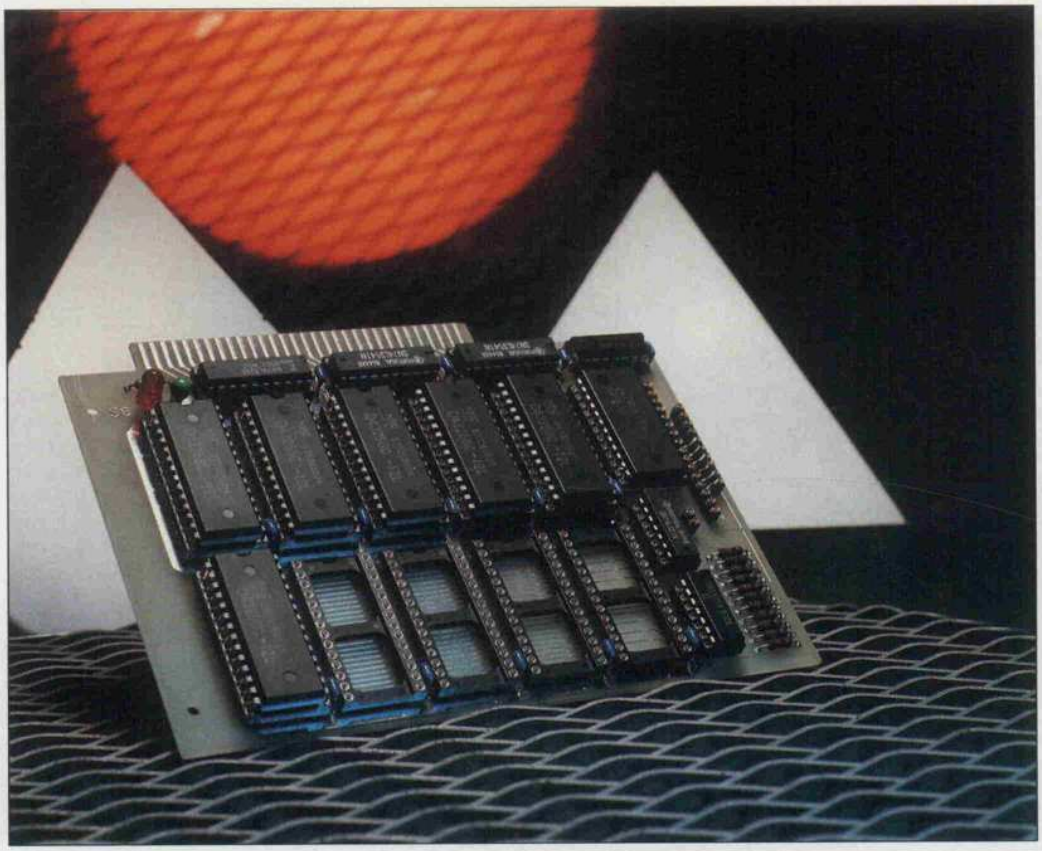


Zwischen den Adaptern

Speichererweiterung plus Softwareunterstützung für PCs

Peter Köhlmann, Gerhard Rubel, Michael Wilde

Oberhalb der berüchtigten 640-K-Grenze in PCs gibt es trotz der Reservierung für Bildschirmkarten und Zusatz-EPROMs noch Freiräume für Speichererweiterungen. Die hier vorgestellte PC-Slot-Karte fügt sich mit ihren CMOS-Bausteinen genau in die Lücken zwischen Adaptern und BIOS ein. Damit können Sie den DOS-Bereich vergrößern, eine resetfeste RAM-Disk einrichten und einiges mehr.

Der Speicher eines MSDOS-Rechners muß nicht an der 640-KByte-Grenze enden. Die Prozessoren 8088 und 8086 haben schließlich über ihre 20 Adreßleitungen Zugriff auf insgesamt 1 MByte RAM. Die CPU des AT (80286) ist zwar darüber hinaus in der Lage, eine erheblich erweiterten Speicherbereich zu erschließen, solange das Betriebssystem sie aber nur im zu den 80xx-Prozessoren kompatiblen Real Mode betreibt, muß auch sie sich mit dem 1-MB-Speicherabbild begnügen.

Daß dennoch fast alle XTs nur bis auf 640 KByte auszurüsten sind, liegt an den Adreßvorgaben der großen Vorbilder von IBM, bei denen der restliche Bereich für Systemzwecke reserviert ist. Dazu gehören das BIOS-EPROM und zumindest ein Bildschirm-Adapter. Trotzdem bleiben je nach Systemausbau Blöcke unterschiedlicher Größe ungenutzt, die man für eigene Erweiterungen heranziehen kann.

Zum Beispiel schließt sich direkt hinter der 640-K-Grenze der Bereich für EGA-Karten an. Ist der Rechner nur mit einer Monochrom-, Hercules- oder Farbgrafikkarte ausgestattet, läßt sich dieser Bereich direkt für das DOS nutzen.

Lassen Sie sich jedoch nicht von der Aussage 'Motherboard mit 1 MByte bestückt' verwirren, die man besonders bei ATs findet, denn auch dort ist der vom DOS nutzbare Systemspeicher meistens auf 640 K begrenzt. Der Rest wird als Extended Memory genutzt, also hauptsächlich für RAM-Disks. Einige Rechner teilen ihren Hauptspeicher so ungeschickt auf, daß sogar nur 512 KByte für das Betriebssystem übrigbleiben und die restlichen 512 K nur als RAM-Disk zu verwenden sind. Auch hier hilft unsere Karte weiter, obwohl wir bei der Vielzahl von verschiedenen Motherboard-Architekturen nicht garantieren können, daß sich die Karte in jeden Rechner einblenden läßt. Im Einzelfall kommt es dann auf einen Versuch an.

Im Prinzip eignet sich die Karte für den Betrieb in jedem IBM-kompatiblen Rechner mit mindestens einem freien PC-Slot. Dazu gehören PC-, XT- und AT-Modelle der verschiedensten Hersteller. Selbst in IBMs Modell 30 läßt sie sich einsetzen. Besitzer des Schneider PC1512 sollten noch ein bißchen warten, weil wir speziell für diesen Rechner eine erweiterte Version in Planung haben, die neben den hier beschriebenen Funktionen dafür sorgen soll, daß man auch diesen Rechner konfliktfrei mit einer EGA- oder Hercules-Karte betreiben kann.

Bevor Sie irgendwelche RAM-Erweiterungen in Ihren Rechner einstecken, sollten Sie sich ein genaues Bild von der derzeitigen Speicherbelegung machen. Parallelgeschaltete Speicher vertragen sich nicht besonders gut miteinander. Dieser Artikel beschäftigt sich daher noch eingehend mit der Frage, welche Adreßblöcke im Systembereich unbenutzt sind. Doch vorerst geht es um die Hardware der Zusatzkarte.

Entwicklungskriterien

Um möglichst vielen PC-Besitzern die Aufrüstung mit wenig Aufwand zu ermöglichen, haben wir uns bei der hier vorgestellten Speichererweiterung für eine Slot-Karte entschieden. Daß es andere Wege gibt, zeigt der Beitrag [1], dessen Lösung des Speicherplatzproblems jedoch nicht ohne Löterei auf dem Motherboard auskommt.

Besonders die hardwareerprobten Leser mag es beim Betrachten des Schaltbildes vielleicht stören, daß wir auf die im Vergleich zu dynamischen RAMs etwas teureren CMOS-RAMs zurückgegriffen haben. Durch diese Lösung ist jedoch eine hohe Aufbau- und Betriebssicherheit gewährleistet, da sämtliche Auffrischlogik entfällt und es auch in Turbo-Rechnern kaum zu Timing-Problemen kommt.

Die Karte läßt sich an die individuellen Ausstattungsverhältnisse in Schritten zu 32 KByte anpassen. Jedes CMOS-RAM stellt genau einen Speicherblock dieser Größe zur Verfügung, der immer an einer 32-KByte-Grenze beginnt. Da insgesamt 10 RAM-Bausteine vorgesehen sind, kommt man auf maximal 320 KByte. Nicht viel im Vergleich zu 2 oder 4 MByte Extended Memory, aber haben Sie schon mal versucht, speicherresidente Programme im Extended Memory unterzubringen?

Auf eine Parity-Logik haben wir ganz verzichtet, weil wir davon ausgehen, daß CMOS-RAMs eine hohe Störsicherheit bieten und nicht zu Datenverlusten neigen.

Außer den Puffern für Adreß- und Datenbus sowie einem Adreßdecoder benötigt man nur einige CMOS-RAMs, um auch oberhalb der 640-K-Grenze Speicher zu installieren. Bis auf die $\overline{CS}$-Signale sind an den RAMs alle Anschlüsse parallelgeschaltet.

Wenige Chips

Mal keine Chip-Ungeheuer von Chips&Technologies, sondern TTL-Hausmannskost, wie man sie in jedem Bastelladen findet: die Schaltung besteht neben den Speicherbausteinen aus Daten- und Adreßbuspuffern und einer Adreßlogik.

Die CMOS-RAMs vom Typ 43256 oder 62256 sind zu 32 K × 8 Bit organisiert, brauchen also die 15 Adreßleitungen A0 bis A14, um sämtliche Adressen ansprechen zu können. Diese Adreßleitungen und die Datenleitungen DB0 bis DB7 werden parallel an jedes RAM-IC geführt. Die genaue Lage des Speicherbausteins im Adreßraum bestimmt die Adreßlogik durch das $\overline{CS}$-Signal (Chip Select), das heißt, jedes IC bekommt eine eigene Chip-Select-Leitung, die immer dann auf logisch Null geht, wenn dieser Chip angesprochen wird.

J1	J3 geschlossen	J3 offen
1	0000:0h	8000:0h
2	0800:0h	8800:0h
3	1000:0h	9000:0h
4	1800:0h	9800:0h
5	2000:0h	A000:0h
6	2800:0h	A800:0h
7	3000:0h	B000:0h
8	3800:0h	B800:0h
9	4000:0h	C000:0h
10	4800:0h	C800:0h
11	5000:0h	D000:0h
12	5800:0h	D800:0h
13	6000:0h	E000:0h
14	6800:0h	E800:0h
15	7000:0h	F000:0h
16	7800:0h	F800:0h

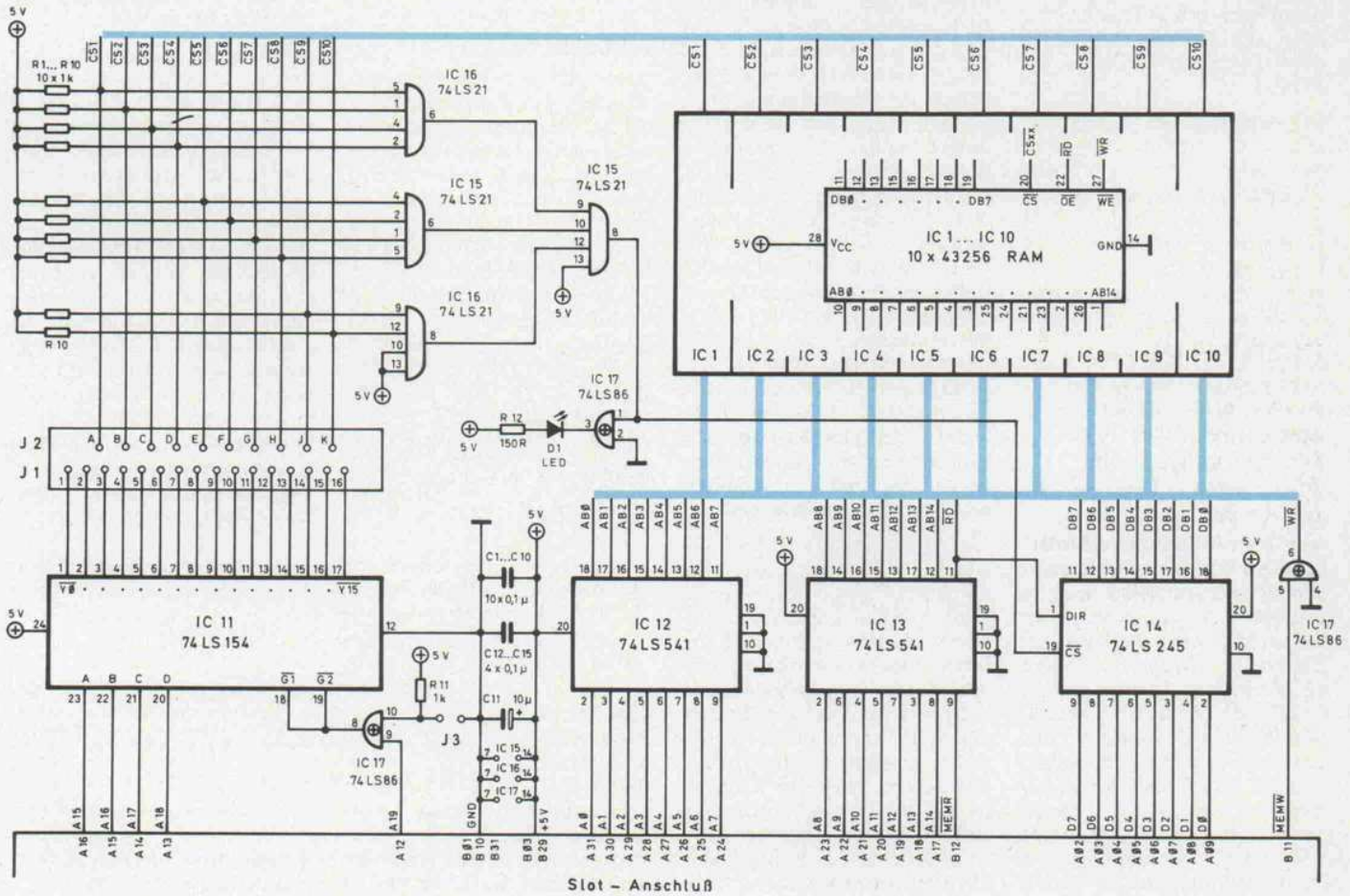
Jumper J3 teilt den erreichbaren Speicher in zwei Teile auf. Zur Verwendung zwischen den Adaptern muß er offenbleiben.

Das Umschalten zwischen Lesen und Schreiben erledigen die beiden Signale an $\overline{OE}$ (Output Enable) und $\overline{WE}$ (Write Enable), die ebenfalls parallel gelegt werden. Bei Rechnern mit einer Taktfrequenz bis zu 8 MHz reichen Bausteine mit 150 ns Zugriffszeit völlig aus. Nur bei sehr schnellen ATs (ab 16 MHz) sollte man an den Einsatz von ICs mit 120 ns denken.

Der Adreßdecoder (74LS154) teilt abhängig von der Stellung des Jumpers J3 die obere oder untere Hälfte des Arbeitsspeichers in die gewünschten Blöcke zu 32 KByte auf. Soll die Karte im oberen Systembereich Verwendung finden, muß J3 offenbleiben, damit die Adreßleitung A19 durchgeschaltet wird. Das Signal $\overline{Y0}$ selektiert den ersten Speicherblock, der dann an der Adresse 8000:0000h beginnt, also noch mitten im DOS-Bereich. Der nächste Block startet bei 8800:0000h, der übernächste bei 9000:0000h. Man kann die Karte also wie eben beschrieben auch als DOS-Speichererweiterung verwenden. Das gilt besonders, wenn J3 geschlossen ist, da der erste Block dann an der Adresse 0000:0000 beginnt.

DOS-RAM

Ältere Rechner, die auf dem Motherboard nur bis



Stückliste

Halbleiter

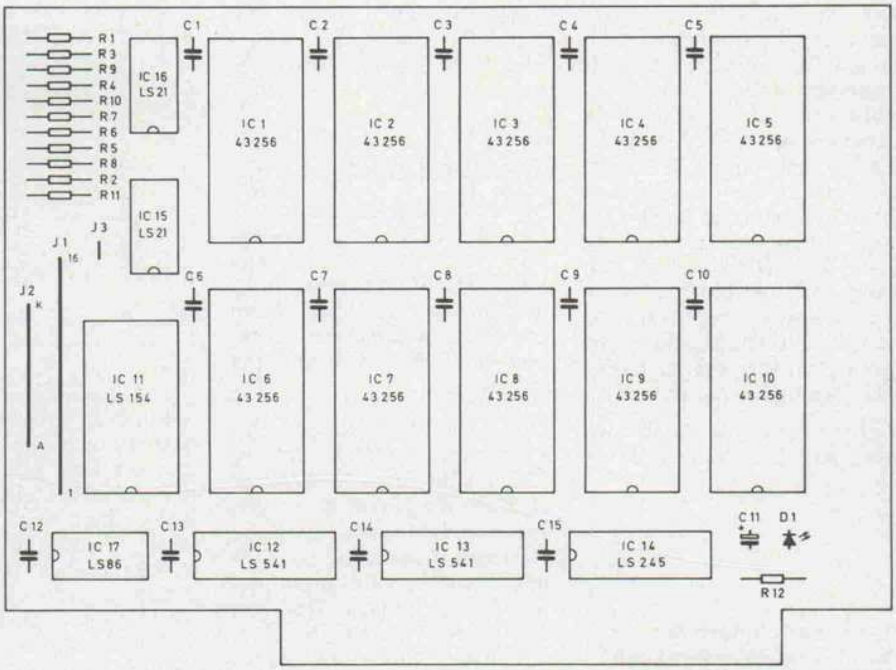
IC1-10	43256 oder 62256 150 ns bis 8 MHz 120 ns über 8 MHz
IC11	74LS154
IC12,13	74LS541
IC14	74LS245
IC15,16	74LS21
IC17	74LS86
D1	LED

Widerstände/Kondensatoren

C1-10, C12-15	100 nF/16 V
C11	10 µF/16 V, Tantal
R1-11	1k
R12	150

Sonstiges

10	IC-Fassungen, 28polig
1	IC-Fassungen, 24polig
3	IC-Fassungen, 20polig
3	IC-Fassungen, 14polig



256 KByte auszubauen sind, lassen sich mit einer CMOS-RAM-Karte auf 512 KByte aufrüsten, wenn man acht Speicherbausteine einsetzt, die Brücke J3 offenläßt und den Bereich 4000h bis 7FFFh jumpert. Mit einer zweiten Karte mit offener Brücke J3 kann man dann den Rechner auf 640 K oder sogar noch weiter bestücken.

Die Ausgänge des Adreßdecoders IC 11 verbindet man über die Leisten J1 und J2 mit den $\overline{CS}$-Leitungen der RAMs, wobei IC 1 an J2/A liegt, IC 2 an J2/B und so fort. Über einige UND-Gatter gewinnt die Schaltung ein gemeinsames Chip-Select-Signal für den bidirektionalen Datenbustreiber IC 14, das heißt, der Treiber wird nur dann aktiviert, wenn der Prozessor einen Baustein auf der Karte anspricht.

Vielleicht ist Ihnen schon die kleine LED auf der Platine aufgefallen. Für den Betrieb ist sie zwar nicht unbedingt erforderlich, leistet aber unter Umständen bei der Fehlersuche gute Dienste, weil sie immer dann aufleuchtet, wenn die Karte selektiert wird. Wenn Sie Freude an blinkenden Lämpchen haben, lassen Sie sie drin, ansonsten kann sie zusammen mit dem Widerstand R12 entfallen. Das gilt jedoch nicht für IC 17, denn das wird noch an anderen Stellen gebraucht. Besonders schön macht sie sich natürlich an der Frontplatte des Rechners. Warum sollen denn eigentlich immer nur Zugriffe auf Diskette oder Festplatte angezeigt werden?

Beim Aufbau der Schaltung dürfte es nicht zu Problemen kommen, wenn Sie die fertige doppelseitige durchkontaktierte Platine benutzen. Zumindest den RAMs sollten Sie Fassungen spendieren, damit Sie jederzeit die Bestückung ändern können. Bei der Inbetriebnahme sollten Sie besonders darauf achten, daß Sie die Karte nicht falsch herum einstecken, denn das würde für die meisten Bausteine tödlich enden. Die LED muß sich im Standard-PC direkt an der Gehäuserückseite befinden. Um ein Verdrehen auszuschließen, sollte man eins der Slot-Bleche über kleine Winkel an der Platine befestigen. An der zur Rechnerrückwand gelegenen Platinenseite finden Sie daher zwei kleine Bohrungen, die zur Montage der Winkel vorgesehen sind.

Die Anschlüsse der beiden Leisten J1 (1. . .16) und J2 (A. . .K) verbindet man über kleine Drahtbrücken miteinander. Um keinen Zweifel aufkommen zu lassen: diese Verbindungen sind sehr stark vom individuellen Ausbau Ihres Rechners abhängig, so daß wir hier kein Patentrezept angeben können. Aber mal angenommen, Ihr Rechner ist auf dem Mainboard schon auf 640 KByte ausgerüstet, besitzt aber keine EGA-Karte, dann brauchen Sie zwei RAM-Bausteine, die zum Beispiel in die Fassungen IC 1 und IC 2 kommen, um den Rechner auf 704 KByte Gesamtspeicher aufzurüsten. Die Brücke J3 bleibt offen, und Sie verbinden Anschluß J1/5 mit J2/A und J1/6 mit J2/B.

Auf der Suche

Bevor Sie sich endgültig entscheiden, welche Speicherbereiche Sie bestücken, kommen Sie nicht umhin, Ihren Rechner genauestens nach unbestückten Bereichen zu untersuchen. In einer Aufstellung finden Sie die verbreitetsten Komponenten für den oberen Systembereich, wie man sie in unterschiedlichen Zusammenstellungen in vielen Rechnern findet. Die 64 KByte ab Segment A000h sind im PC, XT und AT für die EGA-Karte reserviert, im Modell 30 liegt dort das 'Multi Colour Graphics Array' (MCGA). Dieser Bereich kann also in PCs immer dann bestückt werden, wenn keine EGA vorhanden ist; im Modell 30 ist er tabu. Sowohl EGA-Karten wie auch das MCGA belegen zusätzlich den Bereich B000h bis BFFFh. Die Monochrom-Karte (MDA) liegt in B000h, ebenfalls die Hercules-Karte, die zusätzlich ab B800h zu finden ist.

Die Hercules-Karte läßt sich in drei verschiedenen Modi betreiben. Wenn man auf die zweite Grafikseite verzichtet und nur im Half-Modus arbeitet, blendet sie normalerweise in B800h keinen Speicher ein, so daß man diesen Bereich mit RAM bestücken könnte. Diese Belegung ist aber mit Vorsicht zu genießen, da man nie ausschließen kann, daß fremde Software die Karte in den Full-Modus schaltet und damit Speicherkollisionen provoziert. Außerdem gibt es Hercules-kompatible Karten, die immer den gesamten Speicher eingeschaltet haben. Bei Bestückung mit einer Hercules-Karte sollte also immer der ge-

```
A000    EGA, MCGA
A800    EGA, MCGA
B000    MDA, Hercules, EGA, MCGA
B800    CGA, Hercules, EGA, MCGA
C000    EGA-BIOS
C800    Festplatten-BIOS
D000    (EP)ROM-Erweiterungen, Expanded Memory
D800    (EP)ROM-Erweiterungen, Expanded Memory
E000    (EP)ROM-Erweiterungen, Expanded Memory
E800    (EP)ROM-Erweiterungen, Expanded Memory
F000    BASIC-ROMs
F800    BASIC-ROMs, BIOS
```

IBM hat den Adreßraum ab Segment A000h für verschiedene Erweiterungen vorgesehen. Ist der Rechner nicht mit diesen Features ausgestattet, bietet er Platz für zusätzlichen Speicher.

samte Bereich von B000h bis BFFFh frei bleiben.

Wenn der Rechner nur mit einer Farbgrafikkarte ausgestattet ist, die immer bei B800h liegt, gewinnt man auf einen Schlag 96 KByte fürs DOS dazu. EGA-Karten brauchen nicht nur viel Grafikspeicher, sondern blenden zusätzlich ein eigenes BIOS ein, das anders als zum Beispiel die Hercules-Karte die Programmierung der im Vergleich zur CGA verbesserten Grafik über das BIOS erlaubt.

IBM hat den Bereich von C000h bis F400h für BIOS-Erweiterungen vorgesehen, die während des Bootens vom Original-BIOS eingebunden werden. Die wichtigsten Startadressen dafür sind C000h für das EGA-BIOS und C800h für den Festplatten-Controller in XTs. Der AT enthält die Harddisk-Funktionen schon im Standard-EPROM und belegt damit keinen zusätzlichen Speicher in C800h; dies gilt aber nur, wenn tatsächlich ein AT-Controller eingebaut ist. Da einige XT-Controller auch im AT funktionieren, kann man dort nicht absolut sicher sein, daß ab C800h kein BIOS vorhanden ist.

Der Speicherbereich zwischen F000h und FFFFh sollte auf keinen Fall mit eigenem RAM bestückt werden, da sich dort am oberen Ende immer das BIOS befindet und davor die BASIC-ROMs liegen können.

Systemanalysator

Für einen schnellen Überblick über die Bestückung Ihres Rechners haben wir ein Turbo-C-Programm abgedruckt, das die Adreßbelegung in Schritten zu 32 KByte analysiert und grafisch darstellt. Jedem Abschnitt entspricht also genau ein RAM-Baustein auf der Speichererweiterungskarte. Die Vorgehensweise des Programms gibt wichtige Hinweise darauf, wie man eine Analyse von Hand durchführen könnte.

Das Programm stellt als erstes den Speicherausbau des Computers fest, der auch dem Betriebssystem bekannt ist, wobei es bei einigen Rechnern durchaus zu Differenzen zwischen dem physikalisch vorhandenen und dem vom DOS verwalteten Speicher kommen kann.

Anschließend ermittelt es, welche Bildschirm-Adapter sich im System befinden, allerdings mit der Beschränkung auf die schon vorgestellten Standardkarten. Sollte in Ihrem Rechner irgendeine Spezialkarte vorhanden sein, bekommen Sie trotzdem Hinweise auf ihre Existenz, da später zusätzlich auf vorhandenes RAM getestet wird.

Die Vorgehensweise bei der Ermittlung von Grafikkarten könnte natürlich erheblich verfeinert werden, aber es geht an dieser Stelle nur darum, festzustellen, ob ein Adapter eingesteckt ist oder nicht. Dazu benutzt das Hauptprogramm 'main()' die Funktion 'adaptervorhanden()', die auf die Frage nach der Präsenz einer der zu Beginn der Datei definierten Bildschirmkarten mit 'true' oder 'false' antwortet. Das Programm geht der Reihe nach alle bekannten Karten durch und merkt sich die belegten Speicherbereiche in dem Array namens 'block'. Das sequentielle Abarbeiten ist deshalb notwendig, weil einige Karten parallel liegen dürfen, wie zum Beispiel Monochrom- und Farbgrafikkarte.

Nun testet eine Schleife den Bereich von C000h bis F400h in Abständen von 2 KByte auf BIOS-Erweiterungen, die sie an der Byte-Kombination 55AAh erkennt. Wenn das Programm eine solche Kennung findet, geht es davon aus, daß es sich um ein BIOS handelt, markiert das Segment als belegt und liest das nächste Byte, das eine Längenangabe in Schritten zu 512 Byte enthält. Vom Ende dieses Offsets an testet es weiter, bis die Adresse F400h erreicht ist, an der die BASIC-EPROMs beginnen.

Das Programm beschäftigt sich nun mit den obersten 64 KByte des Hauptspeichers, die es zu Beginn als BIOS- und BASIC-Bereich deklariert. Wenn Sie auf Ihrem Rechner in der untersten Zeile das Ergebnis 'BIOS/BASIC' erhalten, heißt das nicht automatisch, daß in Ihrem Rechner ein residentes BASIC vorhanden ist, sondern nur, daß dort eines sein könnte. In vielen Rechner findet man gespiegelte EPROMs, was Sie daran erkennen können, daß sich beispielsweise das BIOS nicht nur an Adresse FC00:0h bis FFFF:0h lesen läßt, sondern auch noch an F600:0h bis F9FF:0h. Falls ein solcher Test erfolgreich verläuft, sind auch diese Bereiche für Speichererweiterungen gestorben.

In einem letzten Speichertest geht das Programm sämtliche noch als frei markierten Bereiche durch und prüft bei jedem Byte, ob es sich nicht doch um RAM handelt. An dieser Stelle deckt es die angedeuteten Differenzen zwischen verfügbarem und dem DOS bekannten RAM auf.

Alles frei?

Der C-Kenner schlägt wahrscheinlich beim Betrachten der Funktion 'ramexist()' die Hände über dem Kopf zusammen, weil das Ganze so umständlich programmiert wurde. Diese 'ungeschickte' Programmierweise soll den Prozessor bremsen, denn es zeigte sich bei einigen Rechnern, daß der kompilierte Code so schnell war, daß er RAM an Stellen erkannte, die völlig unbestückt waren. Schreiben und Lesen erfolgten dabei derartig schnell aufeinander, daß der Prozessor den Buszustand las, den er selbst gerade erst hergestellt hatte, ohne daß auch nur ein RAM-Baustein diesen Zustand festhielt. In der Funktion wird nun über den Umweg einer Wertzuweisung an die Variable 'test' Zeit für eine Stabilisierung des Bus gewonnen.

Der RAM-Test liest ein Byte, invertiert es und schreibt es zurück, macht eine winzige Pause, liest das Byte erneut und vergleicht das geschriebene Byte mit dem gelesenen. Sind beide identisch, geht er davon aus, daß an dieser Stelle RAM vorhanden ist. Nach dem Test des Bytes invertiert die Routine den Inhalt ein weiteres Mal, um den alten Zustand wiederherzustellen.

Die folgende Routine hat zwar unmittelbar nichts mit der Speicherbelegung zu tun, gibt aber eine weitere allgemeine Information über den Computer. Das Programm holt aus dem sogenannten ID-Byte des BIOS eine verschlüsselte Angabe über den Computer-Typ. Zu guter Letzt gibt das Programm alle gewonnenen Informationen auf einen Schlag auf den Bildschirm aus.

Nachdem Sie das Programm im Small-Model kompiliert und gestartet haben, sollten Sie etwas Geduld mitbringen, denn es dauert (abhängig von der Rechenleistung) einige Sekunden, bis Sie die erste Ausgabe sehen werden. Besonders der RAM-Test schluckt dabei einiges an Zeit, da er unter Umständen einige hundert KByte bearbeiten muß.

Brücken legen

Wenn Ihnen nun mit Hilfe des eben vorgestellten Programms oder eigener Untersuchungen klargeworden ist, wo mit absoluter Sicherheit aufgerüstet werden kann, sollten Sie die Karte entsprechend bestücken. Die Belegung der Chip-Select-Leitungen verdeutlicht ein konkretes Beispiel: Das eben vorgestellte Programm SYSMAP liefert bei einem von uns untersuchten Micromint-AT das im Bild gezeigte Ergebnis. Der Rechner ist nur mit 512 KByte RAM bestückt, enthält eine Hercules-Karte und eine BIOS-Erweiterung, die von einem XT-Festplatten-Controller eingeblendet wird. Neben den für BIOS und BASIC reservierten 64 K blockiert das gespiegelte BIOS weitere 32 KByte.

```
Computer-Typ: AT / XT-286

0000: DOS --> RAM          0800: DOS --> RAM           64 KByte
1000: DOS --> RAM          1800: DOS --> RAM          128 KByte
2000: DOS --> RAM          2800: DOS --> RAM          192 KByte
3000: DOS --> RAM          3800: DOS --> RAM          256 KByte
4000: DOS --> RAM          4800: DOS --> RAM          320 KByte
5000: DOS --> RAM          5800: DOS --> RAM          384 KByte
6000: DOS --> RAM          6800: DOS --> RAM          448 KByte
7000: DOS --> RAM          7800: DOS --> RAM          512 KByte
8000: ...........          8800: ...........          576 KByte
9000: ...........          9800: ...........          640 KByte
A000: ...........          A800: ...........          704 KByte
B000: Hercules-Karte       B800: Hercules-Karte       768 KByte
C000: ...........          C800: BIOS-Erweiterung     832 KByte
D000: ...........          D800: ...........          896 KByte
E000: ...........          E800: BIOS gespiegelt      960 KByte
F000: BIOS/BASIC           F800: BIOS/BASIC          1024 KByte
```

Die Systembelegung eines vom Programm SYSMAP getesteten AT. Die Speicherkarte kann hier alle mit Punkten als frei deklarierten Bereiche auffüllen.

```
#include <stdio.h>
#include <bios.h>
#include <dos.h>

#define laenge 30
#define maxbloecke 32
#define CGA      1
#define MDA      2
#define Hercules 3
#define EGA      4

/* ---------------------------------------------------- */
/* Test, ob in numerierten Speicherblock RAM vorhanden */
/* ---------------------------------------------------- */

char ramexist(char blocknr)
{   unsigned int far *ram;
    unsigned int j,test;

    ram=MK_FP(blocknr*0x800,0);
    for (j=0; j<=0x3FFF; j++)
       {test=*ram=~*ram;
       if (test==*ram)
          {*ram=~*ram;
          return(1);}
       else
          ram++;}
    return(0);
    }

/* -------------------- */
/*  Video-Modus setzen  */
/* -------------------- */

void setmodus(char modus)
{  union REGS regs;

    regs.h.ah = 0;
    regs.h.al = modus;
    int86(0x10,&regs,&regs);
}

/* ------------------------------ */
/* Aktuellen Video-Modus ermitteln */
/* ------------------------------ */

char getmodus()
{   union REGS regs;

    regs.h.ah = 15;
    int86(0x10,&regs,&regs);
    return(regs.h.al);
}

/* ---------------------------- */
/* Bildschirm-Adapter ermitteln */
/* ---------------------------- */

char adaptervorhanden(char adapter)
{  unsigned int status;
   union REGS regs;
   struct time now;
   char result,hsek;

   switch (adapter){
   case EGA:  regs.x.ax = 0x1210; /* EGA-BIOS ansprechen */
              regs.x.bx = 0xFFFF; /* Testwert */
              int86(0x10,&regs,&regs);
              result=((regs.h.bh <= 1)&&(regs.h.bl <= 3));
              break;
   case CGA:  setmodus(7);
              result=getmodus() < 7;
              break;
   case MDA:
   case Hercules:
          setmodus(7);
          if (getmodus()==7)
             /* Unterscheidung zwischen MDA und Hercules
                Bit 7 des Statusports ändert sich auf der
                Hercules-Karte mit einer Frequenz von 50 Hz */
            {status=inport(0x3BA)&0x80;
             gettime(&now);
             hsek=now.ti_hund;
             do {if ((inport(0x3BA)&0x80)!=status)
                    {result=adapter==Hercules;
                     break;}
                 gettime(&now);
                 result=adapter==MDA;
             } while (now.ti_hund<hsek+10);}
          else
             result=0;
   }
   return(result);
}

/* ------------- */
/* Hauptprogramm */
/* ------------- */

void main()
{  int maxmem,i,j;
   char altmodus,*block[maxbloecke],*type;
   unsigned char far *bios;
   unsigned char far *bios1;
   unsigned int huge *biosext;
   unsigned char far *idbyte = (unsigned char far *) 0xF000FFFE;

   altmodus=getmodus();

   /* Ausgabe-Array löschen */
   for (i=0; i<maxbloecke; i++)
      block[i]="...........";

   /* Für DOS bekannten Speicher eintragen */
   maxmem=biosmemory()/maxbloecke;
   for (i=0; i < maxmem; i++)
      block[i]="DOS --> RAM";

   /*Bildschirm-Adapter ermitteln */
   if (adaptervorhanden(EGA))
      block[20]=block[21]=block[22]=block[23]="EGA-Karte  ";
   if (adaptervorhanden(Hercules))
      block[22]=block[23]="Hercules-Karte";
   if (adaptervorhanden(CGA))
      block[23]="Farbgrafik";
   if (adaptervorhanden(MDA))
      block[22]="Monochrom";

   /* Zusätzliche BIOS-EPROMs suchen */
   biosext=MK_FP(0xC000,0);
   do{
      if (*biosext==0xAA55){
         block[FP_SEG(biosext)/0x800]="BIOS-Erweiterung";
         biosext+= *(biosext+1)&0xFF<<9;}
      else
         biosext+=0x1000;
   }while (biosext < (unsigned int huge *) 0xF4000000);

   /* Größe des BIOS-EPROMs ermitteln */
   block[31]=block[30]="BIOS/BASIC ";

   /* Gespiegelte EPROMs suchen */
   bios=MK_FP(0xFC00,0);
   for (j=0; j<=4; j++)
      {bios1=MK_FP(0xD400+j*0x800,0);
      for (i=0; i<100; i++)
         if (*bios!=*bios1) break;
      if (i==100) block[26+j]="BIOS gespiegelt";}

   /* In freien Bereichen testen, ob RAM vorhanden */
   for (i=0; i<maxbloecke; i++)
      if (*block[i]=='.')
         if(ramexist(i))
            block[i]="RAM           ";

   /* Computer-Type aus ID-Byte ermitteln */
   switch (*idbyte){
      case 0xFF: type="PC            ";  break;
      case 0xFE:
      case 0xFB: type="PC XT         ";  break;
      case 0xFD: type="PCjr          ";  break;
      case 0xFC: type="AT / XT-286   ";  break;
      case 0xFA: type="Model 30      ";  break;
      case 0xF9: type="Laptop        ";  break;
      default:   type="unbekannt     ";}

   setmodus(altmodus);

   /* Ergebnis ausgeben */
   printf("Computer-Typ: %s\n\n",type);
   for (i=j=0; i < 16; i++,j+=64)
      {printf("%04X: %s\t\t%04X: %s\t%4d KByte \n",
          j*64,block[i*2],j*64+0x800,block[i*2+1],j+64);}
}
```

SYSMAP liefert einen schnellen Überblick über die Systembelegung, auch wenn man den Rechner nicht mit einer zusätzlichen RAM-Karte ausrüsten will.

Für die Speicherkarte sind das ideale Voraussetzungen, denn insgesamt sind 10 Blöcke à 32 KByte frei. Das DOS läßt sich mit 6 ICs auf 704 KByte zusammenhängend aufrüsten. Leider sind die restlichen 128 KByte vom Controller-BIOS unterbrochen, so daß man sie nicht in einem Stück verwalten kann, beispielsweise mit der am Ende des Artikels abgedruckten RAM-Disk, aber immerhin läßt sich eine 96-KB-RAM-Disk von D000h bis E7FFh aufbauen.

Die Verdrahtung dürfte nun eigentlich kein Problem sein, wenn man das Ergebnis von SYSMAP mit der Tabelle zur Adreßdekodierung vergleicht. Die Karte arbeitet oberhalb 8000h, also muß Brücke J3 offenbleiben und die Lötbrücken sollten wie folgt angeordnet werden:

J1/1–J2/A, J1/2–J2/B,
J1/3–J2/C, J1/4–J2/D,
J1/5–J2/E, J1/6–J2/F,
J1/9–J2/G, J1/11–J2/H,
J1/12–J2/I, J1/13–J2/K

Nach einer derartigen Aufrüstung gibt es kaum noch eine unbestückte Adresse. Was nach dem Einschalten des Rechners passiert, ist nun sehr von der BIOS-Version abhängig. Die Besitzer eines V20-BIOS haben es da relativ einfach, da dieses BIOS selbsttätig auch oberhalb der 640-K-Grenze nach RAM sucht und die Größe des Speichers bis zum ersten Bildschirm-Adapter an das DOS meldet [2].

Integrierter Speicher

Wenn Sie Ihren Rechner nicht mit dem V20-BIOS bestücken können oder wollen, bleibt Ihnen das in [3] vorgestellte Programm, mit dem Sie dem Rechner die Speichergröße selbst mitteilen. Dabei handeln Sie sich zwar eine kleine Verzögerung beim Systemstart ein, weil der Rechner ein zweites Mal booten muß, trotzdem ist es wahrscheinlich der einfachste und unkomplizierteste Weg, dem DOS das Wissen über den erweiterten Speicher beizubringen.

Ein etwas umständlicheres Verfahren setzt voraus, daß Sie ein EPROM-Programmiergerät zur Verfügung haben und die Assemblersprache beherrschen. Sie müssen nämlich die Stelle im BIOS finden, an der während des Speichertests mit einem Compare-Befehl getestet wird, ob bereits das Segment A000h erreicht wurde. Es handelt sich meistens um Befehle wie CMP BX,A000 oder CMP BH,A0, die gegen CMP BX,B000 oder CMP BH,B0 ausgetauscht werden müssen, wenn der Speichertest zum Beispiel bei B000h enden soll. Das Problem ist nur, den richtigen Befehl zu erwischen, denn diese Byte-Kombinationen können ja mehrmals im BIOS auftauchen.

Jenseits des Adapters

Der Bereich über den Bildschirm-Adaptern läßt sich nicht ohne weiteres an das DOS angliedern, da es ohne besondere Hilfestellungen den Speicher nur am Stück verwalten kann. Wir arbeiten jedoch an einer Lösung, die auch diesen Speicher direkt für das Betriebssystem verfügbar macht. 960 KByte DOS-Workspace sind zwar noch Zukunftsmusik, aber per Software machbar. Dennoch gibt es keinen Grund, die Segmente C000h, D000h und E000h vorerst einfach brachliegen zu lassen: man kann sie vorerst für eine RAM-Disk einsetzen, die gegenüber VDISK den Vorteil hat, daß sie resetfest ist. Bei einem Warmstart merkt das Treiberprogramm, ob die RAM-Disk schon initialisiert wurde, und verändert sie in diesem Fall nicht.

Bei der RAM-Disk handelt es sich um ein Programm, das sich auch auf der V20-BIOS-Diskette befindet, was vielen das Abtippen ersparen wird. Sie ist als Device-Treiber geschrieben, muß also ins CONFIG.SYS aufgenommen werden. Bevor Sie die Datei mit MASM, LINK und EXE2BIN wie ein COM-File übersetzen, müssen Sie noch einige Software-Schalter setzen, die am Kopf des Listings zu finden sind.

Zur Verwendung in PCs und XTs dürfen Sie die Konstante V20 nur dann auf '1' setzen, wenn Ihr Rechner mit einem V20-Prozessor ausgestattet ist, anderenfalls tragen Sie hier eine Null ein. Die Angabe über das erste mit RAM bestückte Segment befindet sich in RAMD_SEG und sollte gegebenenfalls geändert werden. Bei dem für SYSMAP herangezogenen AT bliebe hier beispielsweise D000h stehen, wäre der Rechner allerdings mit einem AT-Festplatten-Controller ausgerüstet, könnte man das Segment gegen C000h austauschen. SIZE gibt den Default-Wert für die Größe der RAM-Disk an, der aber beim Aufruf durch einen Parameter überschrieben werden kann.

Übrigens sollten Sie die Datei auf der Diskette nicht als EXE- oder COM-File ablegen, da es sich nicht um ein direkt ausführbares Programm, sondern um einen Treiber handelt. Für Treiber hat sich die Endung SYS eingebürgert. Die RAM-Disk aktivieren Sie, indem Sie CONFIG.SYS um die zu Beginn der Datei angegebene Zeile erweitern. Für den AT mit Festplatte sähe ein Aufruf etwa folgendermaßen aus:

DEVICE=
RAMD.SYS/D48/S96

Die RAM-Disk belegt dabei 96 KByte im Bereich von D000h bis E7FFh und das Directory ist auf 48 Einträge begrenzt, was in den meisten Fällen ausreicht.

(mw)

```
;SET UP: Put line into CONFIG.SYS with contents

;DEVICE = Programname.Ext /Dxxx/Sxxx/1/2/8/9

;where  /1 = single sided        ,not really needed
;       /2 = double sided                "
;       /8 = 8 sectors                   "
;       /9 = 9 sectors                   "
;       /D = Directory entries   ,set to 48 or 64
;       /S = Size of RAMDISK     ,set to 64 or 128
;
.186
V20     EQU     1       ;SET TO 0 FOR 8088
;
RAM     PROC    FAR
CSEG    SEGMENT 'CODE'
ASSUME  CS:CSEG,DS:CSEG,ES:CSEG
;
RAMD_SEG        EQU     0C000H  ;FIRST SEGMENT OF RAMDISK
;RAMD_SEG        EQU     0D000H  ;FIRST SEGMENT OF RAMDISK

SIZE            EQU     128     ;SIZE IN KB

SYS_CTRL_PORT   EQU     061H
SECTLEN         EQU     200H

;DEVICE HEADER
        DB      4 DUP (0FFH)
        DW      2000H   ;BLOCK DEVICE,no IOCTL,IBM FORMAT
        DW      DEV_STRATEGY
        DW      DEV_INT
        DB      1       ;NUMBER OF UNITS
;
REQ_HEADER_BX   DW      0       ;REQUEST HEADER ADDRESS
REQ_HEADER_ES   DW      0
;

COMMAND_CODE_ADDRESS:
RAMD_STRT DW    INIT
        DW      MEDIA_CHECK - OFFSET BOOT_SECTOR
        DW      BUILD_BPB - OFFSET BOOT_SECTOR
        DW      IOCTL_INP - OFFSET BOOT_SECTOR
        DW      INPUT - OFFSET BOOT_SECTOR
        DW      NON_DESTR_INPUT - OFFSET BOOT_SECTOR
        DW      INPUT_STATUS - OFFSET BOOT_SECTOR
        DW      INPUT_FLUSH - OFFSET BOOT_SECTOR
        DW      OUTPUT - OFFSET BOOT_SECTOR
        DW      OUTPUT_VERIFY - OFFSET BOOT_SECTOR

DEV_STRATEGY:
        MOV     CS:[REQ_HEADER_ES],ES   ;STORE  REQ HEADER
        MOV     CS:[REQ_HEADER_BX],BX   ;SUPPLIED IN ES:BX
        RET
;
DEV_INT:
        CLD     ;MS-DOS ENTERS HERE WITH REQ HEADER
        PUSH    DS      ;SAVE THE REGISTERS
        PUSH    ES
        IF      V20
        PUSHA
        ELSE
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    DI
        PUSH    SI
        PUSH    BP
        ENDIF
        ;the following Statement is not needed now, since
        ;MS-DOS now enters here directly after STRATEGY
        ;but could be needed later if DEV_INT occurs
        ;asynchronously
        LES     BX,DWORD PTR CS:[REQ_HEADER_BX]
        MOV     AL,ES:[BX+2]    ;GET COMMAND CODE
        MOV     SI,RAMD_SEG
NOP_NOP:
        MOV     SI,CS   ;is replaced by 2 NOP's during INIT
NOT_INIT:
        CMP     AL,0AH  ;TEST IF ILLEGAL COMMAND
        JB      NOT_ILLEGAL
        MOV     AL,3    ;GET ILLEGAL CODE
NOT_ILLEGAL:
        SHL     AL,1    ;DOUBLE COMMAND-VALUE
        PUSH    CS
        PUSH    SI
        MOV     SI,OFFSET COMMAND_CODE_ADDRESS
        XOR     AH,AH   ;AX NOW HAS COMMAND-VALUE * 2
        ADD     SI,AX   ;ADD TO COMMAND_TABLE ADDRESS
        PUSH    CS
        POP     DS
        PUSH    [SI]    ;EXECUTE COMMAND
        RET             ;RETURNS TO COMMAND

;THIS CODE, INCLUDING THE I/O-COMMANDS, IS MOVED TO THE
;BOOT-SECTOR OF THE RAM-DISK. IT WILL BE EXECUTED THERE,
;THE MEMORY OF THE DRIVER-I/O SECTION IS GIVEN BACK TO MS-DOS
```

```
BOOT_SECTOR             DB      0E9H ;BPB-BLOCK,contains most
                        DW      0    ;values for 160 K drive
OEM_NAME                DB      "Köhlmann" ;by default
BYTES_PER_SECTOR        DW      512        ;will be replaced
SECTORS_PER_ALLOC_UNIT  DB      1          ;during init by
RESERVED_SECTORS        DW      1          ;values for options
NUMBER_OF_FATS          DB      1
DIR_ENTRIES             DW      64
SECTORS                 DW      256
MEDIA_DESCRIPTOR        DB      0FEH
FAT_SECTORS             DW      1
SECTORS_PER_TRACK       DW      8
NUMBER_OF_HEADS         DW      1
HIDDEN_SECTORS          DW      0

BPB_POINTER DW OFFSET BYTES_PER_SECTOR - OFFSET BOOT_SECTOR
;
COMPUTE_SECTOR  PROC    NEAR
        LDS     SI,DWORD PTR ES:[BX+0EH] ;GET TRANSFER ADDR
        MOV     AX,WORD PTR ES:[BX+14H] ;GET STARTING SECTOR
        MOV     CX,32
        MUL     CX
        MOV     DX,RAMD_SEG
        ADD     DX,AX
        PUSH    DX    ;DX CONTAINS SEGMENT ADDRESS OF SECTOR
        MOV     AX,ES:[BX+12H]   ;GET NUMBER OF SECTORS
        MOV     CX,512
        XOR     AH,AH
        MUL     CX
        JNB     SEGM_OFL         ;TEST IF SEGMENT OVERFLOWS
        MOV     AX,0FFFFH
SEGM_OFL:
        MOV     CX,AX            ;CX HAS # OF BYTES
        ADC     AX,SI            ;TEST IF TRANSFER-SEG OVERRUN
        JNB     A0090            ;IF NOT,OK
        MOV     AX,0FFFFH
        SUB     AX,SI
        ADD     AX,1
        XCHG    CX,AX  ;CX NOW HAS # OF SECTORS UNTIL SEG-END
        JNB     A0090
        MOV     CX,0FFFFH
A0090:  POP     DX
        RET
COMPUTE_SECTOR  ENDP
;
ILLEGAL:         ;all these function calls are not supported
IOCTL_INP:       ;by the RAM-driver, they are not needed
NON_DESTR_INPUT:
INPUT_STATUS:
INPUT_FLUSH:
OUTPUT_STATUS:
OUTPUT_FLUSH:
IOCTL_OUTP:
        MOV     AX,8103H         ;GET "UNKWOWN COMMAND"
        JMP     SHORT   SET_RET_CODE
;
NORMAL_OUT:     MOV     AX,0100H         ;GET "no ERROR"
;
SET_RET_CODE:
        POP     DS       ;GET SEGMENT OF REQUEST-HEADER
        LES     BX,DWORD PTR DS:[REQ_HEADER_BX]
        MOV     ES:[BX+3],AX     ;PUT RETURN-CODE IN
                                 ;RETRIEVE THE REGISTERS
        IF      V20
        POPA
        ELSE
        POP     BP
        POP     SI
        POP     DI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        ENDIF
        POP     ES
        POP     DS
        RET
;
;
OUTPUT_VERIFY   PROC NEAR
;
OUTPUT  PROC NEAR
        CALL    COMPUTE_SECTOR
        PUSHF
        MOV     ES,DX
        XOR     DI,DI
        JMP     SHORT MOVE

OUTPUT  ENDP
OUTPUT_VERIFY ENDP
;
INPUT   PROC NEAR
        CALL    COMPUTE_SECTOR
        PUSHF
        PUSH    DS
        POP     ES
        XCHG    DI,SI
        MOV     DS,DX
        XOR     SI,SI
MOVE:   REPZ    MOVSB
        POPF
        JNB     NORMAL_OUT
        MOVSB
        JMP     SHORT NORMAL_OUT
INPUT   ENDP
;
MEDIA_CHECK     PROC NEAR
        MOV     BYTE PTR ES:[BX+0EH],1  ;SET "MEDIA HAS NOT
        JMP     NORMAL_OUT      ;BEEN CHANGED"
MEDIA_CHECK     ENDP
;
BUILD_BPB       PROC NEAR
        MOV     AX,OFFSET BYTES_PER_SECTOR - OFFSET BOOT_SECTOR
        MOV     ES:[BX+12H],AX  ;POINT TO BPB
        MOV     ES:[BX+14H],RAMD_SEG
        JMP     NORMAL_OUT
BUILD_BPB       ENDP
;
INIT    PROC    NEAR
;INIT MEMORY FROM D000 TO E000 SEGMENT, NOT REALLY NEEDED WITH
;V20-BIOS BECAUSE THAT BIOS ALREADY HAS INITIALIZED THIS MEMORY
;JUST A SAFETY PRECAUTION FOR COMPUTERS WITH A SNAIL-BIOS

        MOV     WORD PTR NOP_NOP,9090H  ;set 2 NOP's first
        PUSH    ES
        PUSH    DI
        MOV     AL,0
        OUT     [0A0H],AL
        IN      AL,SYS_CTRL_PORT
        XOR     AL,30H
        OUT     SYS_CTRL_PORT,AL
        mov     ax,0c000h        ;if you do not have
        call    init_ram         ;a Harddisk
        MOV     AX,0D000H
        CALL    INIT_RAM
        MOV     AH,0E0H
        CALL    INIT_RAM
        IN      AL,SYS_CTRL_PORT
        XOR     AL,30H
        OUT     SYS_CTRL_PORT,AL
        MOV     AL,80H
        OUT     [0A0H],AL
        POP     DI
        POP     ES
;memory now good,no Parity will occur on reads

        LDS     SI,DWORD PTR ES:[BX+12H] ;DS:SI POINT TO
        CLD     ;       PARAMETER AFTER '=' FROM CONFIG.SYS
FIND_SLASH:
        LODSB
        CMP     AL,0DH
        JZ      CR_FOUND
        CMP     AL,"/"
        JNZ     FIND_SLASH      ;LOOP UNTIL SLASH OR CR
GET_OPTION:
        LODSB           ;GET NEXT CHARACTER
        CMP     AL,0DH  ;SEE IF CR
        JZ      CR_FOUND
        OR      AL,20H  ;NOW MAKE lowercase
        CMP     AL,"d"  ;LEGAL OPTIONS ARE :
        JZ      D_FOUND ;/D/S/1/2/8/9
        CMP     AL,"1"
        JNZ     NOT_1
        OR      BYTE PTR CS:[OPTION_FLAG],41H
NOT_1:  CMP     AL,"8"
        JNZ     NOT_8
        OR      BYTE PTR CS:[OPTION_FLAG],42H
NOT_8:  CMP     AL,"9"
        JNZ     NOT_9
        OR      BYTE PTR CS:[OPTION_FLAG],40H
NOT_9:  CMP     AL,"2"
        JNZ     NOT_2
        OR      BYTE PTR CS:[OPTION_FLAG],40H
NOT_2:  CMP     AL,"s"
        JNZ     GET_OPTION
        CALL    COMPUTE_VALUE
        MOV     CS:[DISK_SIZE],AX
        JMP     FIND_SLASH

DISK_SIZE       DW      SIZE    ;SIZE IN K-BYTES
RAMD_END        DW      0       ;POINTS TO END OF RAMDISK
;
INIT_RAM:
        MOV     DS,AX
        MOV     ES,AX
        XOR     DI,DI
        MOV     SI,DI
        MOV     CX,8000H
        REP     MOVSW
        RET
;
D_FOUND:
        CALL    COMPUTE_VALUE
        ADD     AX,000FH        ;ROUND TO NEXT DIR ENTRY #
        AND     AX,02F0H        ;TEST FOR MAX
        CMP     AX,0            ;SEE IF TOO MUCH OR NONE
        JNZ     DIR_ENTRY
        MOV     AX,0010H        ;AT LEAST 16 ENTRIES
DIR_ENTRY:
        MOV     WORD PTR CS:[DIR_ENTRIES],AX
        OR      BYTE PTR CS:[OPTION_FLAG],20H ;SIGNAL MANUAL
        JMP     FIND_SLASH
;
CR_FOUND:
        PUSH    CS
        POP     DS
        MOV     DL,[OPTION_FLAG]
        TEST    DL,40H  ;SEE IF 1,2,8 OR 9 SPECIFIED
        JZ      GOT_SIZE        ;SEE IF ENOUGH MEMORY
```

```
        MOV     AX,SIZE
DSK_SIZE:
        MOV     [DISK_SIZE],AX
;
GOT_SIZE:
        MOV     AX,[DISK_SIZE]
        ;if bigger than 180 K ,we need 2 FAT-sectors
        CMP     AX,180
        JB      ONE_FAT_SECT
        INC     WORD PTR FAT_SECTORS
ONE_FAT_SECT:
        SHL     AX,1    ;2 TIMES DISK-SIZE = # OF SECTORS
        MOV     DX,[FAT_SECTORS]
        SHL     DX,1
        MOV     SI,[DIR_ENTRIES]
        MOV     CL,4
        SHR     SI,CL
        ADD     DX,SI
        ADD     DX,+6
        AND     DX,00FEH
        CMP     AX,DX
        JNB     TEST_SIZE
        XCHG    DX,AX
TEST_SIZE:
        MOV     [DISK_SIZE],AX
        SHR     WORD PTR [DISK_SIZE],1
        MOV     [SECTORS],AX
        MOV     CL,5
        SHL     AX,CL
        MOV     [RAMD_END],AX
PREP_DISK:
        MOV     DX,RAMD_SEG     ;BEGINN IN HIGH MEM
        MOV     [RAMD_STRT],DX
        CALL    DISPLAY_CAPACITY
        MOV     ES,[RAMD_STRT]
        XOR     DI,DI
        MOV     SI,OFFSET BOOT_SECTOR
        MOV     CX,100H
        REPZ    CMPSB
        JCXZ    ENDED
        XOR     DI,DI
        MOV     SI,OFFSET BOOT_SECTOR
        MOV     CX,512H
        REPZ    MOVSB
        MOV     BX,1    ;PREPARE FAT
        MOV     AX,[FAT_SECTORS]
        CALL    COMPUTE_SEG
        XOR     AL,AL
        REPZ    STOSB
        XOR     DI,DI
        MOV     AL,[MEDIA_DESCRIPTOR]
        MOV     ES:[DI],AL
        MOV     WORD PTR ES:[DI+1],0FFFFH
        MOV     AL,[NUMBER_OF_FATS]
        CMP     AL,2
        JNZ     PREP_DIR
        PUSH    ES
        PUSH    DI
        CALL    COMPUTE_SEG
        POP     SI
        POP     DS
        REPZ    MOVSB
PREP_DIR:
        MOV     CX,10H  ;PREPARE DIRECTORY,INIT TO ALL ZEROS
        XOR     DX,DX
        MOV     AX,CS:[DIR_ENTRIES]
        DIV     CX
        CMP     DX,0
        JZ      ROUND_DIR
        INC     AX
ROUND_DIR:
        CALL    COMPUTE_SEG
        XOR     AL,AL
        REPZ    STOSB
        XOR     DI,DI   ;now put a volume-label in
        MOV     CX,20H
        MOV     SI,OFFSET LABEL
        PUSH    CS
        POP     DS
        REP     MOVSB
        JMP     SHORT OUT
ENDED:
        MOV     SI,OFFSET RETAIN
        CALL    GET_IT
OUT:    LES     BX,DWORD PTR CS:[REQ_HEADER_BX]
        MOV     WORD PTR ES:[BX+0EH],OFFSET BOOT_SECTOR
        MOV     ES:[BX+10H],CS  ;SET BREAK ADDRESS
        MOV     BYTE PTR ES:[BX+0DH],1  ;NUMBER OF UNITS
        MOV     AX,OFFSET BPB_POINTER - OFFSET BOOT_SECTOR
        MOV     ES:[BX+12H],AX  ;POINT TO BPB-ARRAY
        MOV     ES:[BX+14H],RAMD_SEG
        JMP     NORMAL_OUT      ;DONE INIT,RETURN TO DOS

COMPUTE_VALUE   PROC    NEAR
        PUSH    BX
        XOR     AX,AX
        MOV     BX,AX   ;BOTH ARE NOW 0
        MOV     CX,10
NUMERIC:
        MOV     BL,[SI]
        CMP     BL,0DH
        JZ      SLASH
        CMP     BL,"/"
        JZ      SLASH
        INC     SI
        CMP     BL,"0"  ;NUMBERS ONLY,PLEASE
        JB      NUMERIC
        CMP     BL,"9"
        JA      NUMERIC
        AND     BL,0FH  ;MAKE BCD-DIGIT
        MUL     CX
        ADD     AX,BX
        JMP     NUMERIC ;DO UNTIL SLASH OR CR
;
SLASH:  POP     BX
        RET
COMPUTE_VALUE   ENDP
;
COMPUTE_SEG     PROC    NEAR
        PUSH    AX
        MOV     AX,BX
        MOV     CX,20H
        MUL     CX
        ADD     AX,CS:[RAMD_STRT]
        MOV     ES,AX   ;ES NOW CONTAINS SEGMENT TO INIT
        XOR     DI,DI   ;DI POINTS TO FIRST BYTE
        POP     AX
        ADD     BX,AX
        MOV     CX,SECTLEN
        MUL     CX
        XCHG    CX,AX   ;CX NOW HAS NUMBER OF BYTES TO WRITE
        RET
COMPUTE_SEG     ENDP
;
DISPLAY_CAPACITY        PROC NEAR
        MOV     AX,[DISK_SIZE]
        MOV     SI,OFFSET PLACE
        MOV     CX,0A64H
DIVIDE: DIV     CL
        CMP     AL,0
        JZ      IS_0
        OR      [SI],AL
        XOR     AL,AL
IS_0:   XCHG    AH,AL
        CMP     AX,0
        JZ      GET_DRIVE
        MUL     CH
        INC     SI
        JMP     DIVIDE
;
GET_DRIVE:
        MOV     AH,19H  ;GET CURRENT DRIVE
        INT     21H     ;AL HAS CURRENT DRIVE
        MOV     DL,AL
        MOV     AH,0EH  ;SELECT DRIVE
        INT     21H     ;AL NOW HAS # OF LOGICAL DRIVES
        ADD     AL,41H  ;ADD TO MAKE ASCII
        MOV     [DRIVE],AL      ;WANT TO DISPLAY NEW DRIVE
        MOV     SI,OFFSET TEXT
GET_IT: LODSB
        CMP     AL,"$"
        JZ      DOLLAR  ;DISPLAY UNTIL '$' FOUND
        MOV     AH,0EH
        INT     10H
        JMP     GET_IT
DOLLAR: RET
;
DISPLAY_CAPACITY        ENDP
;
TEXT    DB      0DH,0AH,"RAMDISK-Drive installed as drive "
DRIVE   DB      "X: with "
PLACE   DB      "000 KB capacity",0dh,0ah,"$"
RETAIN  DB      'Contents of Ramdisk found,left intact'
        DB      0dh,0ah,'$'
LABEL   DB      'RAMDISK    ',28H,10 DUP (0),00H,00H
        DB      8FH,10H,18 DUP (0)
OPTION_FLAG     DB      0
;
INIT    ENDP
RAM     ENDP
CSEG    ENDS
        END
```

Dieser RAM-Disk-Treiber kann auch Speicher oberhalb der Bildschirm-Adapter verwalten. Bei einem Warmstart tastet er bestehende Dateien nicht an.

Literatur

[1] Rudolf Bremer, Mehr als 640 K in PCs, c't 11/86, Seite 94

[2] Peter Köhlmann, V-Chip-Power, c't 10/87, Seite 208

[3] Andreas Landenberger, Booten mit List, c't 11/87, Seite 154