIN LF OUTPUT
34  PRINT D$;"PR#0": PRINT D$;"IN#0"
35  HOME : INVERSE
36  PRINT "          BERT-INTERFACE-BETRIEB          "
37  NORMAL
38  PRINT "    1200 BAUD: PORT E AUF 5 = XXX101"
40  PRINT "   BITTE RESET-TASTE AN BERT DRUECKEN!"
50  PRINT D$;"PR#2": PRINT D$;"IN#2"
52  GET DU$: IF DU$ <  > ":" THEN 52
60  PRINT
61  PRINT D$;"PR#2": PRINT D$;"IN#2"
63  FOR I = 1 TO 8
64  READ C$
65  PRINT C$;
66  NEXT I
67  PRINT
68  DATA  6,0,@,%,1,0,1,8
80  PRINT D$;"PR#0": PRINT D$;"IN#0"
82  HOME : INVERSE
84  PRINT "          BERT-INTERFACE-BETRIEB          ";
90  PRINT "           START BASIC-PROGRAMM           "
91  NORMAL
92  POKE 34,2: REM BILDSCHIRMFENSTER
```

Bild 2.6: Das Initialisierungs-Programm INIT
a) für den Commodore C64 bzw. VC20;
b) für den IBM-PC;
c) für den Apple II. SSC bedeutet: Super Serial Card.

Dieser Befehl kann von dem in BERT aktiven Systemprogramm interpretiert werden. Er versetzt BERT in den sogenannten *Kommando-Modus* (vgl. *Kapitel 9*), in dem BERT nun als Interface für unsere Experimentier-Schnittstelle arbeitet. Die Initialisierung ist damit abgeschlossen, und nach der Bildschirm-Meldung

START BASIC-PROGRAMM

beginnt die Ausführung des eigentlichen Steuerprogramms ab Zeile 100.
Im Kommando-Modus kennt BERT eine ganze Reihe von teilweise sehr leistungsfähigen Befehlen, die wir im folgenden auch häufig als *Kommando-Routinen* oder *BERT-Kommandos* bezeichnen. Sie erleichtern das Arbeiten mit den Eingabe- und Ausgabe-Ports von BERT und unterstützen auch kompliziertere und zeitkritische Aufgabenstellungen. Wir werden darauf bereits im nächsten Kapitel ausführlich eingehen. Hinter diesen für eine normale Interfaceplatine eher ungewöhnlichen Fähigkeiten steckt natürlich die Tatsache, daß BERT ein Computer mit einem leistungsfähigen Systemprogramm ist. Eine Liste aller BERT-Kommandos findet man im Anhang. Sie werden in den folgenden Kapiteln bei der jeweiligen Aufgabenstellung vorgestellt und dabei mit Programm-Beispielen ausführlich erläutert.
Der Initialisierungsteil bleibt bei allen Steuerprogrammen gleich. Es empfiehlt sich daher, die Zeilen 10 bis 90 bzw. 92 nach dem Eintippen in den Rechner sofort auf Diskette oder Kassette abzuspeichern. Wenn man eine neue Aufgabe für BERT programmieren will, kann man so diese gesamte Initialisierung gemäß *Bild 2.6* in den Rechner laden und muß dann nur noch das eigentliche Steuerungsprogramm ab Zeile 100 anschließen.

Die BERT-Kommandos für Ein- und Ausgabe

Wir wollen jetzt unser Programm EINAUS von *Bild 2.4* für die Benutzung der in *Bild 2.5* gezeigten

8. BERT ist ein eigenständiger Computer

Schon mehrfach klang in den vorigen Kapiteln die Fähigkeit unserer Interfaceplatine BERT an, gewisse Dinge eigenständig zu erledigen. An dieser Stelle sei zum Beispiel an die komplette Ablaufsteuerung des Analog/Digital-Wandlers (*Kapitel 5*) oder das Anzeigen- und Tastatur-Multiplexen (*Kapitel 6*) erinnert.
Solche Dinge gehen in aller Regel über die Leistungsfähigkeit „normaler" Interfaceplatinen hinaus. Und besonders groß ist die Wandlungsfähigkeit der Interfaceplatine BERT: eben noch Ereigniszähler, im nächsten Moment Schrittmotortreiber, dann wieder Frequenzgeber, usw.
Hinter dieser Vielseitigkeit der Interfaceplatine BERT steckt das Konzept, nicht einen Interface-Baustein einzusetzen, sondern einen kompletten Computer. *BERT ist ein eigenständiger Computer.* Nur so ist es möglich, ein derart universelles Instrument zu erhalten. Dieses Konzept ist in der Computertechnik jedoch gar nicht so selten. Bei den meisten Computern ist z. B. die Tastatur nicht direkt über den Bus an den Mikroprozessor angeschlossen. Die Geschwindigkeits-Einbuße durch das zeilen- oder spaltenweise Abfragen der Tastatur und die nachfolgende Codewandlung wäre zu groß. Auch das automatische Wiederholen eines Zeichens nach einem längeren Tastendruck wäre für den zentralen Mikroprozessor eine unnötige Last. Daher sind viele Computer mit einem weiteren Mikroprozessor ausgestattet, dessen alleinige Aufgabe es ist, die Tastatur zu überwachen und die Zeichen fein säuberlich in einem Register zu übergeben.
Ein Mikroprozessor allein kann jedoch die Abfrage der Tastatur noch nicht bewerkstelligen; er braucht dazu ein Programm, einen Datenspeicher sowie Eingabe- und Ausgabe-Bausteine. Bei einer Reihe von Mikroprozessoren wurde dies alles in einem einzigen Schaltkreis untergebracht. Da diese Schaltkreise somit alle Merkmale eines Computers besitzen, nennt man sie (im Gegensatz zu den Mikroprozessoren) *Mikrocontroller* oder *Single-Chip-Computer* oder auch *Ein-Chip-Computer*.
Solche Mikrocontroller werden auch meist in Druckern eingesetzt, um etwa durch einen Zeichenpuffer die Datenübertragung zu entlasten oder eigenständig eine Kontrolle über Formatierungs-Angelegenheiten auszuüben.

Ein Blick auf BERT

Einen Mikrocontroller haben wir auch bei BERT eingesetzt. Wie Sie allerdings mit einem Blick auf die Platine (*Bild 8.1*) leicht feststellen können, ist es mit einem einzigen Schaltkreis trotzdem nicht getan. Damit Sie mit BERT besser vertraut werden, betrachten wir zunächst den Aufbau der Platine. Gleichzeitig werden wir damit ein wenig den Blick für die Funktions-Einheiten eines Computers im allgemeinen schärfen.
Fangen wir links oben in *Bild 8.1* an. Hinter dem Versorgungsspannungs-Eingang befinden sich ein Brückengleichrichter, ein Siebkondensator und ein Spannungsregler mit Kühlkörper. Diese Bauelemente machen die Baugruppe *Stromversorgung* aus.
Rechts daran anschließend finden wir den *Mikrocontroller*. Er enthält, wie schon oben gesagt, nicht nur den Mikroprozessor, sondern auch einen Festwertspeicher, einen Schreib-/Lese-Speicher und Ein-/Ausgabe-Einheiten. Damit Sie sich einen Begriff von der Funktionsvielfalt dieses Schaltkreises machen können, wollen wir kurz aufzählen:

- seine *Zentraleinheit* verarbeitet 8-Bit-, teilweise auch 16-Bit-Daten;
- sein *Schreib-/Lese-Speicher (RAM)* umfaßt 128 Bytes; das ist nicht gerade viel, jedoch ausreichend für alle Aufgaben des Betriebssystems.

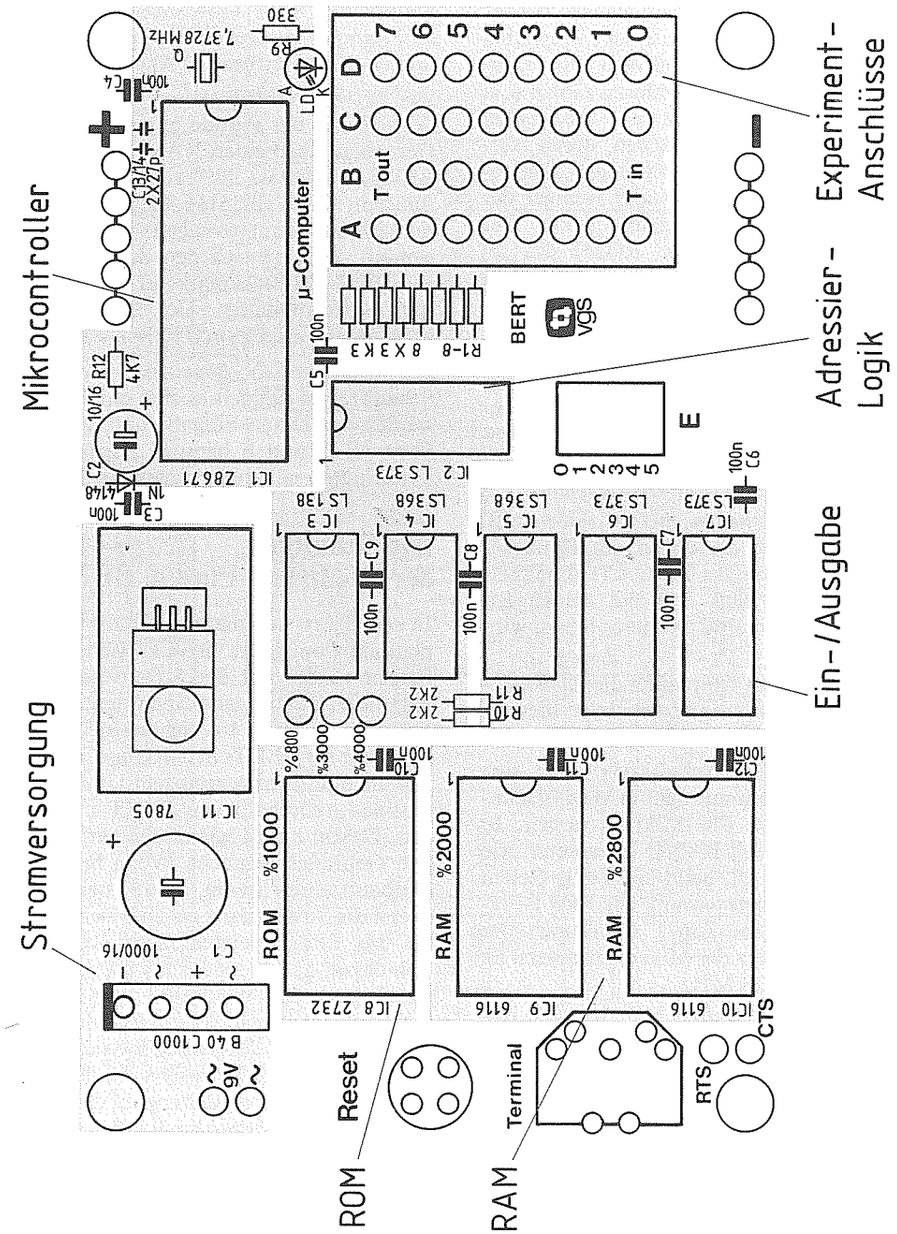


Bild 8.1: Ein Blick auf die BERT-Platine. Nähere Erläuterung im Text. Vgl. auch das Foto in *Bild 1.12.*

– sein *BASIC-Interpreter:* heute noch eine Besonderheit ist das Vorhandensein eines kompletten Hochsprachen-Interpreters auf dem Mikrocontroller. (Mit Hochsprachen – hierzu zählen u. a. BASIC, FORTRAN und Pascal – bezeichnet man solche Programmiersprachen, deren Aufbau und Regeln von der Maschinensprache relativ weit entfernt und mehr oder weniger der menschlichen Sprache angelehnt sind). Der Festwertspeicher (1 K Byte ROM) des im BERT eingebauten Mikrocontrollers enthält also, wie gesagt, einen BASIC-Interpreter. Dieser zählt zwar nicht zu den umfangreichsten, jedoch läßt sich mit ihm komfortabel die Steuerungstechnik bearbeiten. Außerdem hat er den unbestreitbaren Vorteil, daß wir beim Programmieren die uns vertrauten Methoden und Programme einsetzen können und nicht auf Maschinensprach-Kenntnisse angewiesen sind.
– *die Ein-/Ausgabe-Einheiten:* hierzu zählen z. B. die Ports A und B, die direkt am Prozessor untergebracht sind. Dabei soll noch einmal herausgestellt werden, daß dies die beiden leistungsfähigsten Ports sind. Näheres hierzu wird später erläutert.
– Weiterhin ist auf dem Chip auch die *Kommunikation mit dem Bediener* untergebracht; hiervon mehr in *Kapitel 10*.

Dieser Hexenkünstler an Schaltkreis nennt sich Z8, nicht zu verwechseln mit dem in vielen Heimcomputern eingesetzten Z80-Mikroprozessor. In der speziellen Version mit BASIC-Interpreter, wie er in BERT eingebaut ist, heißt er Z8671. Gehen wir weiter in unserer Betrachtung der BERT-Platine. Rechts neben dem Z8671 steht noch der *Schwing-Quarz*, das Herz des Mikrocomputers. Er schwingt hier mit 7,3728 MHz.

Etwa in der Mitte der Platine befinden sich insgesamt drei TTL-Schaltkreise, die die *Adressier-Logik* des BERT bilden (auch hierzu später mehr). Sie sind notwendig, weil der Z8 in BERT zusätzlich mit externem Festwertspeicher und Schreib-/Lese-Speicher sowie mit Ein-/Ausgabe-Bausteinen ausgestattet ist.

Der *externe Festwertspeicher (ROM)* befindet sich in dem Schaltkreis direkt unterhalb der Stromversorgung. Er umfaßt vier K Byte und enthält alle die BERT-Kommando-Routinen, die wir bislang benutzt haben, und noch einige mehr, die wir in Kürze besprechen werden. Als Baustein wird hier ein EPROM 2732 oder 2732A eingesetzt. Die Abkürzung EPROM bedeutet Erasable Programmable Read Only Memory. Der Baustein läßt sich somit löschen und wieder neu programmieren. Mehr darüber in *Kapitel 9*.

Der *externe Schreib-/Lesespeicher (RAM)* befindet sich auf der Platine unterhalb des EPROMs. Der obere der beiden RAM-Steckplätze ist beim BERT, so wie Sie ihn kaufen können, mit einem Baustein des Typs 6116 bestückt; in ihm lassen sich 2 K Bytes unterbringen. Wem dies noch nicht genügt, der mag sich den daran anschließenden zweiten RAM-Steckplatz mit einem weiteren Baustein 6116 ausstatten. Der Schreib-/Lesespeicher ist dann auf insgesamt 4 K Bytes ausgebaut.

Die *Ein-/Ausgabe-Bausteine:* bei diesen drei Schaltkreisen handelt es sich um TTL-Gatter. Sie werden zur Daten-Eingabe und -Ausgabe an den Ports C, D und E benutzt. Auch hierzu werden weiter unten nähere Erklärungen folgen.

Die Schaltung der Computer-Platine BERT

In *Bild 8.2* ist der gesamte Schaltplan unseres Einplatinen-Computers BERT wiedergegeben. Wir fangen wieder mit der Stromversorgung an. Sie hat die Aufgabe, eine Spannung von +5 V bei Belastungen bis zu 1 A sicherzustellen. Das genügt nicht nur für BERTs Schaltkreise, sondern es können auch noch in begrenztem Umfang Peripherie-Platinen außerhalb von BERT mitversorgt werden. Davon haben wir in den vergangenen Kapiteln Gebrauch gemacht. Wenn Sie jedoch eigene Aufbauten realisieren, dann stellen Sie immer zuerst die Strombilanz auf, um sicherzugehen, daß Sie das BERT-Netzteil nicht überlasten. Ergibt sich dabei, daß Sie mehr als 700 mA verbrauchen (der Eigenbedarf von BERT liegt bei 300 mA), so sollten Sie ein zusätzliches Netzteil bemühen, z. B. DPS 6 aus dem DIGIPROB-System zur *Einführung in die Digitalelektronik*. In diesem Fall werden nur die Masse-Leitungen verbunden, damit ein gemeinsames Bezugspotential vorliegt.
Doch zurück zur BERT-Stromversorgung. Die an die beiden Eingangsstifte anzulegene Speisespannung sollte zwischen 6,5 V und 9 V liegen und natürlich ebenfalls mindestens 1 A abgeben können. Geeignete Steckernetzteile sind im Elektronikhandel leicht erhältlich. Die Speisespannung für BERT kann Wechselspannung sein, denn auf den Anschluß folgt auf der Platine direkt ein Brückengleichrichter zum Erzeugen einer Gleichspannung.

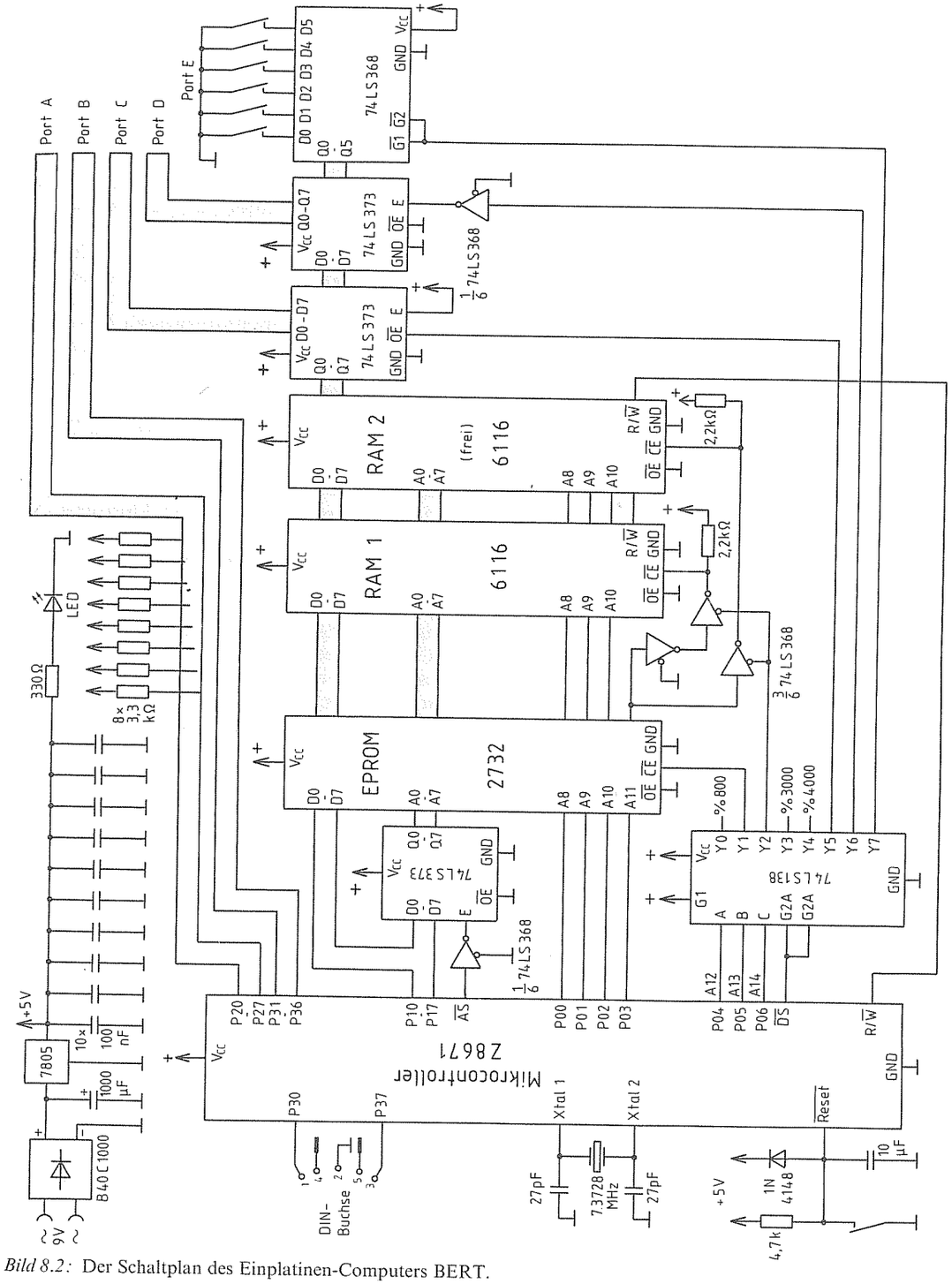


Bild 8.2: Der Schaltplan des Einplatinen-Computers BERT.

Diese pulsiert jedoch noch mit 100 Hz, weil sie sich aus allen gleichgerichteten Halbwellen der 50-Hz-Wechselspannung zusammensetzt. Daher wird als nächstes ein Ladekondensator (1000 Mikrofarad) gespeist, der ein Absinken der Gleichspannung zwischen den Halbwellen, abhängig von der Last, mehr oder weniger verhindert.
Dies ist die Ausgangs-Situation für die eigentliche Konstantspannungs-Erzeugung. Glücklicherweise benötigen wir nur eine einzige Spannung, da die Speicherbausteine und der Mikrocontroller in ihrer Betriebsspannung an die TTL-Gatter angepaßt sind. (Dies war nicht immer so; die ersten Speicherbausteine und Mikroprozessoren benötigten oft drei Spannungen: −5 V, +5 V und +12 V.)
Der zulässige Spannungsbereich der bei BERT verwendeten Schaltkreise liegt zwischen 4,75 V und 5,25 V. Diese Bedingung läßt sich leicht mit dem integrierten Spannungsregler 7805 erfüllen. Im Schaltplan rechts neben ihm ist eine ganze Reihe von keramischen Kondensatoren mit jeweils 100 nF dargestellt. Auf der Platine sitzen sie allerdings nicht so in Reih und Glied wie im Schaltplan, sondern jeder Kondensator ist einem IC zugeordnet.
Um den Grund für diesen großen Aufwand zu erläutern, müssen wir ein wenig ausholen. Ein Mikrocomputer-System ist keine statische Schaltung, wie man sie bei den ersten Experimenten der Digitalelektronik aufbaut. Auch wenn sich scheinbar nichts rührt, weil der Computer z. B. auf eine Eingabe wartet, findet trotzdem ein emsiges Treiben statt. In diesem Fall muß BERT ja ständig seine Eingabe-Leitungen überprüfen. Dazu sind einige Zeilen Programm abzuarbeiten. Diese werden aber auch immer wieder nacheinander aus dem Speicher geholt und ausgeführt. Man kann sich leicht klarmachen, daß die Daten- und Adreß-Leitungen dabei ständig ihren Zustand wechseln, und das mit recht hoher Frequenz. Wie wir schon zuvor gesagt haben, schwingt der Quarz des Computers BERT mit über 7 MHz und bestimmt damit die Geschwindigkeit der Arbeitsschritte. All die eben genannten Aktivitäten des Prozessors führen zu entsprechend schnellen Lastwechseln auf der Versorgungsleitung; bei dieser hohen Frequenz wirken schon die kurzen Leiterbahnwege als Induktivitäten, so daß es zu beachtlichen Spannungsspitzen kommen kann. Diese müssen nach Möglichkeit gleich am Entstehungsort abgebaut werden. Daher ist jeder der sogenannten „Siebkondensatoren" auch möglichst nahe an dem ihm zugeordneten Schaltkreis untergebracht. Den Abschluß der Stromversorgung bildet eine Leuchtdiode mit Vorschaltwiderstand als Betriebsanzeige.
Die nächste Baugruppe auf der BERT-Platine ist der *Mikrocontroller* (Single-Chip-Computer) Z8671 mit seiner Beschaltung. Die Beschreibung des Computerbausteins werden wir später bei der Diskussion der Software geben. Wir konzentrieren uns hier auf die Außenbeschaltung.
Da ist der schon mehrfach erwähnte Schwingquarz zu nennen. Er wird direkt an den Computerbaustein angeschlossen. Auch dies zeugt von dessen Komplexität, weil die meisten Mikroprozessoren noch einen externen Oszillator mit Verstärker benötigen. Hier jedoch kommen als einzige Zusatzbeschaltung noch zwei kleine Kondensatoren hinzu, die das Anschwingen des Quarzes erleichtern.
Der Reset-Eingang des Z8671 ist mit einem RC-Glied beschaltet. Dadurch erfolgt der Spannungsanstieg an diesem Eingang beim Einstecken der Versorgungsspannung um etwa 50 ms verzögert gegenüber dem Versorgungseingang V_{cc}. Diese Zeit genügt für das Anschwingen des Oszillators und die Betriebsaufnahme des Computers. Dieser erkennt vorrangig das L-Signal am Reset-Eingang und startet damit ein diesem Signal fest zugeordnetes Programm im Festwertspeicher. Im Verlaufe dieses Programms wird die Speicher-Ausstattung von BERT geprüft und eine Reihe von Registern auf bestimmte Werte vorgesetzt. Der Reset-Vorgang kann auch von Hand über eine Taste ausgelöst werden. Damit können Sie zu jedem beliebigen Zeitpunkt einen Betriebsprogramm-Neustart bewirken. Damit beim Abschalten der Versorgungsspannung die im Kondensator gespeicherte Energie nicht den Reset-Eingang des Computers zerstört, wird der Kondensator dann über eine Diode entladen.
Zu den wichtigen Dingen, die beim Betriebsprogramm-Neustart festgelegt werden, zählt die Verwendung der vier 8 Bit breiten Ports des Z8-Computers. (Diese Ports sind nicht zu verwechseln mit den Ihnen schon bekannten Experimentier-Ports A, B, C und D; vgl. *Kapitel 2ff.*). Je nach Einsatz des Computerbausteins Z8 sind verschiedene Konfigurationen möglich. In dem bei BERT benutzten Neustart-Programm wird festgelegt, daß Port 0 (Anschlüsse P00–P07) und Port 1 (P10–P17) den Adreß- und Datenbus des Computers bilden. Port 2 (P20–P27) allerdings ist nichts anderes als der unter dem Namen Port A auf die Steckstifte geführte Experiment-Anschluß. Die acht Leitungen dieses Ports sind über sog. pull-up-Widerstände an den Pluspol der Versorgungsspannung gelegt. Auf diese Weise sind unbeschaltete Eingänge immer auf einem definierten H-Pegel.
Port 3 (P30–P37) des Z8671 wird auf der BERT-Platine aufgeteilt: Das erste und das letzte Bit (P30 und P37) sind auf die fünfpolige DIN-Buchse geführt. Über diese Leitungen ist BERT an die V24-Schnittstelle Ihres PCs angeschlossen. Die verbleibenden sechs Leitungen (P31–P36) bilden den Port B des Experiment-Anschlusses. Nun wird auch sichtbar, warum dieser Port nur 6 Bits umfaßt.
Wenden wir uns nun der *Adressier-Logik* zu. Das Bussystem des Z8 umfaßt acht Daten- und sechzehn Adreßbits. Da diese jedoch nicht alle an Port 0 und 1 Platz haben, wird, wie bei vielen anderen Mikroprozessoren auch, ein *Multiplex* zwischen den acht Datenbits und den untersten acht Adreßbits durchgeführt. Dies bedeutet, daß bei jedem Speicherzugriff des Z8 an Port 1 zuerst das niederwertige Byte der Adreß-Information und anschließend die Daten-Information anliegt. Zu welchen Zeitpunkten dies geschieht, wird durch die zwei Leitungen $\overline{AS}$ (Adreß-Strobe) und $\overline{DS}$ (Daten-Strobe) angezeigt. Das *Bild 8.3a* zeigt die Situation beim Schreiben von Daten, wenn der Z8671 z. B. Daten im Schreib-/Lese-Speicher ablegen will. Beim Lesen (siehe *Bild 8.3b*) erscheinen die Daten gegenüber der Darstellung in *Bild 8.3a* etwas verzögert, weil sie vom Speicher- oder Eingabe-Baustein erst auf Anforderung durch $\overline{DS}$ geliefert werden.
Weil die von uns verwendeten Speicherbausteine die Daten und Adressen gleichzeitig benötigen, dient der TTL-Baustein 74LS373 zum Zwischen-

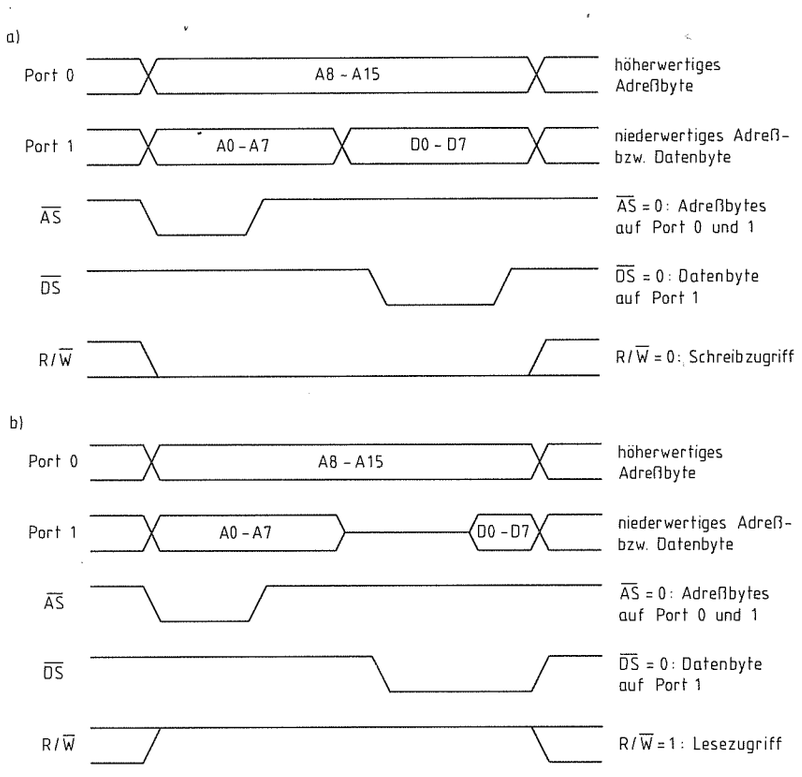


Bild 8.3: Das Multiplexen von Adreß- und Datenleitungen beim Prozessor Z8671. Die Signale $\overline{AS}$ und $\overline{DS}$ geben an, wann ein Adreß- und wann ein Daten-Byte auf Port 1 des Prozessors liegt.
a) Schreib-Zugriff;
b) Lese-Zugriff.

Eingänge				Ausgänge							
				frei	EPROM	RAM	frei	frei	Port C	Port D	Port E
$\overline{DS}$	A14	A13	A12								
G2	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
1	X	X	X	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1
0	1	1	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	0

Bild 8.4: Die Wahrheitstabelle des (1-aus-8)-Decoders 74LS138. Die Ausgänge (jeweils einer geht auf L-Pegel) dienen zum Ansteuern der verschiedenen Seiten des Adreßraums von BERT (siehe Text).

speichern der Adressen. Der Speicherimpuls wird durch Invertieren des $\overline{AS}$-Signals gewonnen (vgl. hierzu noch einmal *Bild 8.2*).
Die Adreßleitungen A0 bis A10 sind direkt bzw. über den erwähnten Zwischenspeicher an die Speicherbausteine geführt. Die Adreßbits mit noch höherer Wertigkeit dienen dann der Unterscheidung der einzelnen Baugruppen. Die drei Adreßleitungen A12, A13 und A14 sind dazu auf einen Binärdecoder (8 aus 3) geführt (zu A11 siehe unten). Jeder der acht möglichen Bit-Kombinationen an seinen drei Eingängen A, B und C ist genau ein Ausgang zugeordnet. Synchron zur Steuerleitung $\overline{DS}$ geht also jeweils *ein* Ausgang auf L-Pegel, während alle übrigen auf H-Pegel verbleiben. Die Wahrheitstabelle in *Bild 8.4* stellt die Zusammenhänge dar.

Die Speicher-Aufteilung in BERT

Die eben beschriebene spezielle Struktur der Adressier-Logik erlaubt es uns, von einer Seiten-Einteilung des Speichers zu sprechen. Ähnlich wie ein Buch in Seiten eingeteilt ist, so ist es auch der Adreßraum des BERT. Die Seiten sind einheitlich 4 K Bytes groß. Zu den folgenden Abschnitten vergleichen Sie auch *Bild 9.3*.

Die *erste Seite* von %0000 bis %0FFF ist dem internen Speicher des Prozessors, seinen Registern und Ein-/Ausgabe-Leitungen zugeordnet. Allerdings belegt der Z8 nur die Hälfte davon, nämlich die Adressen %0000 bis %07FF. Wir benutzen hier übrigens in Übereinstimmung mit der Syntax des Z8-BASIC-Interpreters ein vorangestelltes Prozentzeichen (%), um eine *Hexadezimal-Zahl* zu symbolisieren; vgl. auch *Kapitel 4*.

Die *zweite Seite* (%1000 bis %1FFF) beinhaltet den externen Festwertspeicher (ROM) mit den Ihnen schon bekannten Kommando-Routinen und anderen Hilfsroutinen, die wir später besprechen (vgl. *Kapitel 9*).

Die *dritte Seite* des BERT-Speichers nimmt den Schreib-/Lese-Speicher (RAM) auf. In der Grundausstattung ist diese Seite mit einem 6116-Baustein nur halb belegt, und zwar von Adresse %2000 bis %27FF. Das nachzurüstende RAM belegt dann die Adressen von %2800 bis %2FFF. Die Unterscheidung der beiden RAM-Bausteine erfolgt in drei Tri-State-Gattern des Bausteins 74LS368 durch Verknüpfung der Adreßleitung A11 mit dem Seitenauswahl-Signal Y2 (Decoder-Ausgang, siehe oben). Dabei wird mit dem Signal Y2 = 0 die dritte Seite des Speichers, also der gesamte RAM-Bereich von %2000 bis %2FFF, angewählt; mit dem Zustand von A11 wird die erste Hälfte der Seite (ab %2000) bzw. die zweite (ab %2800) angesteuert.

Die *vierte und fünfte Seite* sind bei BERT unbenutzt. Für Kenner der Materie sind jedoch die Seitenauswahl-Signale Y3 und Y4 auf Lötpunkte geführt, so daß auch dieser Adreßraum genutzt werden kann. Diese Punkte sind mit %3000 und %4000 beschriftet. Entsprechendes gilt auch bei der *ersten* Seite für die zweite Hälfte, die vom Prozessor ja nicht benutzt wird. Adressen im Bereich von %0800 bis %0FFF aktivieren mit einem L-Pegel die Leitung an dem mit %0800 beschrifteten Lötpunkt.

Die Adreßauswahl-Leitung Y5 der *sechsten Seite* führt zu dem TTL-Gatter 74LS373, einem achtfach-D-Flipflop, das mit Port C verbunden ist. Wenn das Seitenauswahl-Signal auf low geht, werden die 8 Ausgangsverstärker des Bausteins 74LS373 aktiviert und damit die Daten des Ports C auf den Datenbus gelegt (Ausgänge Q0 bis Q7). Port C ist also – wie Sie aus den vorigen Kapiteln schon wissen – ein reiner Eingabe-Port.
Der Adreßbus ist *nicht* zum Baustein 74LS373 des C-Ports geführt, sondern nur das Seitenauswahl-Signal. Offensichtlich wird das Gatter angesprochen, unabhängig davon, welche Adresse auf den unteren Adreßleitungen A0 bis A11 anliegt. In den

Adreßraum übersetzt bedeutet dies: Die Daten des Ports C können unter jeglicher Adresse zwischen %5000 und %5FFF einschließlich erhalten werden. Auf den ersten Blick scheint hier Adreßraum verschwendet, weil ein ganzer 4-K-Adreßbereich für das Einlesen einer einzigen 8-Bit-Information benutzt wird. Andererseits bietet sich diese Lösung aber an, wenn nicht noch zusätzlicher Schaltungsaufwand getrieben werden soll. Dieser Schaltungsentwurf ist in Mikrocomputer-Systemen recht häufig anzutreffen und wird *unvollständige Adressierung* genannt.
Nach dem gleichen Prinzip geschieht die Adressierung von Port D (*siebente* Seite: von %6000 bis %6FFF) und Port E (*achte* Seite: von %7000 bis %7FFF). Dabei ist Port E (wie Port C) ein reiner Eingabe-Port, aber Port D ein Ausgabe-Port. Der Datenbus (D0 bis D7) ist an die Eingänge des für Port D zuständigen Bausteins 74LS373 gelegt, und seine Ausgänge führen an die Lötstifte des Experiment-Anschlusses.
Mit der unvollständigen Adressierung kommt es allerdings noch schlimmer! Der Z8 kann insgesamt 64 K Bytes adressieren und besitzt daher auch noch eine sechzehnte Adreßleitung (A15). Diese ist jedoch im Schaltungsaufbau von BERT überhaupt nicht verwendet. Demzufolge erscheinen alle oben besprochenen Adressen (bis auf die internen des Prozessors) auch dann, wenn zu jeder Adresse der Wert %8000 addiert wird. Der Schreib-/Lese-Speicher (%2000–%27FF) kann also auch genausogut mit Adressen zwischen %A000 und %A7FF angesprochen werden. Von dieser Tatsache wird Gebrauch gemacht: Nach jedem Reset liest das Betriebssystem des Z8 das auf der Adresse %FFFD befindliche Byte ein, um daraus die Geschwindigkeit der Datenübertragung zu ermitteln. Wir haben oben gesehen, daß einerseits wegen der unvollständigen Adreß-Decodierung die gesamte achte Seite mit dem Port E belegt ist; und wir wissen nun andererseits, daß sie nicht nur im Adreßbereich %7000 bis %7FFF, sondern auch unter %F000 bis %FFFF erreichbar ist. Also wird nach einem Reset der Port E mit den Schiebeschaltern gelesen, was ja durchaus in unserer Absicht liegt.

Die Programmierung von BERT

Wenn BERT schon ein vollständiger Computer ist, dann wollen wir ihn auch programmieren. Daß er BASIC „spricht", gibt uns die Zuversicht, mit den uns vertrauten Methoden zurechtzukommen. Vergleichen wir BERT mit unserem PC.
Beim PC ist die Situation ganz einfach: Er wird über die Tastatur programmiert. Wenn wir einen Buchstaben oder ein sonstiges Zeichen der Tastatur anschlagen, so wird eine entsprechende Information irgendwo im Arbeitsspeicher des Computers abgelegt. Meist stellt das Betriebssystem dieses Zeichen auch sofort auf dem Bildschirm dar. Die genauen Abläufe hängen von dem gerade aktiven Betriebs- oder Anwender-Programm ab.
BERT besitzt jedoch keine Tastatur und keinen Bildschirm. Damit steht er jedoch keineswegs allein da. Die meisten Computer, d.h. nahezu alle Minicomputer und alle Großrechenanlagen, haben Bildschirm und Tastatur ebenfalls nicht direkt angeschlossen. Meist werden an diese Computer eine oder mehrere Daten-Endeinrichtungen angeschlossen. Diese sogenannten *Terminals* besitzen einen eigenen Prozessor und damit eine eigenständige „Verwaltung" ihres Bildschirms und ihrer Tastatur und sind mit dem eigentlichen Computer über eine *Daten*-Leitung verbunden.

Der PC als Terminal für BERT

Genau dasselbe werden wir auch bei BERT vorsehen. Jedoch brauchen Sie kein Terminal für ihn zu kaufen. Ihr PC, mit dem Sie die Beispiele und Übungen der vorangegangenen Kapitel programmiert haben, wird jetzt in ein Terminal umfunktioniert.
Dies ist ein entscheidender Einschnitt in die Ihnen bisher bekannte Arbeitsweise. Wer noch keine Übung in solch einem Terminalbetrieb hat, könnte vielleicht etwas ins Schwimmen geraten, wenn er nicht mehr weiß, auf welchem Computer er gerade rechnet. Daher wollen wir hier die Arbeitsweisen vergleichend gegenüberstellen:

Bisher:

Der PC übernimmt:
- Die Eingabe der Programme per Tastatur
- Das Ausdrucken der Programme auf dem Bildschirm
- Die Ausführung der Programme
- Die Änderung der Programme
- Das Abspeichern der Programme auf Kassette/Diskette
- Das Laden der Programme von Kassette/Diskette

BERT übernimmt:
- Die Ausführung der steuerungstechnischen Ein- und Ausgabe

Nachher:

Der PC übernimmt:
- Das Senden der Tastaturzeichen an BERT
- Das Darstellen der von BERT empfangenen Zeichen auf dem Bildschirm

BERT übernimmt:
- Die Eingabe der Programme vom Terminal
- Das Ausdrucken der Programme auf dem Terminal
- Die Ausführung der Programme
- Die Änderung der Programme
- Die Ausführung der steuerungstechnischen Ein- und Ausgabe

Wer genau mitgezählt hat, wird feststellen, daß zwei Funktionen in der Nachher-Liste nicht enthalten waren:

- Das Abspeichern der Programme auf Kassette/Diskette
- Das Laden der Programme von Kassette/Diskette

Auf diese Punkte kommen wir später zu sprechen.

Betrachten wir zunächst das Flußdiagramm in *Bild 8.5*. Es definiert den Kern eines Terminalprogramms auf dem PC. Alles, was später dazukommt, mag vielleicht ein Mehrfaches an Programmtext erfordern, liefert jedoch nur Nebenfunktionen zum Erhöhen des Bedienungskomforts. Daher halten wir es für wichtig, zunächst einmal mit diesem einfachen Programm zu arbeiten.
Die erste Aktion des Terminalprogramms ist die Initialisierung des Übertragungskanals. An dieser Stelle sind Datenübertragungs-Rate und Datenformat festzulegen. Wir haben die entsprechenden Entscheidungen schon für Sie getroffen und werden sie in *Kapitel 10* noch erläutern. Die Datenübertragungs-Rate wird auf 300 Bits pro Sekunde eingestellt und das Format auf 8 Bits je Wort, ohne Paritätsbit und mit einem Stopbit.
Die gleiche Datenübertragungs-Rate muß natürlich auch an BERT eingestellt werden, d.h. die Schiebeschalter 0, 1 und 2 des Ports E müssen auf ON stehen. Die übrigen Schiebeschalter spielen im Moment keine Rolle.
Wie schon beim Initialisierungs-Programm in *Kapitel 2* wird nun der Benutzer aufgefordert, die Reset-Taste an BERT zu drücken. Das führt dazu, daß ein eventuell gerade in BERT laufendes Programm abgebrochen wird. Das Drücken der Reset-Taste wird vom Terminalprogramm am Auftreten eines Doppelpunktes auf der Übertragungsleitung erkannt. BERT sendet ihn als sein Bereit-Zeichen (sog. Prompt). Anschließend wird vom PC noch ein „Escape"-Zeichen gesendet, um ein erneutes Bereit-Zeichen auszulösen. Damit sind alle Vorbereitungen abgeschlossen, und das Terminalprogramm startet mit der Datenübertragungs-Schleife.

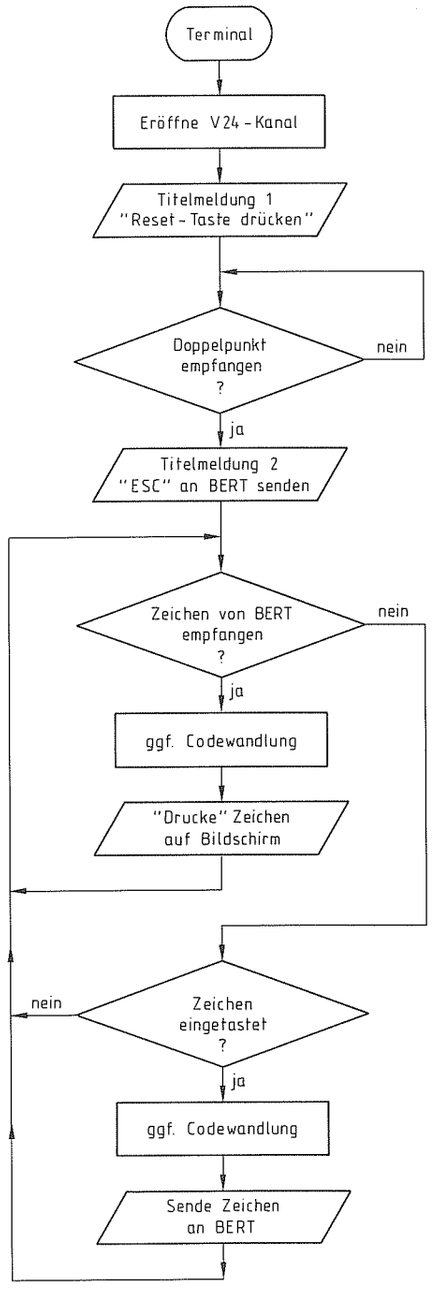


Bild 8.5: Das Flußdiagramm des kleinen Terminalprogramms.

Als erstes wird geprüft, ob von BERT ein Zeichen anliegt. Wenn ja, wird es auf dem Bildschirm „ausgedruckt". Bei einigen Zeichen muß man jedoch aufpassen und darf sie nicht direkt verwerten. Wie so oft ergeben sich Unterschiede zwischen den Zeichensätzen der einzelnen Computer, insbesondere in der Verwendung von Steuerzeichen. Durch Abfragen im Terminalprogramm werden diese Zeichen ermittelt und gegebenenfalls unterdrückt oder umgewandelt. Anschließend erfolgt der Rücksprung zum Überprüfen des Dateneingangs von BERT (Raute: „Zeichen empfangen?").
Falls dort kein Zeichen anliegt, wird die Tastatur abgefragt (Raute: „Zeichen eingetastet?"). Liegt auch dort kein Zeichen an, so wird erneut zum Überprüfen des Dateneingangs von BERT zurückgesprungen. Falls jedoch ein Tastaturzeichen anliegt, muß dieses auch wieder ein Filter von Abfragen durchlaufen. Bei dieser Gelegenheit können speziell vereinbarte Steuerzeichen erkannt werden, die besondere Funktionen im Terminalprogramm auslösen sollen; ein größeres Terminalprogramm mit einigen komfortablen Besonderheiten wird in *Kapitel 9* vorgestellt. Alle anderen Zeichen (also die „normalen", außer den Steuerzeichen) sollen jedoch an BERT geschickt werden. Anschließend geht es wieder zurück an den Anfang der Abfrageschleife, ob Daten von BERT anliegen.
Es fällt auf, daß das Programm nach dem Durchlaufen der Initialisierung in zwei Hälften eingeteilt werden kann (siehe *Bild 8.6*):

- Überprüfen des Dateneingangs und gegebenenfalls Darstellung auf dem Bildschirm
- Überprüfen der Tastatur und gegebenenfalls Senden auf dem Datenausgang.

Das Programm ist allerdings nicht strikt symmetrisch: Der Dateneingang, der von BERT kommt, wird in der Abfrage-Häufigkeit bevorzugt. Machen Sie sich das am Flußdiagramm (*Bild 8.5*) klar: nach dem Erkennen eines Tastaturzeichens und dessen Verarbeitung wird zuerst wieder der Dateneingang von BERT überprüft, bevor auf ein neues Tastaturzeichen abgefragt wird. Anders

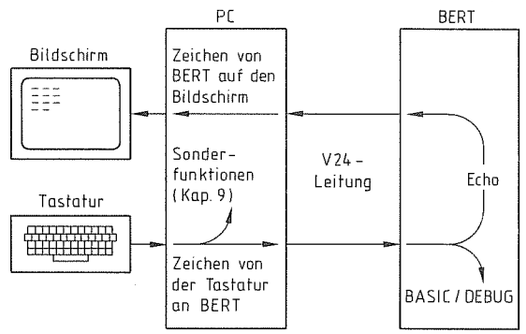


Bild 8.6: Die Funktion des Terminalprogramms: Auf der PC-Tastatur angeschlagene Zeichen gehen an BERT; Zeichen von BERT gehen auf den PC-Bildschirm, und auf diesem wird dann das Echo dargestellt.

aber beim Dateneingang von BERT: wird hier ein Zeichen empfangen, so wird sofort wieder die gleiche Leitung abgefragt, und erst, wenn kein Zeichen ankam, wird die Tastatur „bedacht". Der Grund für das Bevorzugen der BERT-Datenleitung liegt in der höheren Sendegeschwindigkeit von BERT, inbesondere beim Ausdrucken von Programmlisten. In diesem Fall folgt ein Buchstabe direkt hinter dem anderen, und das Terminalprogramm muß schnell sein, um alles mitzubekommen. In der umgekehrten Datenrichtung wird zwar jeder einzelne Buchstabe genauso schnell gesendet; es entstehen jedoch zwischen den Buchstaben immer wieder Pausen, die von unserer bescheidenen Anschlag-Geschwindigkeit auf der Tastatur herrühren.
Noch etwas ist zu beachten. Wenn wir am PC einen Buchstaben eingeben und das System „verschluckt" ihn (d.h. er kommt bei BERT nicht an), weil es gerade mit anderen Aufgaben beschäftigt ist, so ist das zwar ärgerlich, jedoch nicht weiter tragisch. Der Buchstabe kann ja noch einmal eingegeben werden. Anders mit BERT: dieser hat keinerlei Kontrolle darüber, ob der Buchstabe auch korrekt im PC angekommen ist. Diesen Unterschieden tragen wir Rechnung, indem wir möglichst häufig den Dateneingang prüfen.
In *Bild 8.7 und 8.8* sind die Terminalprogramme für den Commodore 64 bzw. für den IBM-PC eingetragen; sie folgen dem Flußdiagramm in *Bild 8.5*. Beim C64 (ebenso beim C128 und beim VC20) werden die Daten typisch über einen „Kanal" übertragen. Dieser wird in Zeile 60 mit Hilfe

des OPEN-Befehls eingerichtet. Im weiteren Programm wird in GET- und PRINT-Befehlen unter Nummer zwei (#2) dieser Datenkanal zu BERT angesprochen.
Beim Commodore 64 (ebenso beim C128 oder beim VC20) ist das Escape-Zeichen nicht auf der Tastatur zu finden. Wir verwenden statt dessen den Linkspfeil in der linken oberen Ecke. Dieser Linkspfeil wird für BERT nicht benötigt und liegt auf der C64-Tastatur an der gleichen Stelle, wo sonst üblicherweise Escape liegt. BERT benötigt auch noch den Rückstrich (backslash), der ebenfalls nicht auf der Commodore-Tastatur zu finden ist. Das Terminalprogramm setzt dafür das Pfundzeichen (£) rechts oben ein. Eine Code-Umsetzung ist hierzu nicht notwendig: Das Pfundzeichen des C64 und das Rückstrich-Zeichen von BERT sind jeweils mit %5C codiert. Daher erscheint der Rückstrich auch als Pfundzeichen am Bildschirm.
Weiter muß der Code der Löschtaste DEL, %14, in den sonst üblichen Code %08 umgewandelt werden. Dieser steht für das Steuerzeichen Backspace. Empfängt BERT diesen Code %08, so löscht er in der Folge der bereits eingegebenen Zeichen das letzte. Außerdem sendet er das Steuerzeichen %08, wie alle anderen Zeichen auch, an den C64 zurück. Damit das letzte bereits dargestellte Zeichen auch auf dem Bildschirm wieder verschwindet, wandelt das Terminalprogramm den Code %08 (Backspace) wieder in den für DEL (%14) zurück und gibt diesen aus, löscht also das letzte Zeichen auf dem Schirm.
Beim IBM-PC sind durch Laden des entsprechenden Zeichensatzes alle Zeichen erreichbar. Da die meisten unserer Leser jedoch mit dem deutschen Zeichensatz arbeiten werden, haben wir der Bequemlichkeit Rechnung getragen. Das At-Zeichen @, auch Klammeraffe genannt, wird in BERT-Programmen recht häufig benötigt. Es fehlt in der deutschen Tastaturbelegung. Das Terminalprogramm setzt daher das von BERT nicht benötigte ß in @ um. Auf dem Bildschirm erscheint es dann als Paragraph §. Das Löschzeichen des IBM-PC, der Linkspfeil in der rechten oberen Ecke der Tastatur, sendet den korrekten Code %08 an BERT; dies tut also *nicht* die mit DEL bezeichnete Taste! Wenn BERT diesen Code zurückschickt, wird das Zeichen auf dem Bildschirm jedoch nicht gelöscht. Daher druckt das Terminalprogramm in diesem

```
10 REM PROGRAMM "TERMINAL" (C64, VC20)
20 REM DIALOGBETRIEB MIT BERT
30 REM BAUDRATE 300 BD, 1 STOPBIT,
40 REM KEINE PARITAET, VOLLDUPLEX
50 REM
60 OPEN2,2,0,CHR$(6)+CHR$(1)
70 PRINT CHR$(147);CHR$(18);"          BERT-TERMINAL-BETRIEB          "
80 PRINT"     300 BAUD: PORT E AUF 7 = XXX111"
90 PRINT"   BITTE RESET-TASTE AN BERT DRUECKEN!"
100 GET#2,DU$:IF DU$<>":" THEN 100
110 PRINT CHR$(147);CHR$(18);"          BERT-TERMINAL-BETRIEB          ";
120 PRINT"          Z-8671  BASIC/DEBUG          "
130 PRINT#2,CHR$(27); : REM ESC SENDEN
140 REM UEBERTRAGUNGSSCHLEIFE
150 GET#2,A$
160 IFA$=""THEN190
170 IFA$=CHR$(8)THENA$=CHR$(20)
180 PRINTA$;:GOTO 150
190 GETB$
200 IFB$=""THEN150
210 IFB$=CHR$(95)THENB$=CHR$(27)
220 IFB$=CHR$(20)THENB$=CHR$(8)
230 PRINT#2,B$;
240 GOTO150
```

Bild 8.7: Das kleine Terminal-Programm für den Commodore C64.


```
10 REM Programm "TERMINAL"  (IBM-PC)
20 REM Dialogbetrieb mit BERT
30 REM 300 Baud, 2 Stopbits
40 REM 8 Datenbits, keine Parität
50 REM
55 CL$ = CHR$(29)+CHR$(32)+CHR$(29)
60 OPEN "COM1:300,N,8,1" AS #2
65 WIDTH 40 : CLS : KEY OFF : COLOR 0,7
70 PRINT"          BERT-TERMINAL-BETRIEB          ";
80 PRINT"     300 Bd: Port E auf 7 = XXX111     "
90 COLOR 7,0 : PRINT"   BITTE RESET-TASTE AN BERT DRUECKEN!  "
95 IF EOF(2) THEN 95
100 DU$ = INPUT$(LOC(2),#2) : IF DU$ <> ":" THEN 95
110 CLS : COLOR 0,7 : PRINT"          BERT-TERMINAL-BETRIEB          "
120 PRINT"              Z-8671 BASIC/DEBUG        "; : COLOR 7,0
130 PRINT#2,CHR$(27); : REM ESC senden
140 REM Übertragungs-Schleife
150 IF EOF(2) THEN 190
160 A$ = INPUT$(LOC(2),#2)
170 IF A$ = CHR$(8) THEN A$ = CL$
180 IF A$ <> CHR$(10) THEN PRINT A$; : GOTO 150
190 B$ = INKEY$
200 IF B$ = "" THEN 150
210 IF B$ = "ß" THEN B$ = CHR$(64)
220 PRINT#2,B$;
230 GOTO 150
```

Bild 8.8: Das kleine Terminal-Programm für den IBM-PC.

Fall die Sequenz %1D %20 %1D. Sie bewirkt, daß der Cursor um eine Position nach links gesetzt, das zu löschende Zeichen mit einem Leerzeichen überschrieben wird und der Cursor zum Schluß wieder zurückgeht.
Das Terminalprogramm für den Apple II arbeitet nach einem ganz anderen Konzept (*Bild 8.9*). Hier bietet sich die Einrichtung eines Kanals nicht so sehr an. Bei diesem Computer wartet das GET-Kommando immer, bis ein Zeichen eintrifft. Da das Terminalprogramm jedoch im Wechsel beide Datenquellen (Dateneingang und Tastatur) überprüfen muß – auch wenn kein Zeichen empfangen wurde –, werden direkt die Register des Datenübertragungs-Bausteins und der Tastatur angesprochen. Die Initialisierung im Terminalprogramm erfolgt durch einige POKE-Befehle in die Kommando- und Steuer-(Control-)Register des Schnittstellen-Bausteins. Ob Daten anliegen, kann aus dem Bitmuster des Statusregisters dieses Bausteins ersehen werden. Die Ein- und Ausgabe der Daten erfolgt hier mit PEEK- und POKE-Befehlen an die beteiligten Ein- und Ausgabe-Register.
Beim Apple II fehlt wie beim C64 der Rückstrich, bzw. er ist nur über komplizierte Tastenkombinationen erreichbar, daher die Wandlung des Rechtspfeils in den Rückstrich. Der Linkspfeil dient auch beim Apple II als Löschtaste und sendet den gewünschten Code %08. Beim Empfang des Zeichens %08 von BERT rückt der Cursor des Apple II nur unsichtbar nach links. Daher wandelt das Terminalprogramm wie beim IBM-PC ein empfangenes Zeichen %08 um in die Sequenz %08 %20 %08.
Wenn Sie nun das Terminalprogramm in Ihren Computer eingegeben haben, kommt der große Moment: Starten Sie das Programm. Der Bildschirm sollte sofort gelöscht werden, und es sollte die Aufforderung erscheinen, die Reset-Taste von BERT zu drücken. Wie zuvor erkennt das Terminalprogramm wieder die Bereitschaft von BERT durch das Erscheinen des Doppelpunktes nach dem Betriebsprogramm-Neustart (ausgelöst durch

```
10  REM PROGRAMM "TERMINAL" (APPLE II MIT SSC)
20  REM DIALOGBETRIEB MIT BERT
30  REM BAUDRATE 300 BD, 2 STOPBITS
40  REM KEIN PARITAETSBIT, VOLLDUPLEX
50  REM
51 V24 = 49320: REM SSC IN SLOT 2
52 STS = V24 + 1:KOM = V24 + 2:KTR = V24 + 3: REM V24-REGISTER
53 TAST = 49152:TST = 49168: REM TASTATURREGISTER
54  REM  MASCHINENCODE ZUM TESTEN VON BIT3 DES V24 STATUS
55  FOR I = 768 TO 776: READ MC: POKE I,MC: NEXT
56  DATA  173,169,192,41,8,168,76,1,227
57  POKE 11,0: POKE 12,3: REM  USR-FUNKTION INSTALLIEREN
58  REM  SETZE UEBERTRAGUNGSPARAMETER
59  POKE KTR,150: POKE KOM,11
60 D$ =  CHR$ (8) +  CHR$ (32)
70  HOME : INVERSE
75  PRINT "         BERT-TERMINAL-BETRIEB          "
77  NORMAL
80  PRINT "     300 BD: PORT E AUF 7 = XXX111"
90  PRINT "   BITTE RESET-TASTE AN BERT DRUECKEN!"
95  WAIT STS,8: REM STATUSREGISTER TESTEN
100  IF  PEEK (V24) <  > 58 THEN  GOTO 95
105  HOME : INVERSE
110  PRINT "         BERT-TERMINAL-BETRIEB          ";
120  PRINT "          Z-8671  BASIC/DEBUG           ";
125  NORMAL : POKE 34,2: REM TEXTFENSTER
130  POKE V24,27: REM ESC SENDEN
140  REM UEBERTRAGUNGSSCHLEIFE
150  IF  USR (0) = 0 THEN  GOTO 190
160 A =  PEEK (V24)
170  IF A = 8 THEN  PRINT D$
180  IF A <  > 10 THEN  PRINT  CHR$ (A);
190 B =  PEEK (TAST)
200  IF B < 128 THEN  GOTO 150
205  POKE TST,0:B = B - 128
210  IF B = 21 THEN B = 92
230  POKE V24,B
240  GOTO 150
```

Bild 8.9: Das kleine Terminal-Programm für den Apple II.

den Reset). Daraufhin wechselt auf dem PC-Bildschirm die Kopfzeile; er meldet jetzt:

BERT-TERMINALBETRIEB
Z8671 BASIC/DEBUG

Außerdem erscheint ein Doppelpunkt in der nächsten Zeile – das Prompt von BERT; es wurde durch das Senden eines Escape-Zeichens ausgelöst.
Bevor es weitergeht, halten Sie sich bitte nochmals die Situation vor Augen: Das Terminalprogramm ist in Betrieb und fragt emsig die Tastatur und BERT ab. Der Doppelpunkt sagt Ihnen jedoch, daß BERT bereitsteht und auf Kommandos von Ihnen wartet. Mit etwas Gewöhnung werden Sie bald Ihr Terminalprogramm vergessen und das Gefühl haben, *mit BERT* in direktem Kontakt zu stehen, auch wenn Sie immer noch an Ihrem PC sitzen.
Geben Sie nun ein kurzes Programm in BERT ein, zunächst die erste Zeile:

10 PRINT "HALLO BERT"

Nach dem Betätigen der Return-Taste muß BERT wieder einen Doppelpunkt senden, der auf dem Bildschirm des PCs erscheint. Geben Sie nun weiter ein:

20 GOTO 10

Wieder kommt nach Return ein Doppelpunkt als Quittung. Sie können sich das Programm anschauen, indem Sie das Kommando

LIST

eingeben, natürlich wie immer mit Return abgeschlossen. Die obigen beiden Zeilen 10 und 20 müssen nun erscheinen. Waren Sie verwirrt, daß nicht das Listing des Terminalprogramms erschien? Nun gehen wir aufs Ganze und starten das Programm in BERT mit

RUN

Mit der Langsamkeit eines Fernschreibers wird der Bildschirm gefüllt:

HALLO BERT
HALLO BERT
HALL...

Daß es nicht schneller geht, liegt am Datenübertragungs-Weg (V24-Schnittstelle). Die eigentliche Rechengeschwindigkeit von BERT ist nahezu die gleiche wie die des PCs.
Wie sollen wir das Programm wieder anhalten? Bitte drücken Sie *nicht* wie gewohnt die Stop- oder Break-Taste Ihres Computers. Damit wäre der Spuk zwar auch vorüber, aber nur weil das Terminalprogramm im PC nicht mehr arbeitete. BERT würde jedoch – für uns unsichtbar – munter weiter ausgeben.
Drücken Sie statt dessen die Escape-Taste bzw. die Taste, die durch das Terminalprogramm mit der Escape-Funktion belegt wurde. Beim Commodore 64 ist dies z. B. die Linkspfeiltaste (s. o.). Nun bleibt BERT ordnungsgemäß stehen mit der Meldung

0! AT XX

und zeigt mit dem Doppelpunkt anschließend erneut seine Bereitschaft an. In der Meldung *0! AT XX* steht die Null als Code für die Meldung „Programm gestoppt" (vgl. auch die Liste im Anhang), und XX ist die Nummer der Zeile, in der das Programm stehenblieb.
Nach diesem ersten Experiment ist es nun Zeit, sich systematisch mit dem BASIC von BERT zu beschäftigen.

Die Programmiersprache BASIC/DEBUG

Allgemeines

Der Name der Programmiersprache BASIC/DEBUG steht zum einen dafür, daß es sich hier um einen der vielen BASIC-Dialekte handelt. Was aber bedeutet DEBUG? Ein *bug* ist im Englischen, etwas frei übersetzt, eine Laus. Dieses Tierchen muß immer für die Programmierfehler herhalten, die ja der Programmierer selbst verursacht hat. Praktisch jedes Programm enthält zu Beginn noch Fehler. Eine „Programmierer-Weisheit" lautet daher: „Wenn ein Programm sofort läuft, dann steckt der Fehler tiefer!" Ein Programm muß also in aller Regel zunächst „entlaust" werden, d. h. den Prozeß des „Debugging" durchlaufen.
Die Entwickler der Programmiersprache BASIC/DEBUG waren nun wohl angetan von der Einfachheit, in einem BASIC-Programm das Debugging durchzuführen. Dies muß man messen an den Problemen des Debuggings etwa eines Maschinensprach-Programms, wie sie bei Mikrocontrollern sonst üblich sind. Von dieser Einfachheit der Fehlerbeseitigung rührt also der Name BASIC/DEBUG her.
Die Programmiersprache BASIC/DEBUG besteht nur aus einem Teil der Ihnen von den PCs her vertrauten BASIC-Anweisungen; mit anderen Worten: es ist eine sogenannte Untermenge der Sprache BASIC. Weil BASIC/DEBUG speziell für Steuerungszwecke entwickelt wurde, sind darin einige gewohnte BASIC-Merkmale nicht enthalten. Darunter fallen die mathematischen Funktionen, Feldvariablen, Zeichenketten (Stringvariablen) und Fließkommazahlen. Auch wurde der Befehlsvorrat soweit wie möglich gekürzt, damit der BASIC-Interpreter im Festwertspeicher des Z8 unterzubringen ist. Dabei wurden, vom „normalen" BASIC ausgehend, alle Anweisungen entfernt, die sich auch aus anderen zusammensetzen lassen. Übriggeblieben sind 15 Kommandos und 2 Funktionen, die wir später beschreiben.
Allerdings wollen wir nicht unerwähnt lassen, daß BASIC/DEBUG auf seinem Gebiet natürlich auch Stärken hat. Mit einfachen Mitteln können steuerungstechnische Programme getestet werden, weil sich alle Ports, alle Register und der Arbeitsspeicher direkt adressieren lassen. Durch wahlweises Verwenden von dezimalem oder hexadezimalem Zahlensystem kann man sehr problemnah und übersichtlich vorgehen.

Vor der Eingabe des Programms müssen Sie die Stromversorgung von BERT abschalten und das Experiment aufbauen. Die genaue Experiment-Beschreibung entnehmen Sie bitte dem *Kapitel 4*. Benötigt werden die Eingabe- und die Ausgabe-Platine. Wer sein Experiment besonders realistisch gestalten will, baut auch noch eine relaisgesteuerte Sirene auf sowie für den Türöffner das Modell mit der Motorsteuer-Platine.
Danach wird die Stromversorgung von BERT wieder eingeschaltet. Wenn Ihr Terminalprogramm im PC während der ganzen Zeit weitergelaufen war, müssen Sie nun auf dem Bildschirm einen neuen Doppelpunkt beobachten, der vom Einschalt-Reset herrührt.
Wenn anschließend das Programm in BERT korrekt eingegeben wurde, können Sie es starten und austesten. Sobald die Sirene losgeht, hilft nur ein Reset an BERT, um wieder die Kontrolle über ihn zu erhalten. Damit herrscht aber noch keine Ruhe, denn die Pegel am Port D werden durch den Reset nicht beeinflußt. Schalten Sie nun den Kommando-Modus ein (s.o.); dann können Sie mit dem Kommando *D=0* die Sirene zum Schweigen bringen.
Nun sind wir aber in einem Dilemma: wenn Reset gedrückt wird, ist bekanntlich das BERT-Programm gelöscht (siehe *Kapitel 8*), und wir müßten es für den nächsten Durchlauf wieder von der Diskette laden. Sie hatten doch hoffentlich unseren früher erteilten Rat befolgt und das Programm vor dem ersten Start auf der Diskette abgelegt?
Diejenigen, die jetzt ein schlechtes Gewissen plagt, und die sich im Geiste schon mit langer Tipparbeit abfinden, erinnern wir an dieser Stelle noch einmal an die Indirektions-Operatoren. Das Registerpaar %08 %09 des Z8 enthält immer die Anfangsadresse des BASIC-Speichers. Dort muß die erste Zeilennummer des Programms wieder eingetragen werden. In unserem Beispiel war dies die 100. Geben Sie also ein:

↑↑8 = 100

und anschließend

LIST

Erfreut werden Sie feststellen, daß Ihr Programm wie von Zauberhand wieder erschienen ist. Nehmen Sie jedoch spätestens jetzt die Gelegenheit wahr und speichern Sie es *vor* dem nächsten Probelauf auf der Diskette ab! Denn nun können Sie das Programm nicht ändern und danach wieder starten (vgl. *Seite 176*).

Drei Verbesserungs-Wünsche

Dennoch bleibt die Situation unbefriedigend. Ideal wäre es gewesen, wenn das Programm durch die Reset-Taste gar nicht erst gelöscht, sondern sogar noch neu gestartet worden wäre. Unser erster Wunsch: das Programm sollte einen *Autostart* haben, d.h. beim Einschalten oder Reset selbsttätig beginnen.
Und wenn wir schon beim Wünschen sind: Ein Codeschloß nutzt natürlich nichts, wenn es beim Stromausfall sein Gedächtnis verliert und erst wieder mit dem Programm geladen werden muß. Unser zweiter Wunsch lautet also: Das Programm soll *nullspannungs-sicher* sein.
Zuletzt der dritte Wunsch: Wenn Programm und Experiment-Aufbau getestet sind und zur Zufriedenheit funktionieren, wenn zudem das Programm auch nicht mehr neu geladen werden muß, weil es nullspannungs-sicher ist, und wenn es schließlich dank des Autostarts nicht mehr durch Eintippen von RUN gestartet werden muß, dann wäre der PC eigentlich wieder entbehrlich: Während BERT als Codeschloß seinen Dienst versieht, könnte am PC schon die nächste Aufgabe, vielleicht mit dem nächsten BERT, bearbeitet werden.
Unser dritter Wunsch: das Programm soll auf BERT *im autonomen Betrieb* laufen.
Wir brauchen keine Zauberfee aus dem Märchen zur Erfüllung unserer drei Wünsche; denn all diese Möglichkeiten sind bereits in BERT eingebaut: Der Schlüssel zu den neuen Möglichkeiten liegt in der Reset-Routine des BASIC/DEBUG, die wir jetzt beschreiben.

Die Reset-Routine des BASIC/DEBUG

Wenn der Reset-Eingang des Mikrocomputer-Bausteins Z8671 auf L-Pegel geht, so stoppt er alle Aktivitäten. Beim Wiederanstieg dieses Signals beginnt der Z8, das Maschinenprogramm im internen Festwertspeicher ab Adresse %000C auszuführen; dieses bewirkt einen Betriebsprogramm-Neustart. In dessen Verlauf werden alle Programm-Unterbrechungen abgeschaltet und einige Register vorbesetzt. Dabei wird auch die Benutzung der Ports 0 bis 3, also der Prozessor-Ports, wie in *Kapitel 8* beschrieben, festgelegt.
Danach wird die Übertragungsrate zum PC festgelegt, indem die Schalterstellung des Ports E eingelesen wird. In *Bild 9.2* sind alle Bitkombinationen des Ports E mit den dazugehörigen Übertragungsraten aufgeführt. Wie man erkennt, spielen dabei nur die niedrigstwertigen drei Bits eine Rolle. Die höheren Bits sind beliebig und können für andere Zwecke verwendet werden.

Einstellung an Port E (0 ≙ OFF)	Übertragungsrate in Bits pro Sekunde
X X X 0 0 0	150
X X X 0 0 1	19200
X X X 0 1 0	9600
X X X 0 1 1	4800
X X X 1 0 0	2400
X X X 1 0 1	1200
X X X 1 1 0	110
X X X 1 1 1	300

Bild 9.2: Einstellung der Übertragungsrate an den Schiebeschaltern des Ports E. Dabei bedeutet X: beliebige Schalterstellung.

In der Reset-Routine folgt dann das Überprüfen der Speicher-Bestückung. Je nach RAM-Ausbau wird den Variablen ihr Platz zugewiesen und der Stapelzeiger für Rücksprungadressen der Unterprogramme gesetzt; außerdem werden Zeiger auf Anfang und Ende des nutzbaren BASIC-Speichers gesetzt. Die Speicher-Aufteilung von BERT im Grundausbau ist in *Bild 9.3* angegeben. Wenn BERT dagegen 4 K Bytes Schreib-/Lese-Speicher hat, werden Eingabepuffer, Variablenbereich und Stapelspeicher an das Ende des zweiten Bausteins, also um %800 höher, gelegt.
Wichtig ist bei der Reset-Routine jedoch auch das Prüfen des Speichers bei niedrigen Adressen. Insbesondere interessiert sich das Betriebssystem für den Inhalt des Speicherzellen-Paares %1020 und %1021. Sollte sich hier RAM befinden, dann nimmt der Z8671 über BASIC/DEBUG den Dialog mit dem Benutzer auf, indem er den Doppelpunkt sendet. Bei BERT wissen wir jedoch, daß die Adressen %1020 und %1021 zum externen Festwertspeicher gehören. Nun kommt es darauf an, ob an dieser Stelle eine Zahl steht, deren höherwertiges Byte gelöscht ist. Mit anderen Worten: die Zahl muß positiv und kleiner als 256 sein. Ist dies nicht der Fall, so wird ebenfalls der Dialog aufgenommen.
Wenn die Bedingung jedoch erfüllt ist, nimmt BASIC/DEBUG die in %1020 und %1021 stehende Zahl als die erste Zeilennummer eines BASIC/DEBUG-Programms und beginnt sofort mit der Ausführung desselben. Dies ist genau, was wir gesucht hatten: die Möglichkeit eines Autostarts.
Schauen wir uns nun einmal an, was der Z8 in unserem BERT unter der Adresse %1020 findet. Wir benutzen hierzu unser Monitorprogramm; es wird in diesem Kapitel ab *Seite 201* ausführlich beschrieben. Geben Sie das Kommando:

GO@%101B,%1000,%1080

Der Bildschirm sollte sich daraufhin mit einem Speicherauszug füllen, wie er in *Bild 9.4* wiedergegeben ist. Die ersten paar Stellen des Festwertspeichers enthalten Sprungadressen; diese bestimmen, was bei den verschiedenen Programm-Unterbrechungen zu unternehmen ist. Es würde hier zu weit führen, die Abläufe bei Programm-Unterbrechungen im einzelnen zu erläutern.

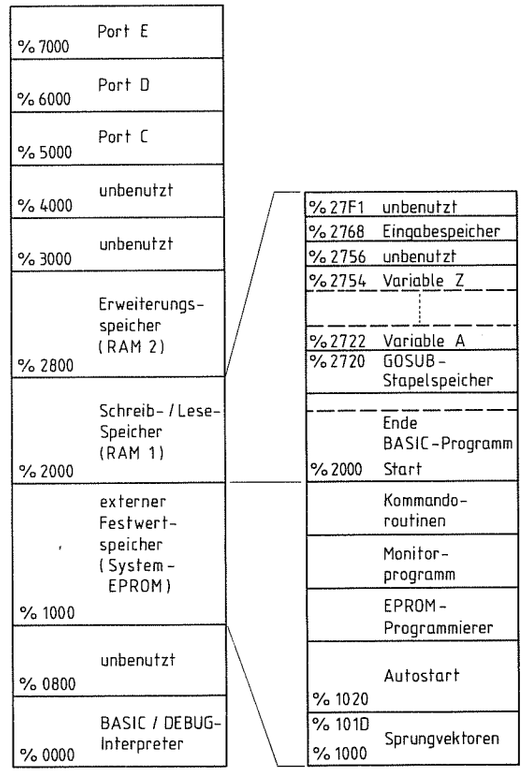
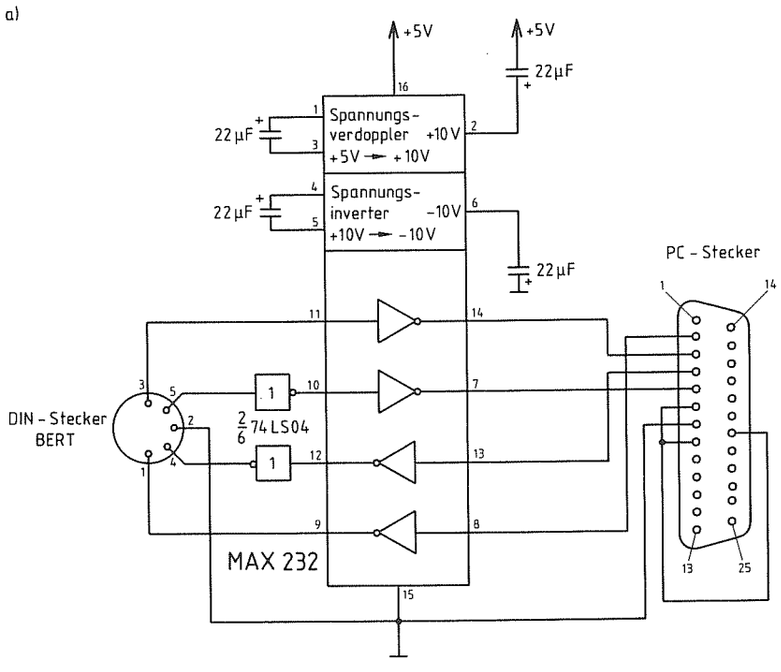
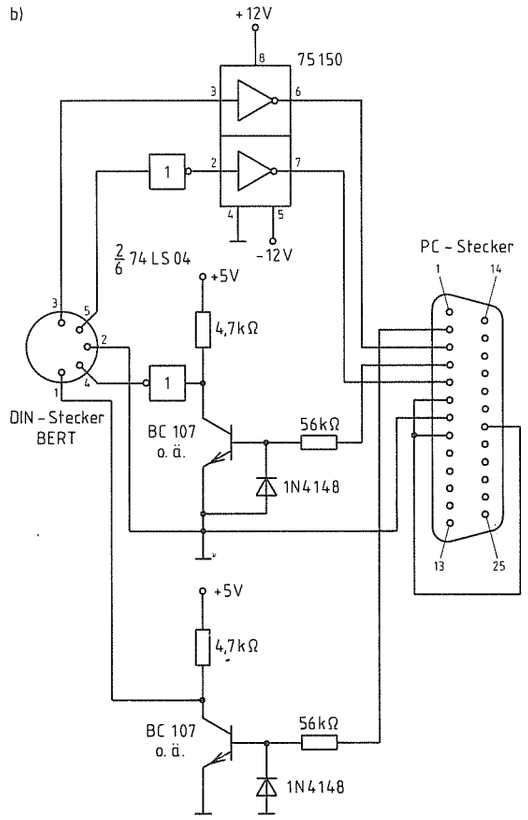


Bild 9.3: Die Speicher-Aufteilung im Grundausbau von BERT: Der Bereich des externen Festwertspeichers (EPROM-Baustein 2732) und der RAM-Bereich sind rechts noch einmal detailliert dargestellt.

B.2 Der Anschluß von BERT an den IBM-PC

Der IBM-PC sowie das Nachfolgemodell XT und die leistungsfähigere Version AT bilden die Basis eines Industriestandards. Neben den IBM-Geräten halten diesen Standard auch noch viele Computer von weiteren Firmen. Gemeinsam ist ihnen der Einsatz eines Mikroprozessors aus der 8086-Familie (z. B. 8088, 80186, 80286) sowie des Betriebssystems MS-DOS. Die BASIC-Version von IBM (BASICA) und das für die kompatiblen Computer lizenzierte Microsoft-BASIC (GW-BASIC) sind weitgehend identisch.
Wenn im folgenden und an anderer Stelle in diesem Buch von IBM-PCs die Rede ist, so wird dies auch für alle kompatiblen Computer gelten. Voraussetzung dafür ist, daß wir mit den PC-Programmen im Rahmen der normalen Benutzung von BASIC und Betriebssystem bleiben und nicht bis in die internen technischen Details vorstoßen, wo sich letztlich doch noch Unterschiede ergeben. Der IBM-PC und die zu ihm kompatiblen Computer besitzen eine Seriell-Schnittstelle oder können mit einer solchen in Form einer Steckkarte ausgerüstet werden. Diese Seriell-Schnittstelle erfüllt alle Vorschriften der V24-Norm, sowohl was die Pegel als auch was die Form des Steckers anbelangt.
Die Verbindung vom IBM-PC zu BERT erfordert somit, wie in *Kapitel 10* ausführlich beschrieben, eine Pegelumsetzung zwischen V24 und TTL. Eine dafür geeignete Auswahl an Schaltungen zeigt *Bild B2.1*. Die drei Versionen erfüllen die gleiche Funktion; allerdings besitzt diejenige mit dem Schaltkreis MAX 232 den Vorteil, nur mit einer Versorgungsspannung von +5 V gegen Masse auszukommen. Die anderen beiden Schaltungen benötigen außerdem noch +9 ... 12 V und −9 ... 12 V; dafür sind aber die hier eingesetzten ICs wohl leichter erhältlich. Die positive und negative Spannung von je 9 V können, wenn keine langen Betriebsdauern zu erwarten sind, auch aus zwei 9-V-Blockbatterien entnommen werden; vgl. *Bild 5.14*. Die Spannung von +5 V kann in jedem Fall BERT entnommen werden.
An BERT muß eine Kabelbrücke zwischen den Anschlüssen RTS und CTS gesteckt werden (vgl. auch hierzu *Kapitel 10*).





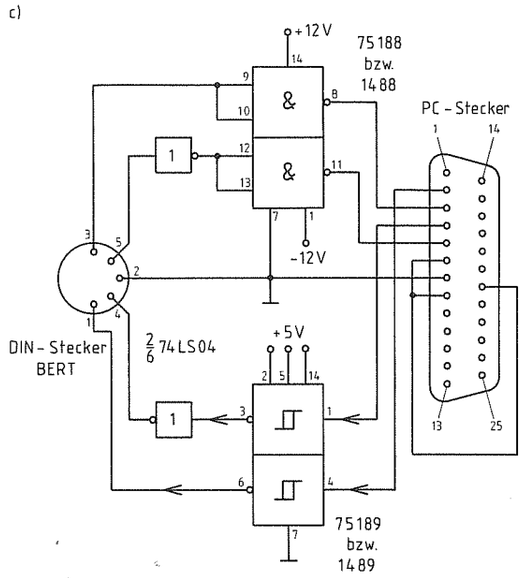


Bild B2.1: Der Anschluß von BERT an den IBM-PC mit Pegelwandlung zwischen V24 und TTL. Die drei hier gezeigten Schaltungs-Varianten sind gleichwertig.

Hinweise zur Programmierung

Die Seriell-Schnittstelle wird im Betriebssystem MS-DOS als COM1, Kommunikations-Schnittstelle 1, bezeichnet. Sie wird unter BASIC mit Hilfe der OPEN-Anweisung eingerichtet:

OPEN "COM1:Baud,Parit,Wortl,Stopb" AS #n

z. B.:

OPEN "COM1:1200,N,8,2" AS #2

Diese Anweisung stellt die Verbindung her zwischen der Seriell-Schnittstelle COM1 und dem logischen File #n, bei uns #2. Im Anschluß an die OPEN-Anweisung kann auf die Seriell-Schnittstelle unter Bezugnahme auf #2 zugegriffen werden (vgl. *Kapitel 2*), z. B.:

PRINT #2, ...
INPUT #2, ...

Die INKEY$-Funktion läßt sich dagegen nicht auf die V24-Schnittstelle umschalten. Statt dessen muß die Funktion

INPUT$(LOC(2), #2)

verwendet werden.
Es wird in der OPEN-Anweisung auch die Betriebsart der V24-Schnittstelle festgelegt. Dabei steht „Baud" für die Übertragungsrate, wobei man die Wahl zwischen 75, 110, 150, 300, 600, 1200, 2400, 4800 und 9600 Baud hat. Weiterhin werden mit „Parit,Wortl,Stopb" die Parität, die Wortlänge und die Anzahl der Stopbits eingestellt; in unserem Falle: N, d. h. No parity (kein Paritätsbit), ferner 8 Datenbits und 2 Stopbits.
Mit der OPEN-Anweisung werden auch noch zwei Pufferspeicher eingerichtet, die Texte zwischenspeichern, so daß sie bei Geschwindigkeits-Unter-

schieden zwischen PC-BASIC-Programm und V24-Übertragung nicht verlorengehen.
Das Vorhandensein der Pufferspeicher hat noch einen unangenehmen Nebeneffekt. Programme, die nur Kommandos an BERT senden, jedoch auf keine Rückmeldung von BERT warten müssen, laufen eventuell schneller ab, als das Senden der Daten aufgrund der eingestellten Übertragungsrate möglich ist. In der Folge werden immer mehr Daten in den Ausgabe-Puffer geschrieben, um auf das Senden zu warten. Erst wenn der Puffer voll ist, wird das BASIC-Programm beim PRINT#2-Kommando angehalten, bis wieder der erforderliche Platz im Zwischenspeicher verfügbar ist.
Ein voller Puffer im PC bedeutet aber, daß BERT mit der Ausführung der an ihn gesandten Kommandos um bis zu 20 oder gar 30 Befehle hinter dem BASIC-Steuerprogramm hinterherhinken kann. Der geschilderte Fall tritt bei den Programmen aus den *Bildern 5.18, 5.19, 5.22 und 7.35* auf. Anhand der Programmvariante des letztgenannten Beispiels in *Bild B2.2* können Sie sich von diesem Effekt überzeugen: Das Programm wurde durch einen PRINT-Befehl und eine je nach Geschwindigkeit des PCs einzustellende Verzögerungsschleife etwas langsamer (und komfortabler), so daß sich der Übertragungspuffer nicht mehr füllt. Das Servo reagiert nun mit wesentlich geringerer Verzögerung auf die Steuerkommandos.

```
100 REM Servosteuerung.2
102 REM ================
110 REM Steuerung eines Modellbauservos
120 REM Der Sollwinkel des Servos wird
130 REM durch die Impulsbreite am Ein-
140 REM gang gesteuert.
150 REM 1.5 ms - Neutralstellung
160 REM 2.0 ms - Linksausschlag
170 REM 1.0 ms - Rechtsausschlag
180 REM B6 - Impulseingang des Servos
200 M=1380 :REM Neutralstellung
210 R=920  :REM Rechtsausschlag
220 L=1840 :REM Linksausschlag
230 VI=10 :REM Stellgeschwindigkeit
240 PRINT"Steuerung eines Servos"
250 PRINT
255 PRINT"Num-Lock einrasten!"
260 PRINT"Cursor links  - nach links"
270 PRINT"Cursor rechts - nach rechts"
280 PRINT"Leertaste     - Neutralstell."
290 V=M :REM Start mit Neutralstellung
295 I=0
300 A$=INKEY$
310 PRINT#2,"I=";V
312 LOCATE 12,1 :PRINT "Impulsbreite/usec";
    INT(1500*V/M)
320 IF A$="" THEN GOTO 300
330 IF A$="6" THEN IF V>R THEN V=V-VI
340 IF A$="4" THEN IF V<L THEN V=V+VI
350 IF A$=" " THEN V=M
360 GOTO 300
```

Bild B2.2: Variiertes Servo-Programm (vgl. Text).

In *Bild B2.3* ist das in *Kapitel 9* erwähnte große Terminalprogramm aufgeführt:

```
10 REM Programm "TERMINAL" (IBM-PC advanced BASIC)
20 REM Dialogbetrieb mit BERT
30 REM Baudrate 300 bd, 8 Datenbits
40 REM keine Parität, 2 Stopbits
50 REM
60 CLEAR ,&H4000:OPEN "COM1:300,N,8,2" AS #2
70 GOSUB 9000
80 CL$=CHR$(29)+CHR$(32)+CHR$(29)
90 RA=&H4000 :REM Pufferspeicher 49152 Zeichen
100 B$=INKEY$ : IF B$=CHR$(27) THEN PRINT#2,B$
110 IF EOF(2) THEN 100
120 DU$=INPUT$(LOC(2),#2) : IF DU$<>":" THEN 100
130 GOSUB 9100
140 GOSUB 9150
150 KEY(1) ON : KEY(2) ON : KEY(3) ON : KEY(4) ON
```



```
160 KEY(5) ON : KEY(6) ON : KEY(7) ON : KEY(8) ON
170 ON KEY(1) GOSUB 1000 : ON KEY(2) GOSUB 2000 : ON KEY(3) GOSUB 3000
180 ON KEY(4) GOSUB 4000 : ON KEY(5) GOSUB 5000 : ON KEY(6) GOSUB 6000
190 ON KEY(7) GOSUB 7000 : ON KEY(8) GOSUB 8000
200 PRINT#2,CHR$(27); : REM ESC SENDEN
210 REM Übertragungsschleife
220 REM ====================
230 IF EOF(2) THEN 290
240 A$=INPUT$(LOC(2),#2)
250 IF A$=CHR$(8) THEN A$=CL$
260 IF A$<>CHR$(10) THEN PRINT A$;
270 IF PF=0 THEN 230
280 A%=ASC(A$):POKE RZ,A%:RZ=RZ+1:GOTO 230
290 B$=INKEY$
300 IF B$="" THEN 230
310 IF B$="ß"THEN B$=CHR$(64)
320 PRINT#2,B$;
330 IF KO THEN PRINT B$;
340 GOTO 230
1000 REM F1 BERT -> DISK (Programm)
1010 REM ==========================
1020 DU$="" : INPUT" > BERT --> DISK (Programm) ok (N)";DU$ : IF DU$="N"
     OR DU$="n" THEN RETURN
1030 GOSUB 1100 :REM BERT-PGM --> RAM
1040 GOSUB 9420 :REM Filenamen erfragen
1050 IF EFLAG THEN RETURN
1060 GOSUB 1200 :REM RAM --> DISK
1070 RETURN
1100 REM BERT-Programm --> RAM
1110 REM ---------------------
1120 RZ=RA
1130 PRINT#2,"LIST"+CHR$(13)
1140 INPUT#2,DU$
1150 IF EOF(2) THEN 1150
1160 A$=INPUT$(LOC(2),#2)
1170 IF A$=":" AND A%=10 THEN RETURN
1180 A%=ASC(A$)
1190 POKE RZ,A% : RZ=RZ+1 : PRINT A$; : GOTO 1150
1200 REM RAM --> DISK
1210 REM ------------
1220 FOR Z=RA TO RZ-1
1230 PRINT#1,CHR$(PEEK(Z));
1240 NEXT Z
1250 CLOSE #1
1260 PRINT":";
1270 RETURN
2000 REM F2 BERT --> DISK (Daten)
2010 REM ========================
2020 IF PF THEN 2100
2030 DU$="":INPUT" > BERT --> DISK (Daten) ok (N)";DU$ : IF DU$="N"
     OR DU$="n" THEN RETURN
2040 PF=1:LOCATE 25,23:COLOR 31:PRINT"SAVE D";:COLOR 7,0:LOCATE 24,1
2050 RZ=RA      :REM Pufferzeiger
2060 RETURN
```

```
2100 PRINT
2105 PF=1:LOCATE 25,23:COLOR 0,7:PRINT"SAVE D";:COLOR 7,0:LOCATE 24,1
2110 GOSUB 9420 :REM Filenamen erfragen
2120 IF EF THEN RETURN
2130 GOSUB 1200 :REM RAM --> DISK
2140 RETURN
3000 REM F3 DISK --> BERT (Programm)
3010 REM ============================
3020 DU$="" : INPUT" > DISK --> BERT (Programm) ok (N)";DU$ : IF DU$="N"
     OR DU$="n" THEN RETURN
3025 SY$=":"
3030 GOSUB 9430 :REM Filenamen erfragen
3040 IF EFLAG THEN RETURN
3050 IF EOF(1) THEN 3150
3060 B$=INPUT$(1,#1)
3070 PRINT#2,B$;
3080 IF EOF(2) THEN 3080
3090 A$=INPUT$(1,#2)
3100 PRINT A$;
3110 IF B$<>CHR$(13) THEN 3140
3120 IF EOF(2) THEN 3120
3130 A$=INPUT$(1,#2) : IF A$<>SY$ THEN 3120
3140 GOTO 3050
3150 CLOSE #1
3160 PRINT":";
3170 RETURN
4000 REM F4 DISK --> BERT (Daten)
4010 REM ========================
4020 DU$="" : INPUT" > DISK --> BERT (Daten) ok (N)";DU$ : IF DU$="N"
     OR DU$="n" THEN RETURN
4030 SY$="?"
4040 GOTO 3030 :REM benutze Routine für Programm
5000 REM F5 BERT --> DRUCKER (Programm)
5010 REM ==============================
5020 DU$="" : INPUT" > BERT --> DRUCKER (Programm) ok (N)";DU$ : IF DU$="N"
     OR DU$="n" THEN RETURN
5030 GOSUB 1100 :REM BERT --> RAM (Programm)
5040 GOSUB 5100 :REM RAM --> DRUCKER
5050 RETURN
5100 REM RAM --> DRUCKER
5110 REM ----------------
5120 FOR Z=RA TO RZ-1
5130 IF PEEK(Z)<>10 THEN LPRINT CHR$(PEEK(Z));
5140 NEXT Z
5150 LPRINT
5160 PRINT":";
5170 RETURN
6000 REM F6 BERT --> DRUCKER (Daten)
6010 REM ==========================
6020 IF PF THEN 6100
6030 DU$="":INPUT" > BERT --> DRUCKER (Daten) ok (N)";DU$ : IF DU$="N"
     OR DU$="n" THEN RETURN
6040 PF=1:LOCATE 25,47:COLOR 31:PRINT"PRINT D";:COLOR 7,0:LOCATE 24,1
6050 RZ=RA      :REM Pufferzeiger
```



```
6060 RETURN
6100 PF=0:LOCATE 25,47:COLOR 0,7:PRINT"PRINT D";:COLOR 7,0:LOCATE 24,1
6110 GOSUB 5100 :REM RAM --> DRUCKER
6120 RETURN
7000 REM F7 Programmende
7010 REM ===============
7020 CLOSE
7030 CLS : KEY ON
7040 END
8000 REM F8 Umschalten KOMMANDO / BASIC
8010 REM ==============================
8020 IF KO THEN 8060
8030 PRINT#2,"GO@%1018" : INPUT#2,DU$ : KO=1
8040 GOSUB 9100
8050 RETURN
8060 KO=0 : PRINT#2,CHR$(27)
8070 GOSUB 9100
8080 RETURN
9000 REM Bildschirmmeldungen
9010 REM ====================
9020 WIDTH 80 : CLS : KEY OFF : COLOR 0,7
9030 PRINT"      B E R T   T E R M I N A L - B E T R I E B
                 ";
9040 PRINT"      3 0 0   B A U D :   P O R T   E   A U F   7   =
       X X X 1 1 1      ";
9050 COLOR 7,0 : PRINT
9060 PRINT"      BITTE <ESC> AM PC ODER RESET-TASTE AN BERT DRÜCKEN !"
9070 RETURN
9100 CLS : COLOR 0,7
9110 PRINT"      B E R T   T E R M I N A L - B E T R I E B
                 ";
9120 IF KO THEN PRINT"      Z 8 6 7 1   K O M M A N D O B E T R I E B
                                        ";: COLOR 7,0 : RETURN
9130 PRINT"      Z 8 6 7 1   B A S I C   /   D E B U G
                 ";
9140 COLOR 7,0
9150 LOCATE 25,1
9160 PRINT"BASIC/DEBUG  1";
9170 COLOR 0,7 : PRINT"SAVE P"; : COLOR 7,0
9180 PRINT" 2";
9190 COLOR 0,7 : PRINT"SAVE D"; : COLOR 7,0
9200 PRINT" 3";
9210 COLOR 0,7 : PRINT"LOAD P"; : COLOR 7,0
9220 PRINT" 4";
9230 COLOR 0,7 : PRINT"LOAD D"; : COLOR 7,0
9240 PRINT" 5";
9250 COLOR 0,7 : PRINT"PRINT P"; : COLOR 7,0
9260 PRINT" 6";
9270 COLOR 0,7 : PRINT"PRINT D"; : COLOR 7,0
9280 PRINT" 7";
9290 COLOR 0,7 : PRINT"ENDE"; : COLOR 7,0
9300 PRINT" 8";
9310 COLOR 0,7 : PRINT"KOM/BASIC"; : COLOR 7,0
9320 LOCATE 4,1
```

```
9330 RETURN
9400 REM Filenamen erfragen
9410 REM ==================
9420 SFLAG=1 : GOTO 9440 : REM Einsprung SAVE
9430 SFLAG=0             : REM Einsprung LOAD
9440 EFLAG=0 : DU$=""
9450 PRINT : PRINT" > Filename :";CHR$(34);F$;CHR$(34);
     : INPUT"  (J/?/Name)";DU$
9460 IF DU$<>"" THEN 9500
9470 INPUT" > Funktion abbrechen (J/N) ";DU$
9480 IF DU$<>"N" AND DU$<>"n" THEN EFLAG=1 : RETURN
9490 GOTO 9440
9500 IF DU$="?" THEN 9700
9510 IF F$<>"" THEN 9540
9520 IF DU$="J" OR DU$="j" THEN 9470
9530 GOTO 9550
9540 IF DU$="J" OR DU$="j" THEN 9560
9550 F$=DU$
9560 DU$=F$+".B-D"
9570 IF SFLAG THEN 9600
9580 OPEN DU$ FOR INPUT AS #1
9590 RETURN
9600 OPEN DU$ FOR OUTPUT AS #1
9610 RETURN
9700 REM Directory ausgeben
9710 REM ==================
9720 FILES
9730 GOTO 9440
```

Bild B2.3: Das große Terminal-Programm für den IBM-PC.

B.3 Der Anschluß von BERT an den Apple II

Die Apple-Computer II, II+, II-europlus und IIe haben keine eingebaute Seriell-Schnittstelle. Wegen des „offenen Konzepts" kann jedoch eine Erweiterungskarte in den Computer eingesteckt werden. Wir beziehen uns im folgenden auf die Super-Serial-Card (SSC) von Apple. Die ältere Serial-Interface-Card (SIC) ist nur zum Betrieb von Druckern mit V24-Schnittstelle geeignet, nicht aber für die bidirektionale Kommunikation. Ein Analogon zur Super-Serial-Card befindet sich auch im (geschlossenen) Apple IIc auf der Hauptplatine, so daß zumindest der Terminalbetrieb auch mit dem Apple IIc möglich ist. Unterschiede gilt es bei der Ausführung der Steckverbinder zu beachten.

Der Anschluß an die Super-Serial-Card

Die Betriebsbedingungen der Super-Serial-Card lassen sich durch Schiebeschalter einstellen. Wir empfehlen, die Betriebsbedingungen, die im INIT-Teil nochmals per Software eingestellt werden (siehe *Kapitel 2*), schon auf der Platine wie folgt einzustellen:

Schalter SW1

1	2	3	4	5	6	7
off	on	on	on	on	on	on

Schalter SW2

1	2	3	4	5	6	7
on	on	on	on	off	off	off

Die Super-Serial-Card muß nun in den Steckplatz 2 des Apple II eingesetzt werden. Schalten Sie hierzu unbedingt die Versorgungsspannung des Apple II und möglichst auch noch die angeschlossener Peripheriegeräte aus. Nehmen Sie die Super-Serial-Card zur Hand; auf der Bestükkungsseite befindet sich ein Stecker, der mit einem Pfeil gekennzeichnet ist. Dieser muß zur Beschriftung „Modem" weisen.
Das Kabel verlegen Sie durch einen der rückwärtigen Schlitze nach außen bzw. montieren es nach der Vorschrift, die der Karte beiliegt, in der Rückwand des Computers.

Die Verbindung von BERT zur Super-Serial-Card wird nach einem der Schaltpläne in *Bild B3.1* hergestellt. Wir zeigen hier mehrere gleichwertige Varianten der Pegelanpassung zwischen V24 und TTL, so daß Sie sich je nach Verfügbarkeit der ICs entscheiden können. Die Schaltung mit dem MAX 232 benötigt nur eine Versorgungsspannung von +5 V, die Sie BERT entnehmen können. Die anderen Schaltungen benötigen darüber hinaus noch die positive und negative Spannung von je 9 ... 12 V für die V24-Pegel. Für kurze Betriebsdauern genügen zwei 9-V-Blockbatterien, die Sie wie in *Bild 5.14* verschalten. Die Nummern der Anschlüsse in *Bild B3.1* beziehen sich auf die benutzten Steckverbinder: ein 25poliger Stiftverbinder zur Super-Serial-Card und ein fünfpoliger DIN-Stecker zu BERT. An BERT muß zudem eine Kabelbrücke zwischen den Stiften CTS und RTS gesteckt werden.
Für diejenigen unter Ihnen, die sich schon mit *Kapitel 10* beschäftigt haben, hier einige weitere Informationen: Von den V24-Signalen der Super-Serial-Card benutzen wir die Sendeleitung TXD, die Empfangsleitung RXD und die Quittierungs-Leitungen CTS und RTS. Nicht benutzt – und gleich mit einer Drahtbrücke auf dem Stecker verbunden – werden die beiden Leitungen DSR und DTR.
Die oben angeführten Schalterstellungen entsprechen folgenden Betriebsdaten:

Kommunikationsbetrieb
1200 Baud
8 Datenbits, 1 Stopbit
kein Paritätsbit
kein Line Feed nach Carriage Return

Die gleichen Parameter werden im INIT-Teil der Programme (siehe *Kapitel 2*) noch einmal per Steuerkommandos eingestellt. Diese sind ähnlich den DOS-Steuerkommandos aufgebaut, indem sie mit einem Steuerzeichen beginnen. In diesem Fall ist es CTRL-A, abgelegt in der String-Variablen A$ = CHR$(1). Der diesem Steuerzeichen nachfolgende Text wird nicht an BERT übertragen, sondern von der Super-Serial-Card interpretiert. Es könnte eventuell von Interesse sein, die Übertragungs-Geschwindigkeit zu ändern:

```
23 PRINT A$;"10 BAUD": REM 2400 BAUD
```

Betrieb des Terminalprogramms an allen Apple-II-Modellen, auch am Apple IIc, möglich.
Wer seinen Apple-Computer (II, II+, II-europlus oder IIe) nur als Terminal zu BERT einsetzen will, benötigt aus dem gleichen Grund auch nicht die aufwendige Super-Serial-Card. Wenn Sie über hinreichend Erfahrung verfügen, können Sie auch eine vereinfachte Seriellkarte nach *Bild B3.4* aufbauen, die im wesentlichen nur den Baustein ACIA 6551 enthält. Die Pegelumsetzung entfällt hierbei, da der Baustein ACIA 6551 wie BERT TTL-Pegel liefert bzw. verarbeitet.

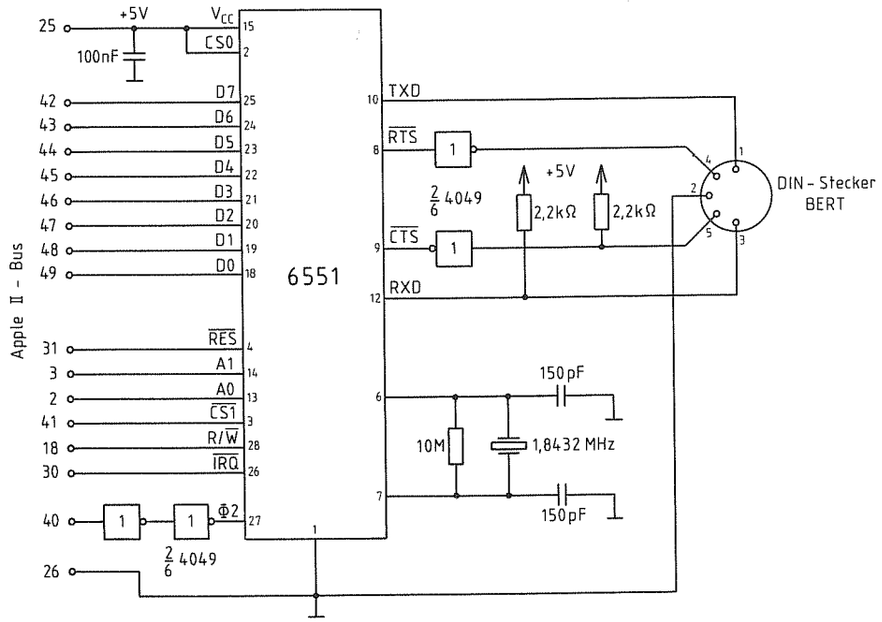


Bild B3.4: Aufbau einer einfachen Seriellkarte für den Apple II und ihr Anschluß an BERT.

C Der Aufbau von BERT

Die Platine

Die Abbildung der BERT-Platine mit dem Bestückungsplan finden Sie auf Seite 159 *(Bild 8.1)*.

Die Experiment-Anschlüsse

Die Abbildung der 30 Port-Anschlüsse finden Sie ebenfalls auf Seite 159 in *Bild 8.1*. Es stehen Ihnen folgende Möglichkeiten für die Ausgabe und Eingabe von TTL-Signalen zur Verfügung:

Port A

An jedem der 8 Anschlüsse A0 bis A7 wahlweise Eingabe oder Ausgabe (vgl. u.a. *Kapitel 2*).
Multiplex-Betrieb (u.a. Spaltenauswahl; siehe *Kapitel 6*).

Port B

B1, B2, B3: Eingabe
B4, B5, B6: Ausgabe

Spezielle Anwendungen:

B1 (T_{in}): serielle Eingabe, insbesondere Zeitmessung und Impuls-Zählung (siehe *Kapitel 3*),
Handshake-Betrieb (siehe *Kapitel 4*).

B3 und B4:

AD-Wandler-Steuerung: Abfrage des EOC-Signals bzw. Ausgabe des START-Signals (siehe *Kapitel 5*).

B4:
erzeugt einen (negativen) Strobe-Impuls für die BERT-Kommandos *AS=* und *AS?*; siehe Anhang D.

B5 und B6:

AD-Wandler-Steuerung: Auswahl des Analogkanals (siehe *Kapitel 5*).

B6 (T_{out}): serielle Ausgabe, insbesondere Frequenz- und Impuls-Erzeugung (siehe *Kapitel 3*),
Handshake-Betrieb (siehe *Kapitel 4*),
Motor-Steuerung: Geschwindigkeits-Steuerung per Pulsbreiten-Variation (siehe *Kapitel 7*).

Port C

Eingabe, auch an einzelnen Anschlüssen, wie in *Kapitel 2* für Port A beschrieben;

Multiplex-Eingabe; vgl. Anhang D und *Kapitel 6*;

AD-Wandlung: Einlesen des Datenbytes mit den BERT-Kommandos *AD?* bzw. *ADn?* (siehe *Kapitel 5*).

Port D

Ausgabe, auch an einzelnen Anschlüssen, wie in *Kapitel 2* für Port A beschrieben;

Multiplex-Ausgabe; vgl. Anhang D und *Kapitel 6*;

Schrittmotor-Steuerung; siehe Anhang D und *Kapitel 7*.

Schaltplan

Den Schaltplan von BERT finden Sie auf Seite 161 *(Bild 8.2)*.

Speicher-Aufteilung und Register-Belegung

Die Speicher-Aufteilung von BERT finden Sie in *Bild 9.3* auf Seite 193.

Die Register-Belegung des Z8671 in BERT ist nachstehend aufgeführt.

%FF Stapelzeiger low
%FE Stapelzeiger high
%FD Registerblock-Zeiger
%FC Flaggenregister
%FB Programmunterbrechung Maskenregister
%FA Programmunterbrechung Anforderungsregister
%F9 Programmunterbrechung Prioritätsregister
%F8 Betriebsartenregister P0 und P1 (Daten- und Adreßbus)
%F7 Betriebsartenregister P3 (Port B und Terminalleitungen)
%F6 Datenrichtungsregister P2 (Port A)
%F5 Vorteilerregister 0
%F4 Zähler-/Zeitgeberregister 0
%F3 Vorteilerregister 1
%F2 Zähler-/Zeitgeberregister 1
%F1 Betriebsartenregister Zähler/Zeitgeber
%F0 Terminal Datenregister
%EF unbelegt
... "
%80 unbelegt
%7F Arithmetik-Stapelspeicher
... "
%6B Arithmetik-Stapelspeicher
%6A Multiplexdaten Port A
... "
%63 Multiplexdaten Port A
%62 Multiplexdaten Port C
... "
%5B Multiplexdaten Port C
%5A Multiplexdaten Port D
... "
%53 Multiplexdaten Port D
%52 Schrittmotor 3 Schrittzahl low
%51 " " high
%50 Schrittmotor 2 Schrittzahl low
%4F " " high
%4E Schrittmotor 1 Schrittzahl low
%4D " " high
%4C Schrittmotor 0 Schrittzahl low
%4B " " high
%4A Handshake Daten
%49 Handshake Flagge
%48 Programmunterbrechung Quelle
%47 Zählerüberlauf
%46 Kopie des Registers %F6 (Port A Datenrichtung)
%45 Abeitsregister bei Programmunterbrechungen
... " "
%40 Arbeitsregister bei Programmunterbrechungen
%3F Zeiger Arithmetik-Stapelspeicher (Kommandos)
%3E Kopie der letzten Ausgabe an Port D
%3D Kopie der letzten Ausgabe an Port B
%3C Kopie der letzten Ausgabe an Port A
%3B Kommandointerpreter Textzeiger low
%3A " " high

%39 Kommandointerpreter Tabellenzeiger low
%38 " " high
%37 Kommandointerpreter Kommandozeiger low
%36 " " high
%35 Kommandointerpreter Arbeitsregister
... " "
%30 Kommandointerpreter Arbeitsregister
%2F Kommandointerpreter Eingabespeicher
... " "
%20 Kommandointerpreter Eingabespeicher
%1F Zeiger Arithmetik-Stapelspeicher (BASIC/DEBUG)
%1E BASIC/DEBUG-Arbeitsregister
... " "
%16 BASIC/DEBUG-Arbeitsregister
%15 USR-Funktion 2. Argument low
%14 " " high
%13 USR-Funktion 3. Argument low
%12 " " high
%11 BASIC/DEBUG-Arbeitsregister
%10 " "
%0F BASIC/DEBUG-Entnahmezeiger in Eingabespeicher low
%0E " " " high
%0D BASIC/DEBUG Aufnahmezeiger in Eingabespeicher low
%0C " " " high
%0B BASIC/DEBUG Obergrenze nutzbares RAM low
%0A " " " RAM high
%09 BASIC/DEBUG-Zeiger auf Programmanfang low
%08 " " " high
%07 BASIC/DEBUG-Zeiger Unterprogramm Stapelspeicher low
%06 " " " " high
%05 BASIC/DEBUG-Zeiger auf Programmende low
%04 " " " high
%03 P3 = Port B
%02 P2 = Port A
%01 P1 = Daten-/Adreßbus (DA0 bis DA7)
%00 P0 = Adreßbus (A8 bis A15)

D Der Aufruf der BERT-Kommandos

Nachstehend sind alle BERT-Kommandos aufgeführt, neben ihren Einsprung-Adressen aus BASIC/DEBUG-Programmen. So ist mit einem Blick zu erkennen, wie Programme, die bislang auf dem PC liefen, in autonome Programme auf BERT unter BASIC/DEBUG umgesetzt werden. In der dritten Spalte sind die Aufrufe aus dem PC-BASIC aufgelistet.

Eingabe (vgl. *Kapitel 2*)

Die Ports A und C und die untere Hälfte des Ports B eignen sich zur Eingabe von Bitmustern. Aber auch Port E, der Baudraten-Einsteller, gehört zu den Eingabeports, wenn auch hier nur Schalterstellungen und keine elektrischen Signale einzulesen sind. Bei der Eingabe an Port A wird Handshake und Multiplex abgeschaltet. Das adressierte Bit bzw. der ganze Port wird auf Eingabe geschaltet. Beim Einlesen von Port B bzw. von Stift B1 wird Handshake, Zeit- und Zählereingabe abgeschaltet.

BERT-Kommando	BASIC/DEBUG-Aufruf	PC-BASIC-Aufruf
A?	v=USR(%1CB9)	PRINT#2,"A?":INPUT#2,var
B?	v=USR(%1CC9)	PRINT#2,"B?":INPUT#2,var
C?	v=USR(%1CD3)	PRINT#2,"C?":INPUT#2,var
E?	v=USR(%1CD9)	PRINT#2,"E?":INPUT#2,var
A0?	v=USR(%1D6E,1)	PRINT#2,"A0?":INPUT#2,var
A1?	v=USR(%1D6C,1)	PRINT#2,"A1?":INPUT#2,var
A2?	v=USR(%1D6A,1)	PRINT#2,"A2?":INPUT#2,var
A3?	v=USR(%1D68,1)	PRINT#2,"A3?":INPUT#2,var
A4?	v=USR(%1D66,1)	PRINT#2,"A4?":INPUT#2,var
A5?	v=USR(%1D64,1)	PRINT#2,"A5?":INPUT#2,var
A6?	v=USR(%1D62,1)	PRINT#2,"A6?":INPUT#2,var
A7?	v=USR(%1D60,1)	PRINT#2,"A7?":INPUT#2,var
B1?	v=USR(%1D7F,1)	PRINT#2,"B1?":INPUT#2,var
B2?	v=USR(%1D7D,1)	PRINT#2,"B2?":INPUT#2,var
B3?	v=USR(%1D7B,1)	PRINT#2,"B3?":INPUT#2,var
C0?	v=USR(%1D9E,1)	PRINT#2,"C0?":INPUT#2,var
C1?	v=USR(%1D9C,1)	PRINT#2,"C1?":INPUT#2,var
C2?	v=USR(%1D9A,1)	PRINT#2,"C2?":INPUT#2,var
C3?	v=USR(%1D98,1)	PRINT#2,"C3?":INPUT#2,var
C4?	v=USR(%1D96,1)	PRINT#2,"C4?":INPUT#2,var
C5?	v=USR(%1D94,1)	PRINT#2,"C5?":INPUT#2,var
C6?	v=USR(%1D92,1)	PRINT#2,"C6?":INPUT#2,var
C7?	v=USR(%1D90,1)	PRINT#2,"C7?":INPUT#2,var
E0?	v=USR(%1DAE,1)	PRINT#2,"E0?":INPUT#2,var
E1?	v=USR(%1DAC,1)	PRINT#2,"E1?":INPUT#2,var
E2?	v=USR(%1DAA,1)	PRINT#2,"E2?":INPUT#2,var
E3?	v=USR(%1DA8,1)	PRINT#2,"E3?":INPUT#2,var
E4?	v=USR(%1DA6,1)	PRINT#2,"E4?":INPUT#2,var
E5?	v=USR(%1DA4,1)	PRINT#2,"E5?":INPUT#2,var

Ausgabe (vgl. *Kapitel 2*)

Eine Ausgabe kann am Port A, ferner an der oberen Hälfte des Ports B und am Port D erfolgen. Bei einer Ausgabe an Port A wird das adressierte Bit bzw. der ganze Port auf Ausgabe geschaltet. Außerdem wird Handshake und Multiplex abgeschaltet. Bei einer Ausgabe an Port B bzw. an Stift B6 werden Handshake-, Zeitgeber- und Motor-Betrieb (Kommando *DC=expr*) abgeschaltet. Bei einer Ausgabe an Port D werden alle acht Multiplexregister des Ports D gleichermaßen gesetzt, so daß sich selbst bei laufendem Multiplex statische Bitmuster an Port D ergeben.

BERT-Kommando	BASIC/DEBUG-Aufruf	PC-BASIC-Aufruf
A=	GO@%1C92,expr	PRINT#2,"A=";expr
B=	GO@%1CA1,expr	PRINT#2,"B=";expr
D=	GO@%1CB0,expr	PRINT#2,"D=";expr
A0=	GO@%1CEE,1,expr	PRINT#2,"A0=";expr
A1=	GO@%1CEC,1,expr	PRINT#2,"A1=";expr
A2=	GO@%1CEA,1,expr	PRINT#2,"A2=";expr
A3=	GO@%1CE8,1,expr	PRINT#2,"A3=";expr
A4=	GO@%1CE6,1,expr	PRINT#2,"A4=";expr
A5=	GO@%1CE4,1,expr	PRINT#2,"A5=";expr
A6=	GO@%1CE2,1,expr	PRINT#2,"A6=";expr
A7=	GO@%1CE0,1,expr	PRINT#2,"A7=";expr
B4=	GO@%1D0D,1,expr	PRINT#2,"B4=";expr
B5=	GO@%1D0B,1,expr	PRINT#2,"B5=";expr
B6=	GO@%1D09,1,expr	PRINT#2,"B6=";expr
D0=	GO@%1D44,1,expr	PRINT#2,"D0=";expr
D1=	GO@%1D42,1,expr	PRINT#2,"D1=";expr
D2=	GO@%1D40,1,expr	PRINT#2,"D2=";expr
D3=	GO@%1D3E,1,expr	PRINT#2,"D3=";expr
D4=	GO@%1D3C,1,expr	PRINT#2,"D4=";expr
D5=	GO@%1D3A,1,expr	PRINT#2,"D5=";expr
D6=	GO@%1D38,1,expr	PRINT#2,"D6=";expr
D7=	GO@%1D36,1,expr	PRINT#2,"D7=";expr

Handshake (vgl. *Kapitel 4*):

Beim Absetzen von Handshake-Kommandos wird nicht nur Port A benutzt, sondern auch die Stifte B1 und B6 bzw. B4. Die AH-Kommandos schalten Multiplex-, Zeitgeber- und Motorbetrieb ab.
Mit dem Kommando *AS=expr* wird der Wert von expr an Port A ausgegeben. Anschließend folgt ein kurzer negativer Impuls an B4. Getriggert durch diesen Impuls kann ein externes Gerät die Daten von Port A übernehmen.
Beim Kommando AS? erscheint am Ausgang B4 ein negativer Impuls, der ein externes Gerät veranlassen soll, seine Daten an Port A zu legen. Nach dem Einlesen von Port A wird B4 wieder auf H geschaltet.
Der USR-Aufruf am Ende des Kommandos *AH=expr* liest die als Quittung ausgegebene Ziffer 1 ein; vgl. *Seite 72*.

BERT-Kommando	BASIC/DEBUG-Aufruf	PC-BASIC-Aufruf
AH?	GO@%1ECB:... ...v=USR(%1B57)	PRINT#2,"AH?":INPUT#2,var
AH=	GO@%1EB6,expr:... ...v=USR(%1B57)	PRINT#2,"AH=";expr... ...INPUT#2,var
AS?	v=USR(%1F89)	PRINT#2,"AS?":INPUT#2,var
AS=	GO@%1F79,expr	PRINT#2,"AS=";expr

Zeitgeber (vgl. *Kapitel 3*)

Die Zeitgeber-Kommandos benutzen die Stifte B1 (T_{in}) und B6 (T_{out}). Die Zeitgeber-Kommandos schalten Handshake-, Multiplex- und Motor-Betrieb ab.

BERT-Kommando	BASIC/DEBUG-Aufruf	PC-BASIC-Aufruf
Z?	v=USR(%1DE7)	PRINT#2,"Z?":INPUT#2,var
Z=	GO@%1DBB,expr	PRINT#2,"Z=";expr
I=	GO@%1E25,expr	PRINT#2,"I=";expr
P=	GO@%1E21,expr	PRINT#2,"P=";expr
F=	GO@%1DF1,expr	PRINT#2,"F=";expr
T?	v=USR(%1DE7)	PRINT#2,"T?":INPUT#2,var
T=	GO@%1DC2,expr	PRINT#2,"T=";expr

Analog-Eingabe (vgl. *Kapitel 5*)

Die Analog-Eingabekommandos benutzen Port C und die Stifte B3 und B4. Wird die Analogkanal-Adresse an den Stiften B5 und B6 ausgegeben, so werden Handshake-, Zeitgeber- und Motorbetrieb (Kommando *DC=expr*) abgeschaltet.

BERT-Kommando	BASIC/DEBUG-Aufruf	PC-BASIC-Aufruf
AD?	v=USR(%1FA9)	PRINT#2,"AD?":INPUT#2,var
AD0?	v=USR(%1FA3,1)	PRINT#2,"AD0?":INPUT#2,var
AD1?	v=USR(%1FA2,1)	PRINT#2,"AD1?":INPUT#2,var
AD2?	v=USR(%1FA1,1)	PRINT#2,"AD2?":INPUT#2,var
AD3?	v=USR(%1FA0,1)	PRINT#2,"AD3?":INPUT#2,var

Motor-Betrieb (vgl. *Kapitel 7*)

Das Pulsbreiten-Kommando *DC=expr* benutzt den internen Zeitgeber mit Ausgabe an Stift B6. Handshake-, Multiplex- und Zeitgeber-Betrieb werden daher abgeschaltet.
Die Schrittmotor-Kommandos benutzen ebenfalls den internen Zeitgeber; die Ausgabe erfolgt jedoch an Port D. Handshake-, Multiplex- und Zeitgeber-Betrieb werden abgeschaltet, aber die Ausgabe-Möglichkeit an Stift B6 bleibt erhalten.

BERT-Kommando	BASIC/DEBUG-Aufruf	PC-BASIC-Aufruf
S0=	GO@%1F0E,1,expr	PRINT#2,"S0=";expr
S1=	GO@%1F0C,1,expr	PRINT#2,"S1=";expr
S2=	GO@%1F0A,1,expr	PRINT#2,"S2=";expr
S3=	GO@%1F08,1,expr	PRINT#2,"S3=",expr
S0?	v=USR(%1F72,1)	PRINT#2,"S0?":INPUT#2,var
S1?	v=USR(%1F70,1)	PRINT#2,"S1?":INPUT#2,var
S2?	v=USR(%1F6E,1)	PRINT#2,"S2?":INPUT#2,var
S3?	v=USR(%1F6C,1)	PRINT#2,"S3?":INPUT#2,var
SP=	GO@%1F52,expr	PRINT#2,"SP=";expr
DC=	GO@%1FC8,expr	PRINT#2,"DC=";expr

Multiplex (vgl. *Kapitel 6*)

Die Multiplex-Kommandos verwenden den internen Zeitgeber zur Steuerung der Abläufe. Daher werden Handshake-, Zeitgeber- und Motor-Betrieb abgeschaltet. Port A wird, beginnend mit Bit 0, entsprechend

BERT-Kommando	BASIC/DEBUG-Aufruf	PC-BASIC-Aufruf
AM0?	v=USR(%1E63,1)	PRINT#2,"AM0?":INPUT#2,var
AM1?	v=USR(%1E62,1)	PRINT#2,"AM1?":INPUT#2,var
AM2?	v=USR(%1E61,1)	PRINT#2,"AM2?":INPUT#2,var
AM3?	v=USR(%1E60,1)	PRINT#2,"AM3?":INPUT#2,var
AM4?	v=USR(%1E5F,1)	PRINT#2,"AM4?":INPUT#2,var
AM5?	v=USR(%1E5E,1)	PRINT#2,"AM5?":INPUT#2,var
AM6?	v=USR(%1E5D,1)	PRINT#2,"AM6?":INPUT#2,var
AM7?	v=USR(%1E5C,1)	PRINT#2,"AM7?":INPUT#2,var
CM0?	v=USR(%1E6F,1)	PRINT#2,"CM0?":INPUT#2,var
CM1?	v=USR(%1E6E,1)	PRINT#2,"CM1?":INPUT#2,var
CM2?	v=USR(%1E6D,1)	PRINT#2,"CM2?":INPUT#2,var
CM3?	v=USR(%1E6C,1)	PRINT#2,"CM3?":INPUT#2,var
CM4?	v=USR(%1E6B,1)	PRINT#2,"CM4?":INPUT#2,var
CM5?	v=USR(%1E6A,1)	PRINT#2,"CM5?":INPUT#2,var
CM6?	v=USR(%1E69,1)	PRINT#2,"CM6?":INPUT#2,var
CM7?	v=USR(%1E68,1)	PRINT#2,"CM7?":INPUT#2,var
DM0=	GO@%1E7F,1,expr	PRINT#2,"DM0=";expr
DM1=	GO@%1E7E,1,expr	PRINT#2,"DM1=";expr
DM2=	GO@%1E7D,1,expr	PRINT#2,"DM2=";expr
DM3=	GO@%1E7C,1,expr	PRINT#2,"DM3=";expr
DM4=	GO@%1E7B,1,expr	PRINT#2,"DM4=";expr
DM5=	GO@%1E7A,1,expr	PRINT#2,"DM5=";expr
DM6=	GO@%1E79,1,expr	PRINT#2,"DM6=";expr
DM7=	GO@%1E78,1,expr	PRINT#2,"DM7=";expr
M=	GO@%1E86,expr	PRINT#2,"M=";expr

der Multiplex-Stellenzahl auf Ausgabe geschaltet, um die Multiplex-Auswahlsignale zu erzeugen. Die verbleibenden Stifte von Port A sind auf Eingabe geschaltet. Bei Abfrage der Port-A-Multiplexregister wird sowohl das Ausgabe- als auch das Eingabe-Bitmuster übergeben. Die Ausgabe an Port D erfolgt nach dem Einschalten des Multiplex nicht mehr durch das Port-D-Register (%3E), sondern durch die Port-D-Multiplexregister (%53 – %5A). Durch einfache Ausgabe-Kommandos werden diese jedoch identisch mit dem Port-D-Register (%3E) vorbesetzt. Daher bleibt die Ausgabe an Port D scheinbar unverändert bestehen, wenn das Multiplex-Verfahren benutzt wird, jedoch keine *DMn*-Kommandos abgesetzt wurden.

Sonstiges

Mit dem Kommando *G=expr* kann ein BERT-Maschinenprogramm aufgerufen werden, das bei der Adresse *expr* beginnt.

BERT-Kommando	BASIC/DEBUG-Aufruf	PC-BASIC-Aufruf
G=	GO@expr	PRINT#2,"G=";expr

Ergänzende Hinweise

Einige BERT-Kommandos benötigen zur Durchführung wegen des Umfangs ihrer Aufgaben einige Millisekunden Ausführzeit. Während dieser Zeit kann ein kurz darauffolgendes Kommando nicht erkannt werden oder wird verstümmelt empfangen. In diesen Fällen muß im PC-BASIC-Programm eine kurze Warteschleife nach dem Absetzen des Kommandos erfolgen. Dies gilt insbesondere für die Kommandos des Multiplexverfahrens und der Motorsteuerung. Die Programmierung der Warteschleife entnehmen Sie am besten den Programmbeispielen der *Kapitel 6 und 7*. Werden die Kommandos unter BASIC/DEBUG benutzt, ist eine Warteschleife nicht notwendig.
Die gleichen BERT-Kommandos, aber auch noch die Zeitmessung und das Zählerkommando, laufen automatisch neben dem eigentlichen Programmfluß. Sie werden durch Unterbrechungsprogramme des Zähler/Zeitgebers (Timer/Counters) gesteuert. Befindet sich ein BASIC/DEBUG-Programm in der Ausführung, wird nach Abarbeiten einiger Befehle die Unterbrechungsfreigabe vom BASIC/DEBUG-Interpreter abgeschaltet. Werden also die genannten Kommandos in BASIC/DEBUG-Programmen benutzt, so muß die Unterbrechungsfreigabe in kurzen Zeitabständen immer wieder eingeschaltet werden. Dies kann z.B. durch erneute Eingabe der Pulsbreite mit *DC=*, durch Abfragen des Zählerstands mit *Z?* bzw. *T?* oder durch erneutes Setzen der Spaltenzahl des Multiplex mit *M=* erfolgen. *Kapitel 11* enthält hierzu einige Beispiele. Derartige eigentlich überflüssige Kommandos sind allerdings nicht im PC-BASIC notwendig, da dann nicht BASIC/DEBUG, sondern der Kommando-Interpreter in BERT läuft.

E Die Programmiersprache BASIC/DEBUG

Liste der BASIC/DEBUG-Kommandos und -Funktionen

Im folgenden werden die Kommandos und Funktionen von BASIC/DEBUG in alphabetischer Reihenfolge aufgeführt. Dabei werden alle wahlfreien Teile mit spitzen Klammern abgegrenzt. Für zusätzliche Erläuterungen sind die Ausführungen in *Kapitel 8 und 9* zu Rate zu ziehen.

AND

Syntax:

AND(Ausdruck⟨,Ausdruck⟩)

Beispiele:

```
PRINT AND(16,@%7000)
LET X=12+AND(127,USR(%54))
```

Die AND-Funktion erzeugt das bitweise logische UND der beiden Argumente, die gültige BASIC/DEBUG-Ausdrücke sein müssen. Es kann dazu benutzt werden, um Bits in einem Datenwort zu löschen oder herauszumaskieren. Wird nur ein Argument angegeben, so wird es mit sich selbst UND-verknüpft. In diesem Fall ist der resultierende Funktionswert gerade das Argument selbst.

GO@

Syntax:

GO@Ausdruck⟨,Ausdruck⟨,Ausdruck⟩⟩

Beispiele:

```
GO@%61,%31
GO@%1018
```

Das GO@-Kommando verzweigt den Programmfluß in ein Maschinensprach-Programm und kann nur benutzt werden, wenn *kein* Wert aus dem Maschinensprach-Programm zurückzugeben ist (dazu dient die USR-Funktion).
Das GO@-Kommando muß wenigstens ein und kann auch zwei oder drei Argumente haben, die gültige BASIC/DEBUG-Ausdrücke sein müssen. Das erste Argument ist die Startadresse des aufgerufenen Maschinensprach-Programms. Die beiden nächsten, wahlfreien Argumente ermöglichen die Übergabe von Parametern an das Maschinensprach-Programm. Das Registerpaar %14 %15 erhält den Wert des zweiten Arguments und das Registerpaar %12 %13 den Wert des dritten Arguments. Fehlt das dritte Argument im GO@-Kommando, dann erscheint der Wert des zweiten Arguments in beiden genannten Registerpaaren. Das Maschinensprach-Programm muß mit der Z8-Instruktion RET (return) enden.

GOSUB

Syntax:

GOSUB Ausdruck

Beispiele:

```
GOSUB 50
GOSUB C
GOSUB 100*B
```

Das GOSUB-Kommando bewirkt einen Sprung in ein Unterprogramm. Der Wert des Ausdrucks muß eine im Programm vorhandene Zeilennummer ergeben. Im Gegensatz zu den üblichen BASIC-Dialekten kann die Zeilennummer auf diese Weise von Berechnungen und Ergebnissen abhängen, so daß sich ein eigenes ON...GOSUB-Kommando erübrigt.
Beim Ausführen des GOSUB-Kommandos merkt sich BASIC/DEBUG die Stelle, von der das Unterprogramm aufgerufen wurde, im Stapelspeicher. Beim abschließenden RETURN-Kommando des Unterprogramms wird diese Adresse verwendet, und es wird in der Programm-Ausführung mit dem Kommando fortgefahren, das dem GOSUB-Kommando folgt. Ein Unterprogramm kann seinerseits mit Hilfe des GOSUB-Kommandos ein weiteres Unterprogramm aufrufen und dieses wiederum weitere. Die Tiefe der Verschachtelung der Aufrufe ist lediglich durch den Speicherplatz BERTs begrenzt, weil mit jedem Aufruf der Stapelspeicher um eine Sprungadresse erweitert wird. Zum Stapelspeicher vgl. *Bild 9.3* auf Seite 193.

Wert des Pegels an der Prüfspitze e direkt an, und L1 zeigt den dazu negierten Pegelwert an. Von den beiden Leuchtdioden L1 und L2 leuchtet also bei L-Pegel nur L1 und bei H-Pegel nur L2; siehe *Bild F.12*.

Signal an e	L1	L2	L3	L4
0	✳	○	✳	○
0	✳	○	○	✳
1	○	✳	✳	○
1	○	✳	○	✳
1/0 ⊓⊓	+	+	+	+
1/0 ⊓⊓	✳	○	+	+
1/0 ⊓⊓	○	✳	+	+

✳ LED hell

\+ LED halb-hell

○ LED dunkel

Bild F.12: Die Zuordnung der am Logik-Prüfstift ausgegebenen Leuchtmuster zu den Eingangs-Signalen (vgl. Text).

Nach dem ersten Inverter gelangt das Signal außerdem an den Takteingang eines D-Flip-Flops, an dessen Ausgängen Q bzw. $\overline{Q}$ die Leuchtdioden L3 und L4 liegen. Ein Wechsel des Eingangssignals von 0 nach 1 verursacht am Ausgang des Flip-Flops einen Signalwechsel. L3 nimmt dann den vorherigen Zustand von L4 an und umgekehrt. Bei einem statischen Signal an e leuchtet also immer nur eine der Leuchtdioden L3 und L4; dabei ist es für die Beurteilung des Pegels an e unwesentlich, welche von beiden leuchtet.
Das ändert sich, wenn eine Pulsfolge an der Prüfspitze e anliegt. Nun wechseln die Leuchtdioden L3 und L4 bei jeder 1-0-Flanke an e ihren Zustand; d.h. sie blinken gegenphasig zueinander, und zwar mit der halben Frequenz des angelegten Signals. Bei hohen Frequenzen kann das Auge diesen Wechsel nicht mehr erkennen; dann scheinen L3 und L4 gleichzeitig mit der Hälfte ihrer normalen Helligkeit zu leuchten.
Auch L1 und L2 blinken bei einer Pulsfolge an e abwechselnd, allerdings mit der echten Signalfrequenz. Doch im Gegensatz zu L3 und L4 erscheinen diese beiden Dioden nur bei einem symmetrischen Signal (mit gleichlangen Pulsen und Pausen) gleich hell. Ist die Dauer der H-Impulse sehr klein im Vergleich zur Dauer der L-Impulse, so wird L1 viel heller als L2 leuchten. Umgekehrt wird man eine Folge von langen H-Impulsen und kurzen L-Impulsen daran erkennen, daß L1 praktisch nicht leuchtet, L2 dagegen recht hell; siehe *Bild F.12*.

Vorschläge zum Aufbau von Experimentier-Schaltungen

Für den konkreten Aufbau der vorgestellten Schaltungen gibt es viele Möglichkeiten. Sie reichen vom „fliegenden" Aufbau bis zum fertigen Bausatz aus dem DIGIPROB®-System der vgs, und jeder Praktiker hat seine bevorzugte Methode. Die folgenden drei Beispiele können daher nur als Anregung dienen.

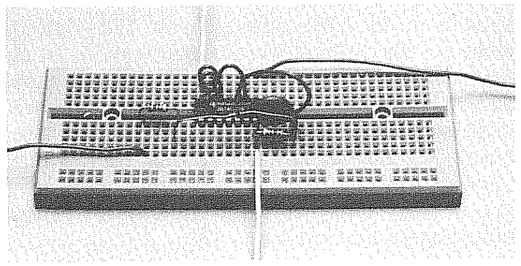


Bild 13: „Fliegender Aufbau" des 50-Hz-Generators nach *Bild 3.5* auf einem Experimentier-Steckbrett.

Ohne Löten, einfach und schnell, lassen sich Schaltungen auf einem *Experimentier-Steckbrett* (auch Experimenter-Board genannt) aufbauen. *Bild F.13* zeigt den Aufbau des *50-Hz-Generators* nach *Bild 3.5* in dieser Technik. In den Löchern des Steckbrettes befinden sich Kontakte, die jeweils in senkrechten Reihen miteinander verbunden sind. Alle Bauelemente wie Widerstände, Kondensatoren, Leuchtdioden, Transistoren, ICs mit oder ohne Sockel, DIL-Schalter, aber auch handelsübliche Drähte (0,6 mm Stärke) können direkt in diese Kontakte gesteckt und damit untereinander verbunden werden. Für vorübergehende

Aufbauten, insbesondere also für die Schaltungs-Entwicklung, bei der man des öfteren Bauelemente austauschen möchte, ist das Experimentier-Steckbrett ideal. Praktisch alle Experimentier-Schaltungen dieses Buches lassen sich auf dem Steckbrett ausprobieren – mit einem 8fach-DIL-Schalter sogar eine 8-Bit-Eingabe.
Dauerhafter und mit weniger „Drahtverhau" kann man Schaltungen auf *Lochraster-Platinen* aufbauen. Die *5 × 7-Punktmatrix-Anzeige* nach *Bild 6.13* bzw. *Bild F.7* wurde z. B. in dieser Technik aufgebaut: *Bild F.14*.

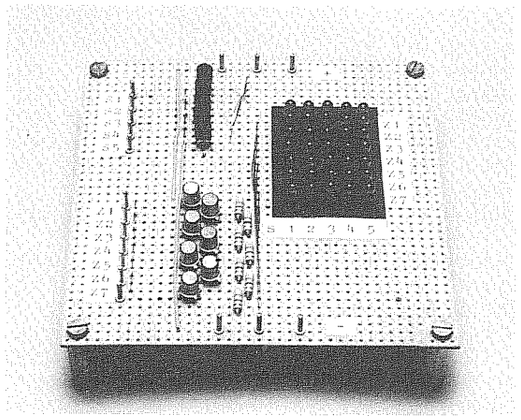


Bild F.14: Dauerhafter Aufbau der 5 × 7-Punktmatrix-Anzeige auf einer Lochrasterplatine. Wie bei BERT werden für die Anschlüsse 1,3 mm starke Lötnägel verwendet.

Je nach System werden die Bauteile auf der Rückseite mit durchgehenden Leiterbahnen oder einzelnen Leiterpunkten verlötet. Sofern sie nicht auf einer durchgehenden Leiterbahn liegen, müssen die Bauteile untereinander mit Drähten verbunden werden. Für dieses „Stricken" von Schaltungen ist die althergebrachte *Fädel-Technik* mit dünnen, lackisolierten Kupferdrähten bestens geeignet. Etwas eleganter geht es mit der sogenannten *Wire-Wrap-Technik*, für die allerdings neben dem Wire-Wrap-Draht auch ein spezielles Werkzeug erforderlich ist. Die Leiterbahnen selbst können, je nach Bedarf, an beliebigen Stellen mit einem Messer oder einem käuflichen Leiterbahn-Unterbrecher aufgetrennt werden. Mit dieser Methode kann man mit verhältnismäßig geringem Aufwand dauerhafte Schaltungen aufbauen und sich auf diese Weise nach und nach ein eigenes System von Experimentier-Platinen zusammenstellen.
Sowohl die Stromversorgung wie auch die Signal-Eingänge und -Ausgänge der Experimentier-Platinen sollten, wie bei BERT, auf 1,3 mm starke Lötnägel geführt werden. Zur Verschaltung der Platinen untereinander und mit BERT kann man dann einfache Drähte mit angelöteten Steckschuhen verwenden. Diese Methode ist sehr preiswert und sicher; sie hat sich in der Praxis bestens bewährt.

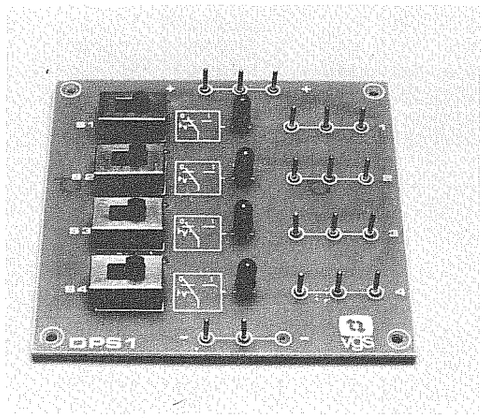


Bild F.15: Die 4fach-Eingabe-Einheit DPS 1 des DIGIPROB®-Systems. Die Platine kann mit allen erforderlichen Bauteilen als Bausatz bezogen werden.

Zum Abschluß sei nochmals auf die Bausätze des DIGIPROB®-Experimentiersystems der vgs verwiesen; vgl. auch die Bezugsquellen Hinweise in *Anhang H*. Alle DIGIPROB®-Bausätze enthalten die zur Funktion benötigten Bauelemente sowie eine vorgebohrte Platine mit Positionsaufdruck zur leichten Bestückung. Beigefügt ist eine ausführliche Aufbauanleitung mit Hinweisen zur digitaltechnischen Funktion.
Folgende DPS-Bausätze sind derzeit (Ende 1986) erhältlich:

4fach-Eingabeeinheit DPS 1
Prellfreie Taste mit zwei Stellschaltern DPS 2
Halterung mit Stecksockel für ICs DPS 3
Halterung für diskrete Bauelemente DPS 4
6fach-Ausgabeeinheit DPS 5
Netzteil (5 V, 1 A) DPS 6
Zwei Sensortasten DPS 8
Motor-Steuereinheit DPS 9
4fach-Pegelwertanzeige (H, L) DPS 12

4fach Binäranzeige (0, 1) DPS 13
Netzteil (12 V, 2 A) DPS 14

DPS-Kabelsatz
30 Kabel à 0,15 m und 60 Stecker

DPS-Sechser-Set
Zwei DPS 3 und je ein DPS 1, 2, 5 und 6

DPS-Vierer-Set
Je ein DPS 1, 3, 5 und 6

G Der ASCII-Code und der EBCDIC-Code

Die nachstehende Tabelle enthält den kompletten ASCII-Code aller Zeichen in hexadezimaler, dezimaler und binärer Darstellung. Zu den nicht druckbaren Steuerzeichen siehe *Bild 4.18*.

Zeichen	ASCII-Code			Zeichen	ASCII-Code			Zeichen	ASCII-Code			Zeichen	ASCII-Code		
	hex.	dez.	binär		hex.	dez.	binär		hex.	dez.	binär		hex.	dez.	binär
NUL	0	0	0000000	SP	20	32	0100000	@ §	40	64	1000000	`	60	96	1100000
SOH	1	1	0000001	!	21	33	0100001	A	41	65	1000001	a	61	97	1100001
STX	2	2	0000010	"	22	34	0100010	B	42	66	1000010	b	62	98	1100010
ETX	3	3	0000011	#	23	35	0100011	C	43	67	1000011	c	63	99	1100011
EOT	4	4	0000100	$	24	36	0100100	D	44	68	1000100	d	64	100	1100100
ENQ	5	5	0000101	%	25	37	0100101	E	45	69	1000101	e	65	101	1100101
ACK	6	6	0000110	&	26	38	0100110	F	46	70	1000110	f	66	102	1100110
BEL	7	7	0000111	'	27	39	0100111	G	47	71	1000111	g	67	103	1100111
BS	8	8	0001000	(	28	40	0101000	H	48	72	1001000	h	68	104	1101000
HT	9	9	0001001	)	29	41	0101001	I	49	73	1001001	i	69	105	1101001
LF	0A	10	0001010	*	2A	42	0101010	J	4A	74	1001010	j	6A	106	1101010
VT	0B	11	0001011	+	2B	43	0101011	K	4B	75	1001011	k	6B	107	1101011
FF	0C	12	0001100	´	2C	44	0101100	L	4C	76	1001100	l	6C	108	1101100
CR	0D	13	0001101	-	2D	45	0101101	M	4D	77	1001101	m	6D	109	1101101
SO	0E	14	0001110	.	2E	46	0101110	N	4E	78	1001110	n	6E	110	1101110
SI	0F	15	0001111	/	2F	47	0101111	O	4F	79	1001111	o	6F	111	1101111
DLE	10	16	0010000	0	30	48	0110000	P	50	80	1010000	p	70	112	1110000
DC1	11	17	0010001	1	31	49	0110001	Q	51	81	1010001	q	71	113	1110001
DC2	12	18	0010010	2	32	50	0110010	R	52	82	1010010	r	72	114	1110010
DC3	13	19	0010011	3	33	51	0110011	S	53	83	1010011	s	73	115	1110011
DC4	14	20	0010100	4	34	52	0110100	T	54	84	1010100	t	74	116	1110100
NAK	15	21	0010101	5	35	53	0110101	U	55	85	1010101	u	75	117	1110101
SYN	16	22	0010110	6	36	54	0110110	V	56	86	1010110	v	76	118	1110110
ETB	17	23	0010111	7	37	55	0110101	W	57	87	1010111	w	77	119	1110111
CAN	18	24	0011000	8	38	56	0111000	X	58	88	1011000	x	78	120	1111000
EM	19	25	0011001	9	39	57	0111001	Y	59	89	1011001	y	79	121	1111001
SUB	1A	26	0011010	:	3A	58	0111010	Z	5A	90	1011010	z	7A	122	1111010
ESC	1B	27	0011011	;	3B	59	0111011	[Ä	5B	91	1011011	{ ä	7B	123	1111011
FS	1C	28	0011100	<	3C	60	0111100	\ Ö	5C	92	1011100	| ö	7C	124	1111100
GS	1D	29	0011101	=	3D	61	0111101	] Ü	5D	93	1011101	} ü	7D	125	1111101
RS	1E	30	0011110	>	3E	62	0111110	^	5E	94	1011110	~ ß	7E	126	1111110
US	1F	31	0011111	?	3F	63	0111111	_	5F	95	1011111	del	7F	127	1111111

Van der Waerden, B.L.: Die Pythagoreer, Religiöse Bruderschaft und Schule der Wissenschaft. Artemis, 1979.
Warnecke, H., Schraft, R.: Industrieroboter. VDI, 1979.
Wirth, N.: Systematisches Programmieren. Teubner, 1982.
Wiegand, M.: Der MACCOM 2000. In: elo 7/1985, 9/1985, 10/1985, Franzis.
Wiesemann, R.: Basic-Einplatinen-Computer. In: mc 2/1983, mc 3/1983, Franzis.
Zaks, R.: Chip und System: Einführung in die Mikroprozessoren-Technik. Sybex, 1983.
Zilog (Hrsg.): Z8671 Seven Chip Computer. 1981.
Zilog (Hrsg.): Z8671 Single Chip BASIC Interpreter, BASIC/DEBUG Software Reference Manual. 1981.
Zilog (Hrsg.): Z8 Microcomputer, Technical Manual, 1983.
Zilog (Hrsg.): Z8 PLZ/ASM Assembly Language Programming Manual, 1980
Zilog (Hrsg.): Z8 Single Chip Microcomputer Technical Manual. 1980.

Bezugsquellen

BERT kann (fertig verlötet und geprüft) außer im Fachhandel auch bezogen werden bei der

vgs-Verlagsgesellschaft
Breite Straße 118/120
5000 Köln 1

Für die im Anhang F aufgeführten Bausätze des DIGIPROB®-Experimentiersystems gilt die gleiche Bestelladresse. Der Versand kann leider nur per Nachnahme erfolgen. Über nähere Einzelheiten informiert Sie der Sonderprospekt „Naturwissenschaft und Technik", den Sie bei der vgs anfordern können.

Der Z8-Miniassembler ist zu beziehen bei:

Bernd Holzhauer
Ing.-Büro für Mikrocomputertechnik
Herderstr. 12
8057 Eching

I Register der Programme