香川大学経済論叢
第64巻 第2・3号 1991年11月 281-310

MS-DOSシステムの最適化

中村邦彦

I はじめに

筆者は以前にMS-DOSの操作性を改善するためのバッチプログラム技法についていくつかの提案をしたが[1]，その後約3年を経てMS-DOSをめぐるパソコン環境にはかなりの変化が見られる。まず，ハードウェアはCPUが80386へと大きくシフトした。そのため大容量のプロテクトメモリ(1)を搭載可能な機種が増えた。メモリの使用方法も従来のバンク切り換え方式のRAMディスクに加えて，EMS(2)，XMS(3)，BMS(4)，UMB(5)などの使い方が開発された。1ドライブ40MB以上の大容量のハードディスクが普及した。

他方ソフトウェアの方は，MS-DOS本体がバージョン3.xxから4.0，5.0へと発展してきた。ただし，これはマイクロソフト社の版であり，NEC版は現在3.3Cである。当時にぎにぎしくデビューしたOS/2は期待されたほど普及せず，MS-Windows3.0が発表されてからは，GUIはそちらの方が主流になりそうな雲行きである。アプリケーションプログラムの多くはEMS対応になった。操作環境を改善するためのツールも数多く開発された。

これらの変化により，先の論文で述べたことが一部陳腐化してしまったこと，また，MS-DOSの操作環境を改善する方法としては，バッチファイルの技

---

(1) 1MB以上のアドレス空間に配置されている主記憶。英語ではextended memoryと呼んでexpanded memoryと区別するが，これを拡張メモリと呼ぶと紛らわしいのでここではプロテクトメモリと呼ぶことにする。
(2) expanded memory specification
(3) extended memory specification
(4) bank memory specification
(5) upper memory block

法のみでなく，もっと広い観点から検討する必要を感じたことから，今回はMS-DOSのシステム環境を最適化するための方法をいくつか検討してみることにした。

本論で検討するのは次の3点である。

(1)　システム構成の最適化

(2)　ハードディスクの最適化

(3)　MS-WINDOWS3.0の利用法

(1)ではMS-DOSマシンとしてのハードウェアシステム構成，ソフトウェアシステム構成の最適化について検討する。特にアプリケーションのための領域を大きく残すための技法を紹介する。(2)では大容量ハードディスクの初期化とシステム効率の問題を考える。(3)ではMS-DOSアプリケーションを実行するためにMS-Windows3.0の386エンハンスドモードを利用する方法を紹介する。

Ⅱ　システム構成の最適化

1.　ハードウェアシステム

ここで検討することは，MS-DOSのもとで，いかにして大きなユーザーメモリを確保するかということである。その際に実用的な速度を維持することは大前提である。こういう問題が浮上してきたのは，MS-DOS自身がその機能強化に伴って大きくなってきたこと，アプリケーションプログラムが肥大化していることによる。本来80286なら16MB，80386なら4GBの広大な主記憶空間を直接アクセスできるはずであるが，MS-DOSは未だに8086のモードでしか走らない。8086のモードではメモリ空間は1MBしかなく，そのうち384KBはシステムROMやVRAM，拡張ボード用に使用または予約されているので，利用可能な主記憶は640KBとなってしまう。たとえXMSやEMSを利用しても，MS-DOS[(6)]においてはこの640KBの部分が基本になるので，これを基本メモリと呼ぶことにする。そこでシステムを考える場合，できるだけ大きな基本メモリをア

(6)　英語ではconventional memoryと呼んでいるので通常メモリと呼ぶ場合もある。

プリケーションのために残すことを考えなければならない。

MS-DOSの最新バージョンである5.0（以後DOS 5と呼ぶ）では，まさにその点についての配慮がなされ，4.0で拡張された機能の一部を削ってまでも，基本メモリを多く残すようになった[2]。すなわち，DOSのレベルでHMA(7)やUMBを正式にサポートし，DOSの本体やデバイスドライバを基本メモリの外に追い出した。その結果DOSは基本メモリの約18KBしか使わなくてすむようになった。NECのDOS3.3Cではほとんど何も組み込まないシステムでも90KBは必要であることを考えれば，これは大きな改善である（リスト5参照）。

ここではDOS3.xxのもとで基本メモリを大きく残す方法を考える。HMAは80286以上の機種で利用可能である。UMBは8086，V30，80286等の機種でも理論的には利用可能であるが，80386のページマッピング機能を利用しないと本格的な利用は不可能である。これらの事情から本論では，対象マシンとして第1に次のような構成を仮定する。

CPU： 80386SX以上

主記憶： 640KB＋プロテクトメモリ3MB

MS-DOSでは直接プロテクトメモリをアクセスすることはないので，増設メモリは必ずしもプロテクトメモリである必要はないが，後にMS-Windows3.0を利用することを考えて，全てプロテクトメモリとして増設してあるものとする。

そうは言っても，CPUがV30や80286のシステムを使用しているユーザーも多いと思われるので，そのようなシステムについても少し検討してみることにする。

なお，本論における各種テスト並びに測定を行ったのは以下のシステムである。

NEC製PC-98XL²

CPU 80386 クロック16MHz

---

(7) High Memory Area

主記憶　640KB＋内蔵プロテクトメモリ３MB

ハードディスク：内蔵40MB，増設ICM製SRC-131（130MB）

また，386の機能を持たない機種としては，NECのPC-9801VM（CPU V30，クロック10MHz）を用いた。

2. システムコンフィギュレーション

・仮想86モードで利用する

同じMS-DOSマシンであっても，ソフトウェア的には種々の構成が可能であるが，80386マシンならば仮想86モードで使うことを勧めたい。そうすれば増設メモリは全てプロテクトメモリのみにしておいても，HMA，BMS，EMSとしての使い分けは自由にできる。

仮想86モードを利用するには，仮想86用EMSドライバをCONFIG.SYSに組み込む。このタイプのドライバーには２種類ある。１つはプロテクトメモリをアクセスするときだけ仮想86モードにするもので，他の１つは完全に仮想86モードで動作するものである[3]。アイ・オー・データ機器のVMM386.SYSは前者に，NECのEMM386.SYS，メガソフトのMEMORY-PRO386，メルコのMELEMM.386は後者になる。ページマッピング機能を利用してROMのアドレスを移動できるのは後者である。筆者はMEMORY-PRO386を使用しているが，これは豊富な機能が提供され，動作も安定している。(8)

CPUを仮想86モードで走らせると，リアルーモードよりスピードが落ちるはずであるが[4]，Dhrystone(9)を含むいくつかのベンチマークテストの結果は同じであった。あまり気にすることはないと思われる。

・RAMディスクに１MBをつかう

現在では，主要なアプリケーションソフトウェアと日本語FEPは，ほとんどEMS対応になっている。またEMS対応のTRS(10)プログラムも数多く開発されて

(8)　筆者のシステムではPC-98XL²-07を使用してくださいと警告が出るが，問題なく動作しているのでそのままにしている。

(9)　CPUの性能を測るベンチマークプログラムの１つ。

(10)　terminated and stay resident

いるので，EMSは必要であろう。

RAMディスクはどんなディスクキャッシュプログラムを使うより高速である。作業ドライブとして1MB程度は用意しておきたい。UMBのために64KBを確保して残りはEMSとして使用するのが妥当なところだろう。

ディスクキャッシュプログラムを使用している人をよく見かけるが，最近のハードディスクは高速のものが多くなっていること，ディスクキャッシュ内蔵のものも多くなっていることから[5]，これは使用しない。下手に使うと返って遅くなることがあるからである。フロッピーディスクドライブに対しては効果があるように考えられるが，それは同じデータを何度も読み出す場合のことであって，一度読むだけの処理ではほとんど効果がない。書き込むときは同じか，返って時間がかかるはずである。フロッピーディスクはファイルの交換と保存用と考えておけばよいだろう。

さて筆者が勧めるCONFIG.SYSと，それに対応したAUTOEXEC.BATを紹介しよう（リスト1，リスト2）。両方とも説明の都合上，余計なものは削除してある。バッファは1つにつきセクタサイズ＋16バイトのメモリが必要になる。buffers＝1となっているのは，ここではできるだけ小さくして基本メモリの消費を抑えておき，実際にはUMBの中に確保しようという訳である。

VEMS.DRVというのはMEMORY-PRO386の仮想EMSドライバーである。MX1STAR.DRVはバンク方式のRAMディスクドライバーで，これはUMBにおく。UMBにおくといっても，現状では全てを外に出す訳にはいかず，実際にはいくらか基本メモリに残る。この例では304バイトが基本メモリに残った。

PRINT.SYSは文字型ドライバであるので，後でも組み込みと取り外しが可能であるが，これもUMBに置くことにすれば，常に組みこんでおいても問題はない。

AUTOEXEC.BAT（リスト2）の中のENVDRVというのは環境変数にそのシステムのドライブ名を登録するプログラムである[6]。環境変数領域の見つけ方，RAMディスクの認識の仕方など，一部を修正して使用している。

これで主記憶がどの程度節約されたか実測してみよう。リスト3にリスト1

のCONFIG.SYSを用いたときのメモリマップを示す。これはVMAP.COM[11]による出力であるが，{…} の部分は筆者があとで書き加えたものである。

何も組み込まない場合（リスト4，リスト5）はフリーエリアが563184バイトである。リスト3の最大のフリーエリアは581808であるから，18624増加している。これはほとんどBUFFERSの数を減らしたことによっている。システム領域はFILESの数や接続ドライブ数によっても影響を受けるが，これは余り大きくない。

VEMS.DRV自身は基本メモリを160バイト，XMSを8ページ使用する。その他UMB用に4ページ使用する。このタイプのドライバは基本メモリ占有量が小さい。VEMS.DRVのオプション/B10によってRAMディスク用にバンクメモリを10バンク（128KB×10）を確保させているが，オプション/Xにより256KBの裏RAMをバンクメモリとして利用してくれるので，実際にはXMSは8バンクが消費される。

・UMBをDOSの管理下に組み入れる

リスト3のメモリマップにはUMBまで含まれているが，これはMCB[12]をUMBまで延長したからである。こうしておけばUMBの空き領域もMS-DOSの管理下に入るので，DOSから利用できるようになる。このためのプログラムLKUMB.COMは筆者が作成したものであるが，MELWAREのUMBSTAT.COMも利用できた。一旦連結すると，MEMORY-PRO386付属のUMB管理プログラムLU.COMは使用できなくなってしまったので，LKUMB.COMには連結を切断するオプションを付け加えた。その他ICMのEOシステムも使えなくなった。連結後はLOAD.COM[13]，ADDBUF.COM[14]が利用できた。リスト2で

---

(11)　エディタVzに付いて来るメモリマップを表示するプログラム。作者　兵藤嘉彦，フリーウェア

(12)　memory control block. MS-DOSの主記憶割当て状況を管理するために使用される。

(13)　常駐プログラムを拡張エリアにロードするプログラム。作者　junk 35（日経MIX），フリーウェア

(14)　バッファをUBMに割り付けるプログラム。作者　junk 35（日経MIX），フリーウェア

'addbuf 30' とあるのは，基本メモリ内の分と合わせて30になるようにUMBにバッファを確保することを意味する．LOAD. COMによりTSRプログラムAUTOLAND. COM[15]，CRTSHUT. COM[16]，EZKEY. COM[17]をUMBにロードしている。

また，そのままではMS-DOSが新たにメモリを割り付けるときには，下位領域から割り付けるようになっているので，LKUMB. COMはDOSのファンクション58H Get/Set Allocation Strategyにより，最小の利用可能ブロックから割り当てるようにしてある。これも変更できるようにCHGSTR. COM[18]が用意されている。

残念なことに，これらの工夫をしても，環境変数領域をUMBに取ってくれる程度で，期待したほどにはうまくいかなかった。これは基本的にDOSシステムがUMB対応になっていないからで，DOS 5が使えるようになるまでの一時しのぎというところである。

・CPUが80286以下の場合

次に80386の機能を持たないCPUについて検討してみる。まずハードウェアの構成であるが，EMS対応のソフトを使いたいのなら2MB程度のEMSボードが必要である。さらにRAMディスクも使いたいなら，EMSを3MB以上にして1部を利用するか，別に従来のバンク方式のメモリがあればそれを利用する。

EMSボードとバンク方式のメモリボードが異なるメーカーの製品であるときは，同時に使用できない場合がある。例えばエービーエム社のバンク方式メモリボード98-MBとメルコのEMSボードEMJmkⅢは，制御に同じI/Oポートを使用しているために同時に実装することはできない。

---

(15) ハードディスクのヘッドをシッピングゾーンに移動させるプログラム。作者 serow（アスキーネット），フリーウェア

(16) CRTの焼き付き防止プログラム。作者 Takuya-N!SS，フリーウェア

(17) キーボード入力を補助するプログラム。作者 兵藤嘉彦，フリーウェア

(18) メモリ割当て方法を変更するプログラム。作者 junk. 35（日経MIX），フリーウェア

この場合EMSはハードウェア方式で実現することになるので，EMMはそれぞれのボードに対応したものを使用しなければならない。どんな場合でも何か追加すればそれだけ貴重な基本メモリが消費されることになるので，この場合もUMBを利用することを考えるべきである。UMBの利用は，386のようにうまくはいかないが，ある程度可能である。例としてNECのPC-9801VM2にアイ・オー・データ機器のEMSボードPIO-PC34HX（2MB），エービーエムのバンク方式のメモリボード98-MB（2MB），SASIのハードディスクという構成での実例を紹介する。

ソフトウェアはEMMとして，同じアイ・オー・データ機器のEMM4J.SYSとバンク式メモリボード対応のRAMディスクドライバを用意する。80286の場合はプロテクトメモリを利用することが可能である。HMAやXMSを利用するプログラムも使いたい場合は，同じメモリをハードウェア方式のEMSとしてもプロテクトメモリとしても利用可能なボードを増設すべきでる。

アイ・オー・データ機器から提供されるもので構成する場合はリスト6，7のようにすればよいだろう。EMM4J.SYSで/wの後にd4が書いてないのは，セグメントD700からD7FFまでは，SASIのROMがあるので使用できないからである。続いて/A2　/Bが指定してあるが，これには次の意味がある。

/A1　EMMのデータ部分をUMBに移す
/A2　EMMのコード部分をUMBに移す
/B　　UMBをその他の目的に利用する

/A1を指定するとEMSの速度が向上する。/A2を指定すると，スピードは遅くなるが基本メモリの消費が少なくなる[7]。

LUMB.COM，BEX.COMはアイオーデータの製品でUMBにプログラムをロード，およびバッファを確保するプログラムである。リスト7でBEX.COMを2度実行しているが，この例のようにUMBが分断されているときは，一度に必要バッファを確保できなかったからである。メモリマップをリスト8に示す。

先の設定では基本メモリを空けるためにEMSのスピードを犠牲にしたので

あるが，それがどの程度であるか調べておこう。EMSそのものの速度はEMSBENCH.COM[19]で測定した（表7）。EMSを使用するアプリケーションの速度についてはLotus123で整数値データのコピー時間を測定した（表8）。その結果次のようなことがわかった。

EMSメモリへのアクセスは全ての場合について同じであるが，その他のEMSファンクションについては，最も速い/A1は/A2より1.1倍から1.6倍，平均で1.4倍高速である。実際のアプリケーションプログラムではその差は1.15倍であった。

ところで，アイ・オー・データのやり方ではSASIのROMがあるページ全体が利用できなかったが，ROMのないところは全て使えようにするプログラムが発表されている。EXT_TPA.COM[20]がそれである。今度はEMSのウインドウにC0からDCまで全てのページが指定できる。/A2指定をしなければD0からDCまでのページのうち，ROMのある4KBを除き，残り60KB全てがUMBとして利用可能になる。残念なことにEXT_TPA.COMでは，ブロック型デバイスドラバをUMBにロードすることができないので，RAMディスクドライバが基本メモリ内に残る。その結果基本メモリ内の空き領域は約1KB減少するが，UMBは11KB増加している（リスト9，10，11）。

さて，このように基本メモリを節約することのメリットであるが，次のようなことが考えられる。

(1) 大きなアプリケーションが余裕をもって使える。

(2) 常駐型プログラムが使用可能になる。

(3) 1つのプログラムを起動したまま，別のプログラムを呼び出して処理をする。

(3)はエディタの中からコンパイラを起動したり，あるプログラムを実行中に

---

(19) EMSの速度を測定するプログラム。作者 内田暁，フリーウェア[3]

(20) EMSの物理ページをUMBに割り付けて使用するプログラム。作者 junk 35（日経MIX），フリーウェア。MELWARE専用の部分があるが，それ以外はアイ・オ・データのEMSでも利用できた。

一時的にDOSへ抜けて，別の処理をしたりすることをさしている。最近のソフトウェアの多くはこのようなことができるようになっている。この点で徹底してるのはPARADOX(21)で，使用中に一時的にDOSに抜けたときは，自分自身はわずか1.3KBを基本メモリ内に残すのみで，あとは全部EMSに待避している（リスト12)。UMBにはPARADOXのための小ブロックと新たに呼び出されたCOMMAND.COMのためのブロックが1個，計496バイトがとられている。これはUMBをDOSの管理下に組み込んださささやかな効果である。その結果PARADOXを終了することなく，入力するデータをエディタを使って編集したり，Lotus123にデータを送り込んで計算処理をしてからすぐに取り込むなどということが可能である。ところがPARADOXは，EMSの空き領域を全て占有してしまうので，後から呼び出されるプログラムはEMSを利用できないという問題はある(22)。この問題はMS-Windows3.0を利用することにより解決できることを第Ⅳ節で示す。

Ⅲ　ハードディスクの最適化

MS-DOSのファイルシステムは，もともとがフロッピーディスク用に作成されたものであるため，大容量のハードディスクには対応しきれないところがある。マイクロソフト社のDOS 5は初めてこれを本格的にサポートしたが，それを利用できないユーザーは既存のシステムを何とか工夫しながら使う必要がある。ここではMS-DOSのハードディスクファイルシステムの問題点とそれを回避する方法をいくつか紹介する。

1. ブート装置と起動メニュー

システムをハードディスクから起動すると，いつも起動メニューが表示され

(21)　ボーランド社のリレーショナルデータベース管理ソフト

(22)　それならPARADOXを起動する前にEMSを適当に確保しておいて，PARADOXを抜けてからそれを解放してやればうまくいくのではないかと考えられる。実際にPARADOXとLotus123で試してみたところ，Lotusでは確かにEMSを利用できたが，PARADOXへ戻るところでハングアップしてしまう。

るという問題がある。ハードディスクをNECのFORMAT.EXEで初期化した場合，複数のパーティションからブート可能に設定しておくと，あるパーティションを自動起動に設定しただけでは，本当に自動的には立ち上がってくれない。メモリスイッチでBOOT装置をそのハードディスクに設定しておく必要がある。

以前はハートディスクに唯1つのブート可能なパーティションしかない場合でもメモリスイッチの変更が必要であったが，3.3Cではパーティションが1つの場合はBOOT装置が標準のままでも自動的に起動するように改善されている。したがって，ここではハードディスクに複数のシステムがインストールされている場合を考える。

何か特殊なプログラムがあって，そのフロッピーディスクから立ち上げなければならないようなことがある。自動起動に設定してしまうと，こういう時にいちいちメモリスイッチを書き換えてからリセットしなければならない。また何らかの理由でハードディスクから起動できなくなったときは，ディップスイッチを操作してメモリスイッチを全て出荷時の設定に戻さなければシステムを起動できなくなる。

この問題は，ICM社のEXFORM.EXEで初期化することにより解決する。その他に，この問題だけを解決するツールもある。(23)

2. パーティションの大きさ

パーティションの大きさにより論理セクタサイズが決定される。IBM-PCでは論理セクタという概念はないので，セクタサイズと言えば物理セクタサイズのことであり，これは512バイトに固定されている。NECでは，物理セクタをいくつかまとめて論理セクタとして扱うことができる。DOSではセクタ番号を16ビットで表すことにしているため，1つのドライブは，2の16乗（65536）までしかセクタをもつことができない。したがって1セクタを1KBにとるなら，1ドライブの容量は64MBが限度となる。それを越えると128MBまで1セ

(23) IPLを変更するプログラム。作者 Ackune，フリーウェア

クタは2KBになる。

このセクタサイズは主記憶の消費にきいてくる。DOSの入出力バッファは，バッファ1個につき，セクタ長＋16バイトのメモリを必要とする。したがって，CONFIG.SYSでBUFFERS＝20としてあれば，パーティションが64MB以下なら20800バイト，65MB以上なら44200バイトのバッファ領域が必要になる。

それなら，1パーティションは64MB以下にしておく方がよいかというと，ことはそう簡単ではない（表1）。表中の比較項目のうち，領域の有効利用については次節で説明する。

アクセス速度の比較には微妙な問題があるが，バッファ領域を大きくとることが可能であるならば，一度のアクセスで2KB読み書きする方が効率がよくなると考えられる。パーティションの大きさを64MBで抑えた場合には，ドライブ数が増えてしまうという欠点がある。あるファイルを探すとき，あちこちのドライブを訪ねて回るなどということもあり得る。そういう時にはJOINコマンドを利用して見かけ上のドライブ数を抑えることもできるが，直接ドライブ名を指定させるプログラムもあるので面倒な面もある。

しかし，それら以外にも問題はある。MS-DOSはファイルを更新するとき，元のファイルの領域に上書きするのではなく，新しい空き領域に書く。これは上書きする前にそのファイルを消去してから書いても同じである。フロッピーディスクの場合は，一旦メディアを取り出してからもう一度セットすることで直前にできた空き領域を認識してくれるようになるが，ハードディスクやRAMディスクのように取り外しのできないメディアのファイルは，こうして書き換えるたびにその位置が移動することになる。こうなると論理的に関連の深いファイルでも，実際には遠く離れて配置されることが起こる。1つのパー

表1　パーティションの大きさによる比較

	64MB	65MB以上
バッファ数	大きくとれる	大きくとれない
領域の有効利用	有利	不利
アクセス速度	やや不利	やや有利
ドライブ数が増える	不利	有利

ティションが大きければそれだけファイルが遠く離れてしまう可能性が高くなることは明かであろう。そのために時々ディスク内のファイルの配置を最適化してやる必要がでてくるが，パーティションが大きければ，最適化に要する時間も長くなるであろう。

さらにパーティションが大きければバックアップとリストアが大変である。パーティションがどうであろうと，全体としてのバックアップ時間は変わらないが，大容量を1度に全部バックアップするのは大変である。小さく分割してあれば1回の処理時間は短くてすむ。

これらのことを総合して考えれば，通常のユーザーは1パーティションを64MB以下にしておく方がよいと思われる。

3. クラスタサイズ

セクタはMS-DOSがディスクをアクセスする単位であるが，ディスクの空き領域の管理はクラスタが単位である。クラスタサイズの問題は先の論文でも触れたが，その後かなり状況が変化しているのでここで再び取り上げる。

ハードディスクのインタフェースにはSASIとSCSIの2つがある。従来はSASIが使われていたが，大容量のものはSCSIが使用されている。ディスクのフォーマットについても標準フォーマットと拡張フォーマットの2つがあるが，前者は20MBまでのディスクしか扱えないのでここでは考えない。

まず，SCSIの場合であるが，表2より64MBの領域まではクラスタサイズが2KBとなるので，余り問題はない。SASIの場合40MBの領域ではクラスタサイズが16KBになる。これでは利用効率は悪くなる。

SCSIのときは1クラスタ2KBでフォーマットできるのに，SASIではそれができないのは何か理由があるのかと思うが，実は特に理由はなく，ただ従来そうしていたからということのようである。実際NECのFORMATプログラムをほんの少し変更すればSASIのハードディスクをSCSIのようにフォーマットできる。

クラスタサイズが小さければ，ディスクスペースの利用効率は高くなるが，FATのサイズが大きくなるため，FATに対するI/Oが増加して全体の効率を

表2　NECのFORMAT. EXEで初期化されたハードディスクの諸元[8]
拡張フォーマットの場合

SASIの場合

領域指定（KB）	5	10	15	20	25	30	35	40
クラスタサイズ	2	4	4	8	8	8	16	16
ディレクトリ数	512	768	1024	1280	1536	1792	2560	3072

全て12ビットFAT，セクタ長は1KB

SCSIの場合

領域指定(KB)	5	10	15	20	25	30	35	40	64	128
クラスタサイズ	2	4	2	2	2	2	2	2	2	4
ディレクトリ数	512	768	1024	1280	1536	1792	2560	3072	3072	3072

10MBまでは12ビットFAT，それを越えると16ビットFAT
セクタ長は64MBまでは1KB，それを越えると2KB

下げる可能性がある。手持ちのハードディスクで簡単なテストをした結果では，バッファ数が極端に少ないときには5％程度の差がでたが，バッファ数20以上ではほとんど差が出なかった。したがって，クラスタのサイズは2KBにする。

以下に筆者のハードディスクの全ファイルのサイズについて調べた結果を示しておく（表3～表5）。最小が0になっているのは，プログラムが異常終了したときにできたもので，MS-Windowsの作業ファイルが3個，その他が1個あった。最大は花子のアウトラインフォントファイルであった。ファイルサイズの分布図（図1）は縦軸がファイルの分布密度で，サイズが1KB変化する区間に何個のファイルがあるかを表している。横軸は対数メモリになっている。

4. まとめ

ここまでハードディスクの抱える問題点をいくつか考えてきた。大容量ディスクの基本的問題の解決にはDOS5を使わなければならないけれども，いくつかはICMのハードディスク拡張フォーマットプログラムEXFORM. EXE[24]を使うことで改善できる。

表3　ファイルサイズ基本統計

ファイル数	2867　個
サイズの合計	85633KB
最小	0
最大	2946
平均	29.9
メディアン	5.9

(24)　EXFORM. EXEは未だにパーティションサイズが64MBのとき，セクタが2KBになるというバグがある。

表4 ファイルサイズの度数分布

階級(KB)	ファイル数	累積数	累積割合
1 以下	565	565	19.7%
2	329	894	31.2%
4	347	1241	43.3%
8	360	1601	55.8%
16	383	1984	69.2%
32	344	2328	81.2%
64	275	2603	90.8%
128	133	2736	95.4%
129 以上	131	2867	100.0%

表5 クラスターサイズと必要容量

クラスターサイズ	1KB	2KB	4KB	8KB	16KB
必要容量（KB）	87073	88664	92260	100048	117120
無駄になる容量	1440	3031	6627	14415	31487
利用効率（%）	98.3	96.6	92.8	85.6	73.1

(1) メモリスイッチでBOOT装置を標準にしたままでハードディスクから自動立ち上げできる。シフトキーを押しながら立ち上げれば起動メニューが表示される。

(2) クラスタサイズを選択できる。これはSASIのハードディスクでもクラスタサイズを小さくできる。

(3) ルートディレクトリのためのセクタ数を指定できる。

(3)については今まで説明していないが，表2にあるように40MB以上の領域のときは，ルートディレクトリのスロット数が3072になっている。通常はこんなに多くは使わないので減らすことができる。

IV MS-Windows3.0

MS-DOSの操作環境を改善するためにMS-Windows3.0を利用することを提案する。マニュアルには，Windowsアプリケーションだけを実行するときは，スタンダードモードの方が速いと書いてある[8]が，本論はDOSの利用環境を論ずるのが目的であるので，エンハンスドモードについてのみ言及する。すなわ

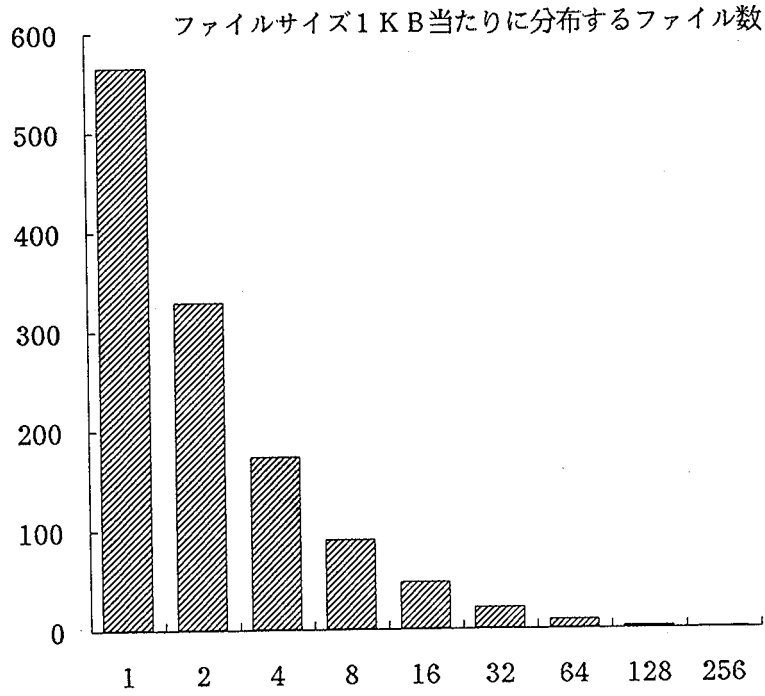
ち，エンハンスドモードの下でDOSのアプリケーションプログラムを使用することにより，従来とは異なる新しい操作環境が得られることを紹介したいのである。

1. 386エンハンストモードにする

説明書[9]によれば，80386以上のCPUをもつマシンの場合，ノーマルレゾリューションモードでは3072KB以上，ハイレゾリューションモードでは4096KB以上のXMSメモリが必要であると書いてある一方で，XMSが1024KB以上あれば，ウィンドウ起動時のオプションにより，エンハンスドモードにすることができるとなっている。実際にはどうなのか確かめてみた。

まずWindowsをエンハンスドモードを指定して立ち上げる場合は（/3のオ

図1　ファイルサイズの分布



プションをつけて立ち上げる)，XMSが1024KBあれば確かにエンハンストモードにはなるが，一太郎，Lotusどちらも起動できなかったので問題にならない。2048KBの時は一応一太郎，Lotusともに起動できたが，EMSを使うLotusはシートの読み込み時間がかなり遅くなってしまう（筆者のシステムでは約5倍)。XMSメモリが3MBのときは2倍程度で収まるので，これはメモリ不足が原因と思われる。また何かの拍子にシステムが停止してしまうこともある。そういう訳で/3で強制的にエンハンスドモードにするのは勧められない。

次にWindowsシステムにモードを選ばせる方で試してみた。説明書にはfilesの値は30以上が必要であること，buffersの値はSmart Driveを使用した場合は10に設定するように指示されているので，リスト13のようなCONFIG. SYSを用意し，中のMINの値をいくつか変えて起動させてみた。2048は標準サイズの指定であるが，これはWindowsのときは自動的に調整されるので意味がない。

その結果，MINの値が768では確かにエンハンスドモードになるが，896ではスタンダードモードになる。システム側では，エンハンスドモードにするためには，Smart Driveの制御領域を含めて最低2.3MB程度のXMSメモリーが必要と判断しているようである。

device＝￥windows￥smartdrv. sys 1024 256

あたりに設定しておく方がよいようだ。

ハイレゾリューションモードではXMSメモリが3MBではMINを0にしても確かにエンハンソドモードにはならない。実際には少なくともあと256KB程度必要であった。さらにSmart Driveを使ってスピードアップを図るためには，説明書どおり4MBのXMSメモリが必要ということになる。

2. プログラムスイッチャーとしてのWindows3.0

386エンハンスドモードを利用すると，複数のウインドウプログラムを同時に起動させておき，それらを切り替えながら使用できることはもちろんであるが，非ウインドウアプリケーションも同時に複数個使用できる。これは，80386の仮想86モードを利用しているからである。この機能は以前のWindows/386においてもある程度は実現されていたが，3.0ではさらに強化された。

(1)　基本メモリによる制約が事実上なくなった。

(2)　仮想86マシンごとに，仮想のEMSメモリが割り当てられるようになった。

(1)に関しては，以前はウインドウプログラムであろうとなかろうと，ひとつのプログラムが起動されれば必ずなにがしかの基本メモリーが消費されていたので，同時に起動できるプログラムの数は結局基本メモリの大きさに制約されていたが，3.0ではこの制約が，スワップファイルの大きさにまで拡大された。また，Windows/386では，EMSの領域は全体として固定されていたため，一太郎4.3やPARADOXのように，使用可能なEMSメモリーを全て占有してしまうアプリケーションを先に起動してしまうと，後から起動したアプリケーションはもはやEMSを利用することができなかった。(2)により個々のプログラムごとに，実装メモリーを越えて任意のサイズのEMSメモリを割り付けておくことができるようになり，この問題が解決された。以下に具体的な使用例とセットアップの手順を紹介しておく。

例として，一太郎4.3，Lotus123，PARADOX，COMMAND.COMを同時に使えるような環境を作成してみよう。これら4つのプログラムを同時に起動しておき，NoWinAppという名前でグループとしてプログラムマネージャに登録しておく。それぞれのプログラムは日本語FEP，マウス，プリンタ等のドライバーが必要であるから，単に目的のプログラムを起動するだけではいけないので，以下の手順が必要である。

(1)　DOSプログラムを起動するためのバッチファイルを作成する。

(2)　Windowsの中での実行のさせ方を指定する＊.PIFファイルを作成する。

(3)　プログラムマネージャに登録する。

例としてLotus123用のバッチファイル（リスト14）とPIFファイルの1部を示しておく（図2，図3）。バッチファイルでは，プロセスが終了したときには当該の仮想86マシンが消えてしまうので，後始末はしなくても良い。PIFファイルでは使用したいEMSのサイズを指定する（図3）。

図2 Lotus123のためのPIFファイル1

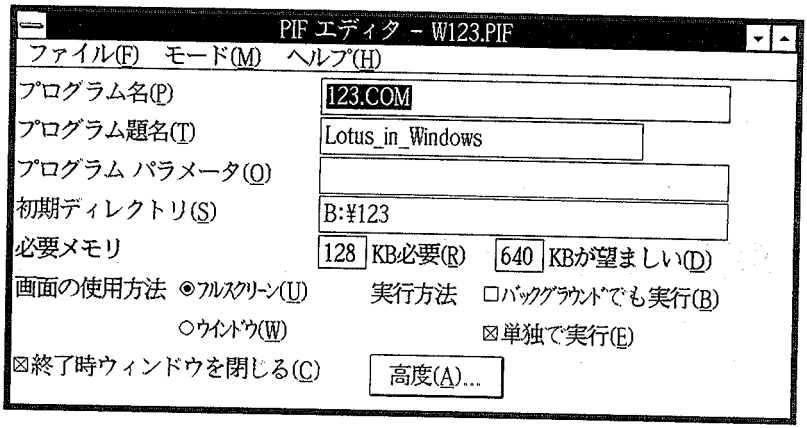
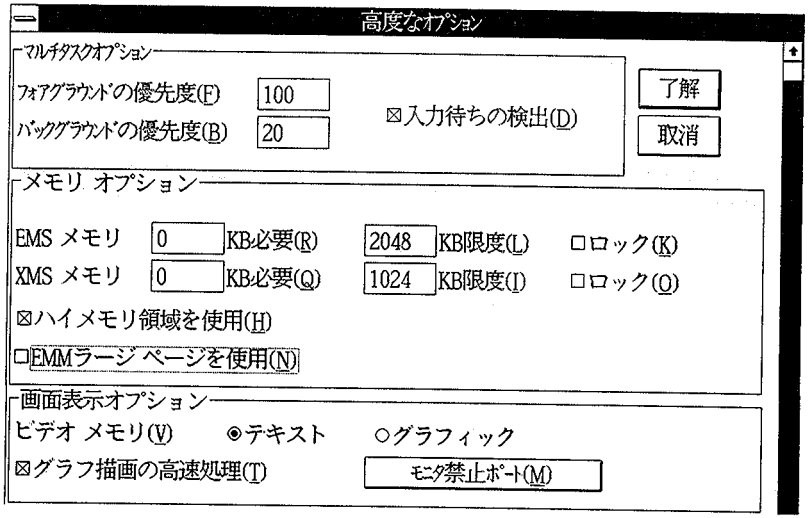


図3 Lotus123のためのPIFファイル2



手順(3)はアプリケーションをプログラムマネージャまたはファイルマネージャから直接PIFファイルを起動させることにすれば，必ずしも必要ではないが，それぞれに好みのアイコンを割り付けてデスクトップに表示しておき，マ

ウスで選択することによって起動できるようにするために行うものである。プログラムを登録する画面を図4に，アイコンを登録している画面を図5に示しておく。最後に出来上がったデスクトップ画面が図6である。なお，アイコンはWindows付属のペイントブラシとシェアウエアIcon Manager[25]を使用した。

図4　プログラムの登録

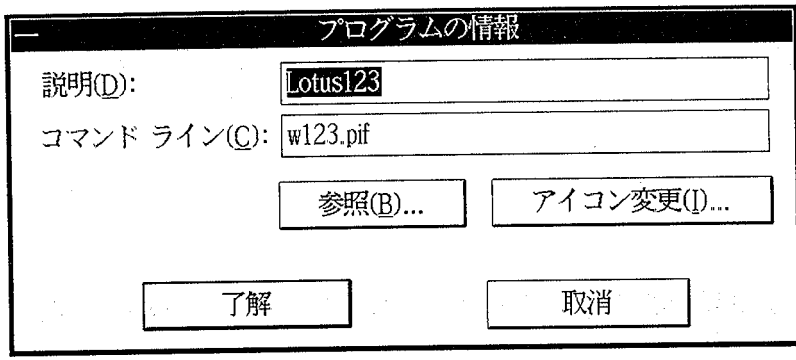


図5　アイコンの登録

Icon Managerには多くのアイコンが付録としてついているので気に入ったものがあればそれをそのまま使用できる。

(25)　Windows3.0のアイコンを管理するユーティリティプログラム。作者　Impact Software（米国），シェアウェア

実際にこれら4つのDOSアプリケーションを起動させてみると，割り付けられているEMSだけで8MBになった（表6）。各アプリケーションにはPIFファイルで指定された大きさ＋512KBのEMSメモリが割り付けられていた。

図6　プログラム登録後のデスクトップ

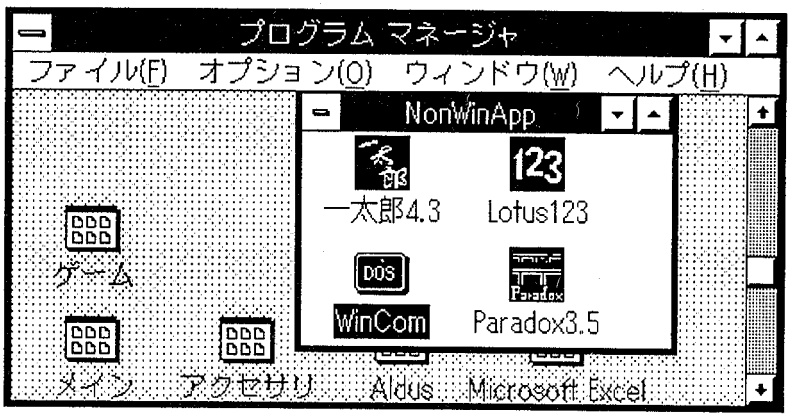


表6　EMSの使用量

プログラム	EMS (MB)
COMMAND.COM	1.5
LOTUS123	2.5
PARADOX	2.5
一太郎	1.5
合計	8.0

3. Windows3.0の実行速度

Windowsの下で実行されるDOSアプリケーションの実行速度を測定したものが表7である。SysCallというのは，MS-DOSの簡単なシステムコールの呼び出し，DhryはDhrystoneベンチマーク，NortonはNortonユーティリティのSIによる比較であり，通常の仮想86モードでの速度を1として表した。fullというのはWindowsのフルスクリーンモード，winはアプリケーションをウインドウ

表7　WINDOWS3.0の実行速度

モード	V86	full	win
SysCall	1.0	0.67	0.39
Dhry	1.0	0.92	0.58
Norton	1.0	0.93	0.46

の中で実行するモードである。Windowsではシステムコールがかなり遅くなっているのがわかる。

次にEMSの基本性能を測定してみた（表8）。Windowsではバックグランドプロセスの影響で測定値にばらつきがでるので，Windowsに関する測定値は10回の測定の平均値を用いた。メモリアクセスを除くEMM機能の速度は，フルスクリーンの場合でさえも，最も速いMEMORY-PROの5倍から10倍遅くなっているが，これはWindowsのEMMがもともとMEMORY-PROに較べて約2倍遅いことを考慮する必要がある。

表8　EMSのベンチマーク

	PC-98XL 2					PC-98VM 2　IODATA		
	PRO386	NECEMM	WINEMM	FULL	WIN	/NA	/A 1	/A 2
(1)	209	350	400	1905	3705	727	494	805
(2)	110	233	229	1073	2110	455	370	526
(3)	133	217	278	1083	2069	714	481	784
(4)	202	312	362	1153	2263	506	536	596
(5)	1730	1720	1720	1969	3767	3970	3960	3960

(1)　論理ページの物理ページへの割当（μ秒）
(2)　ページマップのセーブ＆リストア（μ秒）
(3)　複数の論理／物理ページの割当　（μ秒）
(4)　部分ページマップの取得／設定　（μ秒）
(5)　EMSメモリアクセス速度（65536*100バイト）（m秒）

PRO386　MEMORY－PRO386verl 50
NECEMM　NECのMS－DOS3.3C付属のEMM386.SYS
WINEMM　NECのWINDOWS3.0付属のEMM386.SYS
FULL　WINDOWS3.0のフルスクリーンモード
WIN　WINDOWS3.0のウインドウモード
/NA　アイオーデータEMM4J.SYS /NA
/A 1　アイオーデータEMM4J.SYS /A 1
/A 2　アイオーデータEMM4J.SYS /A 2

これらのEMS基本性能の差が実際のアプリケーションでどの程度でてくるのかを測定したものが表9である。EMSの影響が出易いものとしてLotus123を用いた。テストはまずA列全体（A1～A8192）に整数データを埋め込んだ後，それをB列からI列までコピーする時間と，それを消去する時間を測定した。整数データを用いたのは，Lotusの拡張メモリマネージャを内部＋拡張にしたときには，整数データはEMSを使用しないのでEMSの影響が推定しやすいからである。Dhrystone値の減少はある程度納得できる値であるが，Lotusのコピー時間の速度低下は大きすぎるように思われる。クロック10MHzの9801VMよりも遅くなっている。

このようにWindows3.0では，EMSの速度が遅すぎる欠点はあるものの，DOSアプリケーションはほぼ完全に同時起動可能である。エディタでデータを編集，修正作業をしながらLotusやPARADOXに読み込ませるとか，PARADOXで必要なデータの検索・抽出を行い，それをLotusで計算してからまたPARADOXに戻すことなどが可能になった。また，クリップボードを利用してアプリケーション間でデータの転送が可能になるなど，DOSの操作環境改善という面では評価できる。

表9　Lotus1-2-3のベンチマークテスト

	PC-98XL 2			Windows		PC-9801VM 2		
	内部	PRO386	EMM386	内部	拡張	内部	/A1	/A2
コピー	11.7	27.4	34.1	12.0	77.6	35.0	72.0	83.0
消去	5.1	6.8	6.8	5.3	7.1	14.0	19.3	19.3

単位　秒
コピー　A列全体（A1～A8192）をB～Iまでコピー
消去　B1～I8192までを消去
内部　内部＋拡張メモリ
拡張　拡張メモリのみ

V　むすび

MS-DOSマシンをNEC版MS-DOS3.3C用に最適化するための試みをいくつ

か紹介した。実行速度をあまり犠牲にすることなく，基本メモリを大きく空けることには一定の成果が得られた。その結果，従来よりメモリに常駐するデバイスドライバやTSRプログラムを利用しやすくなるであろう。また，ひとつのアプリケーションを起動したままでも，エディタなどの小型のプログラムは十分に使用できる環境が得られた。

Windows3.0の386エンハンスドモードは複数の非ウインドウアプリケーションを同時に起動し，それを切り換えながら使用できる環境を与えてくれることがわかったが，EMSの速度低下が大きいことがわかった。

本論で検討した問題はMS-DOS5.0が使用可能になればある程度解決されるはずであるが，それまではここで述べた技法が役に立つものと信ずる。またDOS5.0の恩恵を受けられない古い機種については，ここで紹介した方法が大切なものとなるであろう。

参考文献

［1］ 中村邦彦，「MS-DOSのバッチプログラムの技法」，香川大学経済論叢第62巻2号，1988年

［2］ 藤田憲治，「MS-DOSの生きる道」，日経バイト，№87，1991年

［3］ 内田暁，「EMSのベンチマークテスト」，TheBASIC，№91，1990年12月号

［4］ 中島信行，「MS-DOSメモリ管理ソフト技法」，CQ出版社，1990年，199～200ページ

［5］ 「パソコン周辺機器・関連製品総覧」，日経バイト№88 特別増刊号，1991年7月

［6］ 山崎福馬，「MS-DOSの道具箱」，JICC出版局，1989年，188ページ

［7］ アイ・オー・データ機器，「IOS-10STDリファレンスガイド」，1990年，30ページ

［8］ NEC，「MS-DOS3.3C ユーザーズリファレンスマニュアル」，1990年

［9］ NEC，「MS-WINDOWS3.0ユーザーズリファレンスマニュアル」，1991年

付　録

リスト1　仮想86モードにするconfig.sys

```
buffers=1
files=20
device = ¥sys¥vems.drv /e /u /b10 /x
device = ¥sys¥lu.drv ¥sys¥mx1star5.drv /v /b10
device = ¥sys¥lu.drv ¥sys¥1135d.sys
device = ¥sys¥lu.drv ¥sys¥print.sys /u
shell = ¥command.com  /p /e:640
```

リスト2　リスト1のconfig.sysのときに使用するautoexec.bat

```
echo off
set sd=b:
set td=b:
¥etc¥envdrv
if not "%rd1%"=="" set rd=%rd1%
if not "%rd%"=="" set td=%rd%
set tmp=%td%¥
for %%i in (hd1 hd2 hd3 hd4 fd1 fd2 fd3 fd4 rd1) do set %%i=
prompt $e[36m$l$p$g$e[33m
path %tmp%;%sd%¥usr;%sd%¥bin;%sd%¥etc;%sd%¥nor
lkumb /l
addbuf 30
load autoland
load crtshut 180,180
load ezkey -b3 -c7 -¥+
set comspec=%sd%¥command.com
set vzdef=%sd%¥usr¥vz¥
set vzbak=¥vz_bak
break on
```

リスト3　リスト1とリスト2を使用したときのメモリマップ

```
addr PSP  blks   size  owner/parameters
---- ---- ---- ------  ------------------------
1007 0008    1   3696  config
10EF 10EF    2   4144  shell
11F4         1 581808  <free>
A000 FFFF    1 196608  {VRAM}
D001 FF02    3   8016  {config}
D1F9 D204    2   1184  autoland
D245         1    160  <free>
D250 D250    1    496  CRT SHUTter ver2.1 (C)N!S
```

```
D270 D270    1     1680  ezkey -b3 -c7 -¥+
D2DA FFFF    1    30176  {buffers*29}
DA39         1    23648  <free>
handle page    size  name     (EMS frame:C000h)
------ ---- ------  --------
  free  116  1856k
 total  116  1856k
```

リスト4　何も組み込まないconfig.sys

```
buffers=20
files=20
shell=¥command.com /p
```

リスト5　リスト5を使用したときのメモリマップ

```
addr PSP  blks   size  owner/parameters
---- ---- ----- ------  ------------------------
1009 0008    1   22688  config
1594 1594    3    3728  shell
1680         1  563184  <free>
```

リスト6　PC−9801VMのためのCONFIG.SYS

```
files=20
buffers=1
device=¥sys¥emm4j.sys /i /w=c0,c4,c8,cc,d0,d8,dc /a2 /b
device=¥sys¥lumb.sys /m ¥sys¥ramdsk.sys /0 /0 /B
device=¥sys¥lumb.sys /i /m ¥sys¥print.sys /u
shell=¥command.com /p /e:512
```

リスト7　リスト6のためのAUTOEXEC.BAT

```
ECHO OFF
¥etc¥envdrv
set sd=a:
set td=a:
set rd=%rd1%
if not "%hd1%"=="" set sd=%hd1%
if not "%rd%"=="" set td=%rd%
for %%p in (hd1 fd1 fd2 rd1) do set %%p=
path %td%¥;%sd%¥usr;%sd%¥bin;%sd%¥etc
set prompt=$l$p$g
set comspec=%sd%¥command.com
set vzdef=%sd%¥usr¥
```

```
set vzbak=¥vz_bak
set tmp=%td%¥
break on
lumb ¥etc¥autoland.com
lumb ¥etc¥crtshut 180,120
lumb ¥etc¥ezkey -b2
bex /a99
bex /a7
lkumb /l
```

リスト8　リスト6とリスト7を使用したときのメモリマップ

```
addr PSP  blks   size  owner/parameters
---- ---- ---- ------  ------------------------
0FFF 0008    1    3296  config
10CE 10CE    2    4016  shell
11CB         1  582464  <free>
A000 FFFF    1  196608  {VRAM}
D001 D001    1    1152  {RAMDISK}
D04A D04A    1    5152  {PRINT}
D18D D801    1      80  autoland (env)
D193 D192    1    7280  {buffer*7}
D35B         1    2624  <free>
D400 FFFF    1   16384  {Not Avairable}
D801 D801    1    1024  autoland
D842 D842    1     496  CRT SHUTter ver2.1 (C)N!S
D862 D862    1    1680  ezkey -b2
D8CC D8CB    1   12480  {buffer*12}
DBD9         1     608  <free>
handle page   size  name    (EMS frame:C000h)
------ ---- ------  --------
     1    2    32k  EMMXXXX0
     2    2    32k  UMB
  free  124  1984k
 total  126  2016k
```

リスト9　EXT_TPAのためのCONFIG.SYS

```
files=20
buffers=1
device=¥sys¥emm4j.sys /i /w=c0,c4,c8,cc,d0,d4,d8,dc /a2
device=¥sys¥ramdsk.sys /0 /0 /B
shell=¥command.com /p /e:512
```

リスト10　リスト９のためのAUTOEXEC.BAT

```
ECHO OFF
¥etc¥envdrv
set sd=a:
set td=a:
set rd=%rd1%
if not "%hd1%"=="" set sd=%hd1%
if not "%rd%"=="" set td=%rd%
for %%p in (hd1 fd1 fd2 rd1) do set %%p=
path %td%¥;%sd%¥usr;%sd%¥bin;%sd%¥etc
set prompt=$l$p$g
set comspec=%sd%¥command.com
set vzdef=%sd%¥usr¥
set vzbak=¥vz_bak
set tmp=%td%¥
ext_tpa
addbuf 20
loaddev "¥sys¥print.sys /u"
load ¥etc¥autoland
load ¥etc¥crtshut 180,120
load ¥etc¥ezkey -b2
break on
```

リスト11　リスト９とリスト10を使用したときのメモリマップ

```
addr PSP  blks   size  owner/parameters
---- ---- ---- ------  ------------------------
0FFF 0008    1    4288  config
110C 110C    2    4016  shell
1209         1  581472  <free>
A000 FFFF    1  196608  {VRAM}
D001 D001    1    5152  {PRINT}
D144 D144    1    1024  autoland
D185 D185    1     496  CRT SHUTter ver2.1 (C)N!S
D1A5 D1A5    1    1680  ezkey -b2
D20F FFFF    1   19776  {buffer*19}
D6E4         1     416  <free>
D700 FFFF    1    4096  {SASI}
D801 D144    1     160  autoland (env)
D80C         1   16192  <free>
handle page   size  name    (EMS frame:C000h)
------ ---- ------  --------
     1    2    32k  EMMXXXX0
     2    3    48k  EXT_TPA
  free  123  1968k
```

```
 total  126  2016k
```

リスト12 Paradox からDOS に抜けたときのメモリマップ

```
addr PSP  blks   size  owner/parameters
---- ---- ---- ------  ------------------------
1007 0008    1   3696  config
10EF 10EF    2   4144  shell
11F4 11F4    1  12688  AS$VJE
150E 150E    2  92000  vz -z
2B86 2B86    1   1120  paradoxr
2BCD 2BCD    2   3744  command
2CB9         1 471888  <free>
9FEF 2B86    1    256  paradoxr (env)
A000 FFFF    1 196608  {VRAM}
D001 FF02    3   8016
D1F9 FFFF    1  30176
D958 D963    2   1184  autoland
D9A4         1    160  <free>
D9AF D9AF    1    496  CRT SHUTter ver2.1 (C)N!S
D9CF D9CF    1   1680  ezkey -b3 -c7 -¥+
DA39 10EF    1     48  shell
DA3D 2B86    1    240  paradoxr
DA4D 2BCD    1    256  command (env)
DA5E         1  23056  <free>

handle page   size  name    (EMS frame:C000h)
------ ---- ------  --------
     1    4   64k  VJE-MAIN
     2    4   64k  VJE-LIB
     3   20  320k  VZ
     4    4   64k
     5   84 1344k
  free    0    0k
 total  116 1856k
```

リスト13 WINDOW3.0 のためのCONFIG.SYS

```
buffers = 10
files = 30
device = ¥windows¥himem.sys
device = ¥windows¥smartdrv.sys 2048 min
shell = ¥command.com a:¥ /p /e:640
```

リスト13　WindowsからLotus123を起動するバッチプログラム

```
echo off
echo device=%sd%¥sys¥mouse.sys >%tmp%dev.$$$
echo device=%sd%¥sys¥atok7a.sys /d=%sd%¥atok7l.dic
  /g=%sd%¥jsw¥jfgaij.ufo /e=1 /s=1 /t=0000>>%tmp%dev.$$$
echo device=%sd%¥sys¥atok7b.sys >>%tmp%dev.$$$
adddrv %tmp%dev.$$$
del %tmp%dev.$$$
%sd%
7to6
cd ¥123
123
```