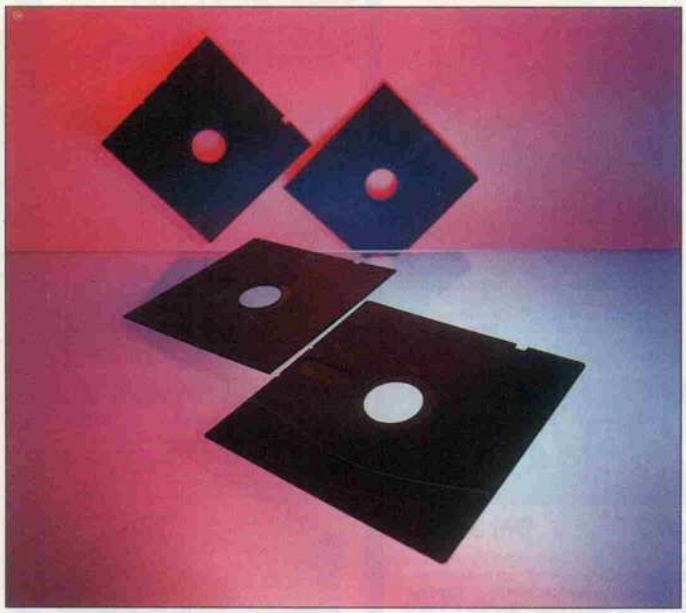


Jenseits der Adapter

PC-/MSDOS nutzt RAM weit oberhalb der 640-K-Grenze

Ralf Preller

Bisher endete jede DOS-Speichererweiterung über 640 KByte hinaus an den Bildschirmadaptern, womit spätestens bei 736 KByte die letzte Grenze erreicht war. Die Videokarte stellt aber keine unüberwindliche Hürde dar, wenn man dem DOS mitteilt, daß es diesen Bereich nicht antasten soll. Jetzt können Sie Ihre residenten Hilfsprogramme einfach in das RAM hinter die Bildschirmadapter schicken.

Wer einmal auf den 'Geschmack' gekommen ist, möchte sie nicht mehr missen: die vielen mehr oder weniger kleinen residenten Helferlein, die neue (natürlich wieder einige 10 KByte längere) DOS-Version, den Drucker-Spooler und und . . . Irgendwann steht die unerbittliche Meldung 'Nicht genügend Speicher' auf dem Monitor. Dann hilft nur Abspecken oder neuen Speicher anschaffen.

In Heft 5/88 haben wir eine Speichererweiterung veröffentlicht, die alle unbenutzten Adreßbereiche im PC oder AT mit RAM versieht. Damals konnten wir jedoch noch keine Lösung anbieten, wie man den Speicherbereich oberhalb des Video-Adapters unter DOS nutzen kann. Hier ist sie nun: bis zu 928 KByte unter DOS. Die neu vorgestellte Software eignet sich jedoch nicht nur für die CMOS-RAM-Karte, sondern auch für jede andere Erweiterung, sei sie kommerzieller Natur oder 'nur' eine Bastellösung.

Wie man unter DOS neu gewonnenes RAM nutzt, wenn man keine EGA-Karte besitzt, und wie man mit Hilfe der c't-Speicherkarte den durchgehenden Hauptspeicher auf 704 oder gar 736 KByte erweitert, haben wir in [3] beschrieben. Darüber hinaus aber läuft nichts mehr, denn spätestens dann versperrt der von IBM auf so unglücklich niedrige Adressen (Segment B000h beziehungsweise B800h) gelegte Video-Adapter den Weg zu den meist unbenutzten Segmenten D000h und E000h. DOS kann den Speicher nämlich leider nur am Stück verwalten.

Soll das DOS mehr Hauptspeicher zur Verfügung haben, kommt man nicht umhin, RAM jenseits des Video-Adapters heranzuziehen. PCDOS 4.0 beschreitet diesen Weg durch Unterstützung des 'LIM-(Lotus-Intel-Microsoft-)'Expanded-Memory-Adapters'. Aber auch dies hilft wenig, wenn die verwendete Software damit nicht umgehen kann – und das ist leider in der Mehrzahl so.

Mit der c't-Speichererweiterung steht eine Menge direkt adressierbares RAM oberhalb des Video-Adapters bereit. Diesen Bereich kann man zwar als RAM-Disk nutzen, doch zumindest bei Festplatten-Rechnern ist der Geschwindigkeitsgewinn wenig attraktiv. Wie aber bringt man dem DOS diesen zusätzlichen Speicher näher?

Die 'saubere' Lösung, entsprechenden Code im IO.SYS (MSDOS) oder IBMBIO.COM (PCDOS) unterzubringen, ist zwar möglich, aber wegen des erheblichen Aufwands und der Versionsvielfalt in der DOS-Welt wenig praktikabel. Statt dessen beschränken wir uns hier darauf, das DOS zu überlisten. Dies geht in zwei Stufen vonstatten.

Gaukeleien

Zuerst wird dem ROM-BIOS vorgegaukelt, daß bis zur höchsten verfügbaren RAM-Adresse durchgehend Hauptspeicher zur Verfügung stehe; das können (mit einem Monochrom-Adapter) bis zu 1008 KByte sein. Beim Booten läßt das DOS dabei glücklicherweise den Video- und Controller-BIOS-Bereich unangetastet. Danach werden die Speicher-Kontrollblöcke des DOS derart verbogen, daß das DOS diesen Bereich als residenten Treiber ansieht, und schon ist die Speicheranbindung perfekt.

Aber der Reihe nach. Das zentrale Hilfsprogramm für die Anbindung der Speichererweiterung heißt CONFRAM und ist in Turbo-Pascal 4.0 geschrieben. Turbo 4.0 eignet sich in diesem Fall etwas mehr als sein Vorgänger, weil es EXE-Programme produziert, denen nur die tatsächlich benötigte Menge Hauptspeicher zur Verfügung gestellt wird. Bei Turbo 3.0 wäre mehr Aufwand erforderlich.

CONFRAM kann ohne, mit einem oder mit drei Parametern aufgerufen werden, je nachdem welche Aufgabe gerade zu erledigen ist:

Ohne Parameter

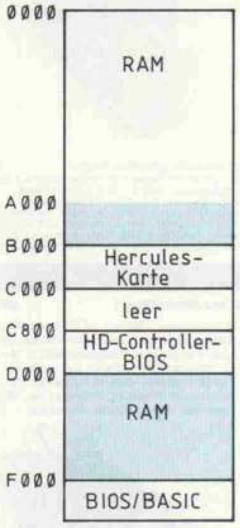
CONFRAM zeigt die momentan gesetzte Speichergröße in Kilobyte an. Diese wird vom Rechner beim (Kaltstart-) Speichertest ermittelt und an der Wort-Adresse 0:413h (40h:13h) im BIOS-RAM abgelegt. Ohne Parameter darf man CONFRAM jederzeit aufrufen, mit Parametern dagegen nur während der Bootprozedur.

Mit einem Parameter

Zum Beispiel:

```
CONFRAM 0B000h
```

In dieser Betriebsart läßt sich das DOS-verfügbare RAM bis zum Anfang des Video-RAMs erweitern, ähnlich wie mit dem Assemblerprogramm in [2]. Der Parameter bezeichnet die erste Segmentadresse, die dem DOS nicht mehr zur Verfügung steht. CONFRAM überprüft, ob die Segmentadresse der im BIOS-RAM abgelegten Speichergröße entspricht; wenn nicht, dann wird die Speichergröße an der Adresse 0:413h dem Parameter angepaßt, das 'Reset-Flag' an



Der Beispielrechner: An die Standardbestückung schließen sich direkt 64 KB Speichererweiterung an. Weitere 128 KB stehen hinter dem Festplatten-Controller-BIOS zur Verfügung.

der Adresse 0:472h auf den Warmstart-Wert 1234h vorbesetzt und der Rechner neu gebootet. Danach verwaltet das DOS einen entsprechend größeren Speicherbereich; mit dem Parameter B000h sind dies 704 KByte.

Natürlich sollte dort wirklich RAM vorhanden sein, sonst stürzt der Rechner ab. Wenn die benutzte Speichererweiterung im Gegensatz zur c't-Karte eine Paritätslogik aufweist, muß CONFRAM nach dem zweiten Booten erneut mit demselben Parameter aufgerufen werden, um den Speicher zwischen der 640-K-Grenze und der Endadresse zu löschen. Es empfiehlt sich, die Aufrufzeile als ersten Eintrag in die Datei AUTOEXEC.BAT aufzunehmen.

Mit drei Parametern

zum Beispiel:

```
CONFRAM F000 A000-D000
```

In diesem Fall erfolgt gegebenenfalls zunächst wieder ein Setzen der Speichergröße im BIOS-RAM auf einen dem ersten Parameter entsprechenden Wert (hier 960 KByte) und ein Neu-Booten des Systems. Im zweiten Durchlauf aber wird nicht nur das RAM zwischen 640 KByte und (hier) 960 KByte gelöscht (das kann bis zu zwei Sekunden dauern – also bitte etwas Geduld), sondern anhand der letzten beiden Parameter werden die Speicher-Kontrollblöcke so verändert, daß DOS den angegebenen Bereich ausläßt. Der zweite Parameter gibt dabei das Anfangssegment des auszublendenden Bereichs an, der letzte Parameter das erste wieder für MSDOS verfügbare Segment. Zu beachten ist, daß keine Irrtümer bei den Parametern passieren dürfen, denn es findet keine Überprüfung statt, ob wirklich RAM (hier im Bereich der Segmente D000h bis EFFFh) vorhanden ist.

Speicherausbau	Segment
639 KB	9FC0h
640 KB	A000h
703 KB	AFC0h
704 KB	B000h
736 KB	B800h
768 KB	C000h
800 KB	C800h
832 KB	D000h
864 KB	D800h
896 KB	E000h
928 KB	E800h
959 KB	EFC0h
960 KB	F000h

Einige typische Parameter für CONFRAM

An die Kette gelegt

Um das Ausblenden des Videospeichers zu verstehen, ist ein Ausflug in die MSDOS-Speicherverwaltung nötig. DOS verwaltet das RAM über eine Kette von sogenannten 'memory control blocks' (MCB, Speicherzuordnungsblock). Jedem Speicherbereich, den DOS an ein Anwenderprogramm (oder zum Beispiel COMMAND.COM) vergibt, lagert es einen 16 Byte langen Block vor, der eine feste Struktur aufweist.

Offset	Bedeutung
0	'M', wenn weitere MCBs folgen 'Z', wenn letzter MCB
1...2	Segment-Zeiger auf den "Program Segment Prefix" (PSP), den Kopf desjenigen Programms, zu dem dieser Speicherblock gehört
3...4	Länge des Speicherblocks in Paragraphen (zu 16 Byte)
5...15	unbenutzt

Jeder MCB belegt genau 16 Bytes.

Den nächsten MCB findet man durch Addition des Segments, an dem sich der Ausgangs-MCB befindet, mit der Länge aus Byte 3 und 4 und einer weiteren Addition mit 1. Unbenutzte Speicherblöcke erkennt DOS daran, daß Byte 1 und 2 auf Null gesetzt sind. Ausführliche Informationen zur DOS-Speichervergabe finden Sie in [4].

Das Verfahren ist in allen DOS-Versionen seit 2.0 bis (mindestens) PCDOS 4.0 gleich; einzig der Startpunkt der MCB-Kette läßt sich seit DOS 3.10 einfacher ermitteln. Seitdem ist der System-Konfigurationstabelle bei Offset –4 ein Zeiger (Offset, Segment) auf den MCB-Kettenanfang zu entnehmen. Den Zeiger auf die Tabelle liefern ES:BX nach dem Aufruf von INT 21h, Funktion 52h.

Als Beispiel haben wir ein AUTOEXEC.BAT und die daraus entstandene MCB-Kette abgedruckt. Man sieht, daß jedem Programm mit seinem MCB noch eine eigene 'Programmierumgebung' (Environment) mit MCB vorgelagert ist. Sie können sich bei stark wechselnder Speichernutzung aber auch mal an anderer Stelle befinden.

Die gesamte Speicherverwaltung erfolgt über MCBs und PSPs. Das Programm-Segment-Präfix (PSP) befindet sich direkt vor jedem DOS-Programm. MSDOS merkt sich sonst nur noch das gerade aktive Programm (den Anfang des aktiven PSP). Glücklicherweise benutzt MSDOS nach dem Residentmachen eines Programms nie wieder dessen PSP-Inhalt, somit steht einer Manipulation der Speicherverwaltung nichts im Wege. Aber auch wenn DOS das PSP nicht mehr braucht, sollten Anwenderprogramme ihn möglichst nicht antasten, da sich dort Daten befinden, auf die Systemprogramme gern zurückgreifen. Insbesondere befindet sich im PSP ein Zeiger auf das zugehörige Environment, aus dem Name und Ladepfad des residenten Programms ermittelt werden können. Davon macht auch das DOS-Analyseprogramm MEMMAP [4] Gebrauch.

CONFRAM verändert zuerst seinen eigenen MCB derart, daß der Speicherblock nicht mehr als letzter Block markiert ist und daß er bis zum Anfang des auszublendenden Adreßbereichs (Bildschirmspeicher) reicht. Direkt vor der Lücke wird ein neuer MCB angelegt, dessen Länge gerade bis zum Anfang des RAM-Blocks oberhalb der Adapter reicht und dessen

Segm	*PSP	Länge	Typ	Programmname
0000				Interrupt-Vektoren
0040				BIOS-Datenbereich
0070				DOS 3.30
09AC	0008	2720 Bytes		DOS-Puffer, Treiber
0A57	0A57	3696 Bytes		COMMAND.COM
0B3F	0000	48 Bytes		freier Speicher
0B43	0A57	160 Bytes	Env	Standard-Environment
0B4E	0B52	48 Bytes	Env	SK.COM
0B52	0B52	109296 Bytes	Prog	SK.COM
2602	2608	80 Bytes	Env	UTI.EXE
2608	2608	96480 Bytes	Prog	UTI.EXE
3D97	0000	467600 Bytes		freier Speicher

Die Kette der Speicherkontrollblöcke beginnt hier an der Adresse 09ABh; Segm bezeichnet das erste Segment des vom MCB reservierten Speichers. Auf das residente SideKick folgt das Analyseprogramm UTI, das diese Liste erzeugt hat. Zu jedem Programm legt DOS ein eigenes Environment an.

```
confram f000 afc0-d000
jenseits sk.com
PROMPT=$t$h$h$h$h$h$h $n$g
```

Um dem DOS einen direkten Zugriff zu ermöglichen, muß CONFRAM im AUTOEXEC.BAT stehen. Das Programm JENSEITS startet Programme im RAM jenseits der Adapter.

Segm	*PSP	Länge	Typ	Programmname
0000				Interrupt-Vektoren
0040				BIOS-Datenbereich
0070				DOS 3.30
09AC	0008	2720 Bytes		DOS-Puffer, Treiber
0A57	0A57	3696 Bytes		COMMAND.COM
0B3F	0000	48 Bytes		freier Speicher
0B43	0A57	160 Bytes	Env	Standard-Environment
0B4E	0B54	80 Bytes	Env	UTI.EXE
0B54	0B54	96480 Bytes	Prog	UTI.EXE
22E3	0000	576960 Bytes		freier Speicher
AFC0	AFC0	132096 Bytes	Prog	unbekannt (PSP zerstört)
D001	D005	48 Bytes	Env	sk.com
D005	D005	109296 Bytes	Prog	sk.com
EAB5	0000	21680 Bytes		freier Speicher

Ein Memory-Control-Block blendet den Bereich zwischen der Hercules-Karte und dem Ende des HD-Controller-BIOS für DOS aus. SideKick liegt nun hinter den Adaptern, damit ist der größte zusammenhängende freie Speicherblock bei gleicher Systembelegung um über 100 KByte größer geworden. Anwenderprogramme können über 640 KByte am Stück nutzen.

PSP-Zeiger in den Adapterbereich hineinzeigt. Dies ist sehr wichtig, da MSDOS die Zugehörigkeit von Speicherblöcken zu einem Programm an den PSP-Zeigern erkennt und bei Verlassen des Programms alle Speicherblöcke mit gleichen PSP-Zeigern automatisch mit freimacht.

Am Anfang der Speichererweiterung jenseits der Adapter bleibt nichts weiter zu tun, als einen 'letzten' und unbenutzten MCB zu erzeugen, dann darf sich CONFRAM verabschieden, und MSDOS wird keine Einwände gegen die verbogene MCB-Kette erheben.

Entwickelt wurde das Verfahren auf einem Tulip-PC unter MSDOS 3.10, getestet ist es aber auch auf IBM XT, Compaq PC, IBM XT 286 und Tulip AT unter PCDOS 3.3 und sogar PCDOS 4.0! Man kann somit annehmen, daß CONFRAM unter allen verbreiteten DOS-Versionen und auf allen wirklich kompatiblen Rechnern läuft – mit Ausnahme alter IBM PCs. Die setzen die obere Speicheradresse selbst bei Ctrl-Alt-Del auf den Kaltstartwert zurück. Dieses Problem bekommt man aber mit dem gegen Ende des Beitrags abgedruckten Bootsektor in den Griff.

Kontrollierte Controller

Bevor Sie CONFRAM zum ersten Mal in Betrieb nehmen, sollten Sie prüfen, ob der Rechner den gesamten Hauptspeicher bis 640 KByte für DOS bereithält. Es gibt nämlich XT-Festplatten-Controller, wie einige Modelle aus der OMTI-Serie, die am oberen Speicherende 1 KByte für den Eigenbedarf abzwacken und damit erhebliche Probleme verursachen. Sicherste Methode, um dies herauszufinden, ist ein Kaltstart des Rechners mit einer Bootdiskette, auf der sich weder CONFIG.SYS noch AUTOEXEC.BAT befinden; CONFRAM und COMMAND.COM sind jedoch erforderlich.

Wenn Sie nach dem Kaltstart CONFRAM ohne Parameter aufrufen, erhalten Sie die Hauptspeichergröße in Kilobyte. Ist diese gleich der erwarteten Zahl (zum Beispiel 640 KByte), dann brauchen Sie die nächsten Ausführungen nicht weiter zu beachten.

Ist das Ergebnis bei einem Soll von 640 KByte aber nur 639 KByte, dann überprüfen Sie als nächstes, was das Controller-BIOS beim Warmstart tut. Rufen Sie dazu CONFRAM mit der angestrebten Endadresse als Parameter auf (beispielsweise CONFRAM F000); CONFRAM sollte nun einen Warmstart veranlassen. Sobald das DOS-Prompt wieder da ist, starten Sie CONFRAM ohne Parameter und vergleichen die jetzt verfügbare Speichergröße mit dem gewünschten Wert. Die folgenden Zahlen beziehen sich auf den Beispielparameter F000, entsprechend 960 KByte. CONFRAM könnte als Ergebnis liefern:

960 KByte

Der Controller nistet sich beim Kaltstart fest ins 640. Kilobyte ein und kümmert sich nicht weiter um Veränderungen der Speichergröße. Würde er immer am oberen Ende des Speichers sein Kilobyte abzwacken, müßte das Ergebnis eigentlich 959 KByte lauten. Die Anfangsadresse des auszublendenden Bereichs ist im späteren AUTOEXEC um ein Kilobyte vorzuverlegen, um keine Konflikte zwischen Controller-Daten und Speicher aufkommen zu lassen, zum Beispiel so:

```
CONFRAM F000 9FC0-D000
```

Weil CONFRAM nur einen Speicherbereich ausblenden kann, ist das A000-Segment nicht zu gebrauchen, da der Controller den unteren zusammenhängenden RAM-Bereich unterbricht. Es sei denn, Sie erweitern CONFRAM so, daß es mehr als einen Bereich ausblenden kann.

959 KByte

Lautet das Ergebnis ein Kilobyte weniger, als es der gewünschten Speichergrenze entspricht, dann rufen Sie CONFRAM immer so auf, daß am oberen Ende des ersten Speicherbereichs ein Kilobyte für den Controller reserviert bleibt, beispielsweise so:

```
CONFRAM F000 9FC0-D000
```

Der Contoller zweigt nach der Speichererweiterung durch CONFRAM zwar nur an der obersten Speichergrenze (959–960 KB) etwas ab, während des Bootens greift er aber auch kurzzeitig auf den niedrigen Bereich (639–640 KByte) zu. Damit es beim Booten nicht zum Absturz kommt, müssen Sie auch dort RAM opfern und verlieren den Bereich von A0000h und AFFFFh. Wenn Sie zum Beispiel mit dem V20-BIOS arbeiten, das ja auch Speicherbereiche zwischen der 640-KB-Grenze und den Bildschirmadaptern erkennt, könnten Sie das A000h-Segment weiterhin fast vollständig nutzen, wenn Sie folgenden Aufruf wählen (bei RAM zwischen A0000h und AFFFFh):

```
CONFRAM F000 AFC0-D000
```

639 KByte

Der Controller setzt den RAM-Bereich bei jedem Warmstart neu. In diesem Fall gibt es leider nur die Möglichkeit, den benutzten Bootsektor auf der Diskette beziehungsweise Festplatte zu manipulieren. Später ist CONFRAM im AUTOEXEC ebenfalls so aufzurufen, daß der Speicherbereich 9FC00h bis A0000h ausgeblendet wird, zum Beispiel:

```
CONFRAM F000 9FC0-D000
```

Speichersalat

Wenn Sie CONFRAM benutzen, müssen Sie natürlich wissen, wo überall RAM vorhanden ist. Dazu ist in [3] das Programm SYSMAP abgedruckt. Wer darüber nicht verfügt, kann sich natürlich auch auf DEBUG oder sein Fingerspitzengefühl verlassen. Dennoch ist ein Speichertest zur Absicherung anzuraten.

CONFRAM ist der Einfachheit halber so gestaltet, daß nur ein Speicherblock ausgeblendet werden kann. Wer keine EGA-Karte, aber einen XT-Festplatten-Controller mit lokalem Speicherbedarf besitzt, der könnte auch drei auszublendende Speicherbereiche sinnvoll nutzen. Der Source-Code von CONFRAM ist aber so klar geschrieben, daß er leicht erweitert werden kann.

Die vielen verschiedenen Schreibweisen für die Parameter von CONFRAM sind übrigens keine Druckfehler, sondern werden allesamt korrekt erkannt.

SideKick ins Jenseits

Um es ganz deutlich zu sagen: Natürlich lassen sich mit dem hier praktizierten RAM-Erweiterungsverfahren immer noch keine Programme von 800 KByte Länge starten, weil die Programme zusammenhängen müssen, der freie Speicher aber aufgeteilt ist. Richtig interessant wird die Aktion aber

```
                        page    90,132
                        title   BOOTBOTH.ASM - PC/MSDOS 3.xx Boot Code
                ;***************************************************************
                ;                       B O O T B O T H . A S M
                ;
                ;               Boot code for PC / XT / AT computers
                ;   boots IBMBIO.COM and compatible IO.SYS from DOS Version 3.xx
                ;           includes boot code for Tulip IO.SYS 1.1x
                ;                   (c) Ralf Preller 20.4.1987
                ;
                ;     c't-Version 1988, unterstuetzt die c't RAM-Erweiterung
                ;
                ;   Anm. zur c't-Version:
                ;   Folgende Parameter muessen ggf. geaendert werden:
                ;   - Kilobytes         Zu setzende Speicher-Obergrenze
                ;   - BPB               bei anderen Diskettenformaten als 360K
                ;   - DisplayMessage:   wenn ein vorhandenes ROM-BASIC bei Boot-
                ;                       fehlern angesprungen werden soll:
                ;                       JMP $ durch INT 18H ersetzen
                ;
                ;   Wenn dieser Bootsektor zur allgemeinen Anwendung dienen soll
                ;   (ohne Speichererweiterung), dann brauchen nur die beiden
                ;   Anweisungen hinter "Set new value for memory size" entfernt
                ;   zu werden.
                ;***************************************************************

= 03C0          Kilobytes       equ     960             ;!!! muss angepasst werden
= 7C00          BootOfs         equ     7C00h
```

```
0000                            Abso            segment at 0
0078                                             org     1Eh * 4
0078                            Disk_Pointer     label   word
0413                                             org     413h
0413                            Memory_Size      label   word
0413                            Abso            ends


0000                            IosysSeg        segment at 70h
0000                                             org     0
0000                            StartIosys       label   far
0000                            IosysSeg        ends


0000                            Code    segment
                                assume  cs:Code
0000                                             org     0
0000  EB 4C                                       jmp     short LoadSystem

0002  90                                         nop                             ;DOS needs nop f.
0003  54 68 41 6C 76 61 38                       db      'ThAlva87'              ;  720K diskettes
      37

                                                 ;360K     720K  1.2M  1.44M
000B                            BPB:             ;=====================================
000B  0200                      SectorSize       dw      512   ;  512   512   512
000D  02                        SecPerCluster    db      2     ;    2     1     1
000E  0001                      ReservedSectors  dw      1     ;    1     1     1
0010  02                        NrofFATs         db      2     ;    2     2     2
0011  0070                      NrofDirEntries   dw      112   ;  112   224   224
0013  02D0                      NrofSectors      dw      720   ; 1440  2400  2880
0015  FD                        MediaID          db      0FDh  ;   F9    F9    F0
0016  0002                      SectorsPerFAT    dw      2     ;    3     7     9
0018  0009                      SectorsPerTrack  dw      9     ;    9    15    18
001A  0002                      NrofHeads        dw      2     ;    2     2     2
001C  00 00 00 00               HiddenSectors    dd      0     ;    0     0     0
0020  0070                      LoadSegment      dw      70h

0022  49 4F 20 20 20 20 20      TextIosys        db      'IO      SYS'
      20 53 59 53
002D  4D 53 44 4F 53 20 20      TextMsdos        db      'MSDOS   SYS'
      20 53 59 53
0038  49 42 4D 42 49 4F 20      TextIbmbio       db      'IBMBIO  COM'
      20 43 4F 4D
0043  49 42 4D 44 4F 53 20      TextIbmdos       db      'IBMDOS  COM'
      20 43 4F 4D

004E  ????                      RemainSectors    dw      ?
0050  ????                      SysStartSecLo    dw      ?
0052  ????                      SysStartSecHi    dw      ?
0054  000B[                    DiskBase         db      11 dup (?)
        ??
                ]

004E                                             org     offset RemainSectors
004E                            LoadSystem      proc far
004E  FC                                         cld
004F  FA                                         cli                             ;setup stack
0050  33 C0                                      xor     ax,ax
0052  8E D0                                      mov     ss,ax
0054  BC 7C00                                    mov     sp,BootOfs
0057  8E C0                                      mov     es,ax

                                ;--- Set a Disk Base

0059  BB 0078 R                                  mov     bx,offset Disk_Pointer ;in Abso seg
005C  26: C5 37                                   lds     si,es:[bx]
005F  BF 7C54 R                                  mov     di,offset BootOfs+DiskBase
0062  26: 89 3F                                  mov     es:[bx],di              ;set disk base
0065  B9 000B                                    mov     cx,11                   ;get parms
0068  F3/ A4                                     rep movsb
006A  0E                                         push    cs
006B  1F                                         pop     ds
                                                 assume  ds:Code
006C  8C 47 02                                   mov     [bx+2],es    ;set disk base vector seg
006F  C6 45 F9 12                                mov     byte ptr [di-7],18      ;set EOT
0073  C6 45 FE 01                                mov     byte ptr [di-2],1       ;head settle time
0077  FB                                         sti                             ;   ("read only")
0078  88 16 7DFD R                               mov     [BootOfs+BootDrive],dl
007C  52                                        push    dx


                                ;--- Set new value for Memory Size

007D  A1 7DFB R                                  mov     ax,[BootOfs+MaxRam]
0080  A3 0413 R                                  mov     ds:[Memory_Size],ax


                                ;--- Load first Directory Sector to 500h (PCDOS compatible mode)

0083  A0 7C10 R                                  mov     al,[BootOfs+NrofFATs]
0086  98                                         cbw
0087  F7 26 7C16 R                               mul     word ptr [BootOfs+SectorsPerFAT]
008B  03 06 7C0E R                               add     ax,[BootOfs+ReservedSectors]
008F  03 06 7C1C R                               add     ax,word ptr [BootOfs+HiddenSectors]
0093  13 16 7C1E R                               adc     dx,word ptr [BootOfs+HiddenSectors+2]
0097  A3 7C50 R                                  mov     [BootOfs+SysStartSecLo],ax
009A  89 16 7C52 R                               mov     [BootOfs+SysStartSecHi],dx
009E  E8 0155 R                                  call    SetupTrkHdSec
00A1  BB 0500                                    mov     bx,500h                 ;Abso:500h is
00A4  B0 01                                      mov     al,1                    ;  load address
00A6  E8 017B R                                  call    ReadSectors

                                ;--- Copy first half of Directory Sector to 600h for Tulip PC

00A9  B9 0080                                    mov     cx,128                  ;move 256 bytes
```

dann, wenn man die unteren 640 KByte Hauptspeicher 'aufräumt' und die sich resident machenden Helferlein (SideKick, Maustreiber und so weiter) in die Speichererweiterung auslagert. Obwohl einige Treiber (zum Beispiel DROPTASK[5]) 'unten' bleiben müssen und die SYS-Treiber auch noch einigen Platz beanspruchen, weist mein kompatibler PC unter MSDOS 3.10 nach diesen Auslagerungen beachtliche 580 KByte, unter Einbeziehung des EGA-Segments bei A000h 645 KByte zusammenhängend freien Hauptspeicher aus.

Auf geht's: hinweg mit SideKick ins Jenseits! Auch das Programm JENSEITS ist in Turbo-Pascal 4.0 geschrieben. Seine Funktion ist ganz simpel: es geht davon aus, daß beim Booten der größte zusammenhängende Speicherblock immer in den unteren 640 KByte liegt. Dieser Bereich wird einfach durch einen entsprechend lang vorgegebenen 'Heap' von 128 bis 640 KByte 'zugemacht', und dann wird das als Parameter übergebene Programm gestartet. Dem DOS bleibt gar nichts anderes übrig, als solche Programme in die Speichererweiterung zu laden.

Zu beachten ist, daß das zu ladende Programm mit voller Extension und vollem Pfadnamen anzugeben ist (sofern es sich nicht im aktuellen Directory befindet). Bis zu acht Parameter können auch noch übergeben werden. Beispiel:

```
JENSEITS C:\DOS\MOUSE.COM 1
```

Allesbooter

Das Verfahren, mittels CONFRAM eine Speichererweiterung anzubinden, hat zwangsläufig einen zweiten Bootvorgang zur Folge. Es geht aber auch in einem Durchgang, wenn man das Setzen der höchsten Speicheradresse im Bootsektor erledigt. Wer einen der gemeinen XT-Festplatten-Controller mit lokalem Speicherbedarf sein eigen nennt, hat eh keine andere Wahl.

Entsprechend universell ist der abgedruckte Bootsektor ausgelegt. Er bootet MSDOS wie PCDOS, und weil ich noch ein Tulip-IO.SYS 1.13 in Gebrauch habe, das nicht ganz den Standards entspricht, wird dessen besondere Anforderung an die

```
00AC  8B F3                   mov     si,bx
00AE  BF 0600                 mov     di,600h                 ;Abso:600h is
00B1  F3/ A5                  rep movsw                       ;  copy address

                        ;--- Check if one of two valid BIOS names

00B3  B0 02                   mov     al,2                    ;check 2 names
00B5  BE 7C22 R               mov     si,offset BootOfs+TextIosys
00B8                    LoadSystem_1:
00B8  8B FB                   mov     di,bx
00BA  B1 0B                   mov     cl,11
00BC  F3/ A6                  repe cmpsb
00BE  74 0D                   je      LoadSystem_2            ;if valid name

00C0  BE 7C38 R               mov     si,offset BootOfs+TextIbmbio
00C3  FE C8                   dec     al
00C5  75 F1                   jnz     LoadSystem_1
00C7                    DispInvDosMsg:
00C7  BE 7DD6 R               mov     si,offset BootOfs+Msg_InvalidDos
00CA  E9 01AE R               jmp     DisplayMessage

                        ;--- Check for valid first data cluster and valid DOS name

00CD                    LoadSystem_2:
00CD  83 7F 1A 02             cmp     word ptr [bx+1Ah],2
00D1  75 F4                   jne     DispInvDosMsg           ;if not first
                                                              ;  data cluster
00D3  8D 7F 20                lea     di,[bx+20h]
00D6  B1 0B                   mov     cl,11
00D8  F3/ A6                  repe cmpsb
00DA  75 EB                   jne     DispInvDosMsg

                        ;--- Calculate Count of Sectors to Load (max. 28K, otherwise
                        ;    Loader required)

00DC  8B 47 1C                mov     ax,word ptr [bx+1Ch]    ;lo file size
00DF  8B 57 1E                mov     dx,word ptr [bx+1Eh]    ;hi file size
00E2  3B D1                   cmp     dx,cx                   ;CX = 0 here
00E4  77 05                   ja      LoadSystem_3

00E6  3D 7000                 cmp     ax, 28 * 1024
00E9  76 05                   jbe     LoadSystem_4
00EB                    LoadSystem_3:
00EB  33 D2                   xor     dx,dx
00ED  B8 7000                 mov     ax, 28 * 1024
00F0                    LoadSystem_4:
00F0  8B 2E 7C0B R            mov     bp,[BootOfs+SectorSize]
00F4  F7 F5                   div     bp                      ;SectorSize
00F6  40                      inc     ax
00F7  A3 7C4E R               mov     [BootOfs+RemainSectors],ax

                        ;--- Calculate Sector of first Data Cluster

00FA  B8 0020                 mov     ax,20h                  ;size of 1 dir
00FD  F7 26 7C11 R            mul     [BootOfs+NrofDirEntries]; entry in bytes
0101  03 C5                   add     ax,bp                   ;SectorSize,
0103  48                      dec     ax                      ;supports partly
0104  33 D2                   xor     dx,dx                   ;used dir sectors
0106  F7 F5                   div     bp
0108  BB 7C50 R               mov     bx,offset BootOfs+SysStartSecLo
010B  01 07                   add     ds:[bx],ax
010D  11 4F 02                adc     [bx+2],cx               ;CX = 0 here
0110  FF 37                   push    ds:[bx]                 ;logical IBMBIO
                                                              ;  start sector
                        ;--- Load BIOS into RAM (to 70H:0)

                              assume  es:nothing
0112                    LoadSystem_5:
0112  A1 7C50 R               mov     ax,[BootOfs+SysStartSecLo]
0115  8B 16 7C52 R            mov     dx,[BootOfs+SysStartSecHi]
0119  E8 0155 R               call    SetupTrkHdSec
011C  3B 06 7C4E R            cmp     ax,[BootOfs+RemainSectors]
0120  76 03                   jbe     LoadSystem_6

0122  A1 7C4E R               mov     ax,[BootOfs+RemainSectors]
0125                    LoadSystem_6:
0125  33 DB                   xor     bx,bx
0127  8E 06 7C20 R            mov     es,[BootOfs+LoadSegment]
012B  E8 017B R               call    ReadSectors
012E  01 06 7C50 R            add     [BootOfs+SysStartSecLo],ax
0132  11 1E 7C52 R            adc     [BootOfs+SysStartSecHi],bx ;BX = 0 here
0136  93                      xchg    bx,ax                   ;loaded sectors
0137  A1 7C0B R               mov     ax,[BootOfs+SectorSize]
013A  B1 04                   mov     cl,4
013C  D3 E8                   shr     ax,cl
013E  F7 E3                   mul     bx
0140  01 06 7C20 R            add     [BootOfs+LoadSegment],ax
0144  29 1E 7C4E R            sub     [BootOfs+RemainSectors],bx
0148  75 C8                   jnz     LoadSystem_5            ;if more sectors
                                                              ;  to load
                        ;--- Start Operating System

014A  5B                      pop     bx                      ;logical IBMBIO
014B  8A 2E 7C15 R            mov     ch,[BootOfs+MediaID]    ;  start sector
014F  5A                      pop     dx                      ;DL = physical
0150  EA 0000 ---- R          jmp     StartIosys              ;  boot drive
                        LoadSystem    endp


                        ;--- SetupTrkHdSec
                        ;  --> DX:AX   logical sector
                        ;  <--  CH, CL, DH, DL as required, AX=max.loadable in track
0155                    SetupTrkHdSec  proc near
0155  F7 36 7C18 R            div     word ptr [BootOfs+SectorsPerTrack]
0159  8A CA                   mov     cl,dl
```

Ladeadresse des Directory-Sektors auch noch mit erledigt. Und sogar PCDOS 4.0 läßt sich booten.

Eine weitere angenehme Eigenschaft macht sich bei Diskettenfehlern bemerkbar, weil statt eines nichtssagenden Universaltextes die BIOS-Fehlermeldung ausgegeben wird. Zusammen mit dem Tulip-IO.SYS und MSDOS 3.10 lassen sich Festplatten-Partitionen oberhalb des IBM-Limits von 32 MByte booten. In seinen sonstigen Aktivitäten bleibt der Bootsektor aber kompatibel zu PCDOS 2.10 bis 4.00.

Wesentlicher Bestandteil des Bootsektors ist außerdem der BIOS-Parameter-Block (BPB). Er beschreibt die Organisationsstruktur einer Diskette oder Festplatte, ohne deren Kenntnis MSDOS darauf keine Daten verwalten kann. Im Assemblerlisting sind neben dem Standard-360K-Format noch die Parameter für drei weitere häufig benutzte Diskettenformate angegeben, wovon das zutreffende eingesetzt werden muß. Für Festplatten-Bootsektoren sollte man das Programm zunächst unverändert assemblieren und den BPB später mit DEBUG nachtragen.

Bevor BOOTBOTH assembliert wird, müssen aber noch Parameter an die Erfordernisse des Rechners und der beabsichtigten Speicheranbindung angepaßt werden. Besonders wichtig ist, daß die Konstante 'Kilobytes' exakt zum ersten Parameter von CONFRAM passen muß, also zum Beispiel

```
Kilobytes EQU 960
```

und

```
CONFRAM F000 9FC0-D000
```

Die sonstigen abänderbaren Parameter sind dem Kopf des Assemblerlistings zu entnehmen. Das Assemblieren selbst geht folgendermaßen vor sich:

```
MASM    BOOTBOTH.ASM,,BOOTBOTH;
LINK    BOOTBOTH;
EXE2BIN BOOTBOTH
```

Übertragen des Bootsektors auf eine Diskette im Laufwerk A:

```
DEBUG   BOOTBOTH.BIN
W 100 0 0 1
Q
```

Übertragen des Bootsektors auf die Festplatte, Laufwerk C:

```
015B  41                        inc     cx                      ;sector
015C  33 D2                     xor     dx,dx
015E  F7 36 7C1A R              div     word ptr [BootOfs+NrofHeads]
0162  8A F2                     mov     dh,dl                   ;head
0164  8A E8                     mov     ch,al                   ;track
0166  D1 E8                     shr     ax,1                    ;upper 2 bits of
0168  D1 E8                     shr     ax,1                    ;  track to CL
016A  24 C0                     and     al,0C0h
016C  0A C8                     or      cl,al
016E  8A 16 7DFD R              mov     dl,[BootOfs+BootDrive]
0172  A1 7C18 R                 mov     ax,[BootOfs+SectorsPerTrack]
0175  40                        inc     ax
0176  2A C1                     sub     al,cl
0178  24 3F                     and     al,3Fh
017A  C3                        ret
                        SetupTrkHdSec   endp

017B                    ReadSectors     proc near
017B  BE 0005                   mov     si,5                    ;retry count
017E                    ReadSectors_1:
017E  50                        push    ax
017F  B4 02                     mov     ah,2                    ;read sector(s)
0181  CD 13                     int     13h
0183  72 02                     jc      ReadSectors_2
0185  58                        pop     ax
0186  C3                        ret
0187                    ReadSectors_2:
0187  8A C4                     mov     al,ah
0189  D4 10                     db      0D4h,10h                ;= aam 16, MASM
018B  3C 0A                     cmp     al,10                   ; doesn't like it
018D  72 02                     jb      ReadSectors_3

018F  04 07                     add     al,7
0191                    ReadSectors_3:
0191  86 E0                     xchg    ah,al
0193  3C 0A                     cmp     al,10
0195  72 02                     jb      ReadSectors_4

0197  04 07                     add     al,7
0199                    ReadSectors_4:
0199  0D 3030                   or      ax,3030h
019C  A3 7DD2 R                 mov     [BootOfs+ErrorCode],ax ;f. error display
019F  52                        push    dx
01A0  33 C0                     xor     ax,ax                   ;reset drive
01A2  B6 00                     mov     dh,0
01A4  CD 13                     int     13h
01A6  5A                        pop     dx
01A7  58                        pop     ax
01A8  4E                        dec     si
01A9  75 D3                     jnz     ReadSectors_1           ;retry

01AB  BE 7DC0 R                 mov     si,offset BootOfs+Msg_DiskError
                        ReadSectors     endp


01AE                    DisplayMessage proc near
01AE  AC                        lodsb
01AF  0A C0                     or      al,al
01B1  75 02                     jnz     DisplayMessage_1

                        ;*** wenn bei Boot-Fehlern ROM BASIC angesprungen werden soll:
                        ;*** aus dem nachfolgenden "jmp $" ein "int 18h" machen

01B3  EB FE                     jmp     $
01B5                    DisplayMessage_1:
01B5  BB 0007                   mov     bx,7                    ;screen attribute
01B8  B4 0E                     mov     ah,0Eh                  ;TTY output
01BA  56                        push    si
01BB  CD 10                     int     10h
01BD  5E                        pop     si
01BE  EB EE                     jmp     DisplayMessage
                        DisplayMessage endp


01C0  0D 0A 42 49 4F 53 20      Msg_DiskError   db      13,10,'BIOS disk error '
      64 69 73 6B 20 65 72
      72 6F 72 20
01D2  3030                      ErrorCode       dw      '00'
01D4  48 00                     BootDrive       db      'H',0
01D6  0D 0A 44 4F 53 20 6E      Msg_InvalidDos  db      13,10,'DOS not found or invalid'
      6F 74 20 66 6F 75 6E
      64 20 6F 72 20 69 6E
      76 61 6C 69 64
01F0  00                        BootIndicator   db      0

01FB                            org     1FBh
01FB  03C0                      MaxRam          dw      Kilobytes               ;in Kbytes
01FD  ??                        BootDrive       db      ?
01FE  AA55                      BootIndicator   dw      0AA55h

0200                    Code    ends
                        end
```

```
DEBUG   BOOTBOTH.BIN
L 300 2 0 1
M 30B L 15 10B
W 100 2 0 1
Q
```

Bei einigen, besonders leistungsfähigen MSDOS-Rechnern (Partitionsstart oberhalb 32 MByte) sollte der Wert 13 gegen 15 ersetzt werden. Soll die vom Bootsektor gesetzte höchste RAM-Adresse ('MaxSeg') später geändert werden, dann ist erneutes Assemblieren nicht erforderlich, sofern man den Hex-Wert der neuen Endadresse in Kilobyte ermittelt und im 508. und 509. Byte des Bootsektors einträgt. Das Übertragen in den Disketten-Bootsektor in Laufwerk A:

```
DEBUG
L 100 0 0 1
E 2FB
(niederwertiges Byte)
(Leertaste)
(höherwertiges Byte)
W 100 0 0 1
Q
```

Für diese DEBUG-Operationen darf keinesfalls SYMDEB Version 4.0 benutzt werden, der stürzt nämlich bei 'L 100 . . .' und 'W 100 . . .' aufgrund eines von Microsoft eingebauten Stack-Fehlers ab.

Nur der Vollständigkeit halber sei vermerkt, daß der Bootsektor nicht das CONFRAM im AUTOEXEC.BAT ersetzt. Er reduziert nur den Doppel-Bootvorgang um eine Runde.

(mw)

Literatur

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

[2] Andreas Landenberger, Booten mit List, c't 11/87, S. 154

[3] Köhlmann, Rubel, Wilde, Zwischen den Adaptern, c't 5/88, S. 164

[4] Thomas Bergler, Betriebssystem-Forscher, c't 9/87, S. 174

[5] Ralf Preller, Notausstieg, c't 1/88, S. 120

Der Bootsektor als Assembler-Protokoll. Dieses Programm sollte mit größter Sorgfalt eingegeben werden, weil Tippfehler zu Datenverlusten führen können. Zudem sollten Sie es erst auf einer Diskette testen, bevor Sie es auf die Festplatte loslassen.

```
{$R+,S+,D-,T-,F-,V+,B-,N-,L+ }
{$M 16384,0,0 }                 { <==== die letzte 0 ist wichtig }

{*************************************************************}
{*                       CONFRAM.PAS                         *}
{*        konfiguriert RAM so, dass RAM oberhalb der         *}
{*         Video-Adapter unter DOS verfuegbar wird           *}
{*              (c) Ralf Preller 30.8.1988                   *}
{*                                                           *}
{*  Aufruf:                                                  *}
{*       CONFRAM  MaxSeg  StartSkipSeg-EndSkipSeg            *}
{*                                                           *}
{*  mit  MaxSeg        hoechste RAM-Segmentadresse + 1       *}
{*       StartSkipSeg  erste nicht verfuegbare Segmentadresse*}
{*                       im "Adapter-Loch"                   *}
{*       EndSkipSeg    erstes wieder verfuegbares Segment    *}
{*                       hinter diesem Loch                  *}
{*                                                           *}
{*  Die Parameter sind als hexadezimale Segmentadressen anzu-*}
{*  geben. Erlaubte Parameterschreibweisen (auch gemischt    *}
{*  verwendbar):                                             *}
{*  - fuehrende Nullen und 'H' hinter der Hexzahl erlaubt    *}
{*  - Parametertrennzeichen: Leerzeichen, Komma oder '-',    *}
{*                                                           *}
{*            mit Turbo 4.0 zu kompilieren                   *}
{*************************************************************}
PROGRAM ConfRAM;

USES
  Dos;

CONST
  VersionSt     = '1.00';

VAR
  MaxSeg,
  StartSkipSeg,
  EndSkipSeg    : word;


PROCEDURE GetCommandLine;
  CONST
    ETX         = #3;
    HexCharSet  : set of char = ['0'..'9','A'..'F','a'..'f'];
  VAR
    I           : byte;
    ParamSt     : string;

  FUNCTION GetParameter : word;
    VAR
      CharCount : byte;
      Number    : word;
    BEGIN
    CharCount := 0;  Number := 0;
    while ParamSt[1] = '0' do Delete (ParamSt, 1, 1);
    while (CharCount < 4) and (ParamSt[1] in HexCharSet)
    do begin
      Number := Number shl 4 + ord(Upcase(ParamSt[1])) - 48
        - 7 * ord(Upcase(ParamSt[1]) >= 'A');
      inc (CharCount);
      Delete (ParamSt, 1, 1);
      end;
    while ParamSt[1] = ' ' do Delete (ParamSt, 1, 1);
    GetParameter := Number;
    END;

  BEGIN
  ParamSt := ParamStr(1) + ' ' + ParamStr(2) + ' ' + ParamStr(3)
    + ' ' + ETX;
  repeat
    I := Pos (',', ParamSt);
    if I = 0 then I := Pos ('-', ParamSt);
    if I = 0 then I := Pos ('H ', ParamSt);
    if I = 0 then I := Pos ('h ', ParamSt);
    if I > 0 then ParamSt[I] := ' ';
    until I = 0;
  MaxSeg := GetParameter and $FFC0;{nur 1Kbyte-Schritte erlaubt}
  StartSkipSeg := GetParameter;  EndSkipSeg := GetParameter;
  if (ParamSt <> ETX) or (MaxSeg < $4000)
    or (StartSkipSeg >= MaxSeg) or (EndSkipSeg >= MaxSeg)
    or (StartSkipSeg <> 0) and (
    (StartSkipSeg < $4000) or (EndSkipSeg <= StartSkipSeg))
  then begin
    writeln (
      'Parameterfehler in Aufrufzeile! Programm abgebrochen');
    Halt(1);
    end;
  END;


FUNCTION Max (No1, No2 : word) : word;
  BEGIN
  if No1 >= No2 then Max := No1 else Max := No2;
  END;


CONST
  ResetSequence = $1234;

TYPE
  tMCB          = record
                    MCBchar : char;
                    MCBPSP  : word;
                    MCBsize : word;
                    end;
```

```
VAR
  I             : word;
  MCBptr        : ^tMCB;
  MemorySize    : word absolute $0:$413;
  Regs          : Registers;
  ResetFlag     : word absolute $0:$472;
  St            : string;

BEGIN
writeln ('CONFRAM  RAM-Erweiterungsverwaltung  Version ',VersionSt);
if ParamCount = 0
then writeln ('Momentan gesetzte Speichergroesse: ', MemorySize,
  ' Kbyte')
else begin
  GetCommandLine;
  if (MaxSeg shr 6 <> MemorySize)
    and (MaxSeg shr 6 <> succ(MemorySize))     { falls Controller }
  then begin                                   {  1 Kbyte belegt  }
    {--- 1st Pass ---}
    MemorySize := MaxSeg shr 6;
    ResetFlag  := ResetSequence;
    Intr ($19, Regs);        { Reboot System }
    end
  else begin
    {--- 2nd Pass ---}
    {--- clear RAM above 640K to avoid parity errors ---}
    for I := Max($A000,EndSkipSeg) to MemorySize shl 6 - 1 do
      Fillchar (ptr(I,0)^, 16, #0);
    if StartSkipSeg > 0 then begin
      for I := $A000 to StartSkipSeg-1 do
        Fillchar (ptr(I,0)^, 16, #0);

      {--- patch Memory Control Blocks ---}
      MCBptr := ptr (PrefixSeg-1, 0);
      MCBptr^.MCBsize := StartSkipSeg - PrefixSeg - 1;
      MCBptr := ptr (StartSkipSeg-1, 0);
      MCBptr^.MCBchar := 'M';
      MCBptr^.MCBPSP  := StartSkipSeg;
      MCBptr^.MCBsize := EndSkipSeg - StartSkipSeg;
      MCBptr := ptr (EndSkipSeg, 0);
      MCBptr^.MCBchar := 'Z';
      MCBptr^.MCBPSP  := 0;
      MCBptr^.MCBsize := MaxSeg - EndSkipSeg - 1;
      end;
    writeln ('DOS-verfuegbares RAM jetzt ',
      MemorySize - (EndSkipSeg - StartSkipSeg + $3F) shr 6,
      ' Kbyte');
    end;
  end;
END.
```

CONFRAM stellt dem DOS einen erheblich erweiterten Speicherbereich zur Verfügung, indem es den Video-Adapter ausblendet.

```
{$R+,S+,I+,D-,T-,F-,V+,B-,N-,L+ }
{$M 65520,130072,655360 }               { <=== ganz wichtig !!! }

{*************************************************************}
{*                      JENSEITS.PAS                         *}
{*    laedt Programme in Speichererweiterung jenseits des    *}
{* Video-Adapters. Max.8 Parameter koennen uebergeben werden.*}
{*              (c) Ralf Preller 1.9.1988                    *}
{*                                                           *}
{*  Aufruf:                                                  *}
{*      JENSEITS  [pfad]progname.ext  param1  param2  ...    *}
{*                                                           *}
{*            mit Turbo 4.0 zu kompilieren                   *}
{*************************************************************}
PROGRAM Jenseits;

USES
  Dos;

VAR
  Path,
  CmdLine : string;
  I       : byte;

BEGIN
Path := ParamStr(1);
CmdLine := ParamStr(2);
for I := 3 to 9 do CmdLine := CmdLine + ' ' + ParamStr(I);
while CmdLine[Length(CmdLine)] = ' ' do dec (CmdLine[0]);
Exec (Path, CmdLine);
if DosError <> 0 then begin
  writeln ('JENSEITS *** DOS-Fehler ', DosError, '.  ',
    Path, ' nicht installiert !!!');
  Halt(255);
  end
else Halt(DosExitCode);
END.
```

Winzling mit großer Wirkung: JENSEITS lädt Programme in das RAM oberhalb der Bildschirmadapter. ct