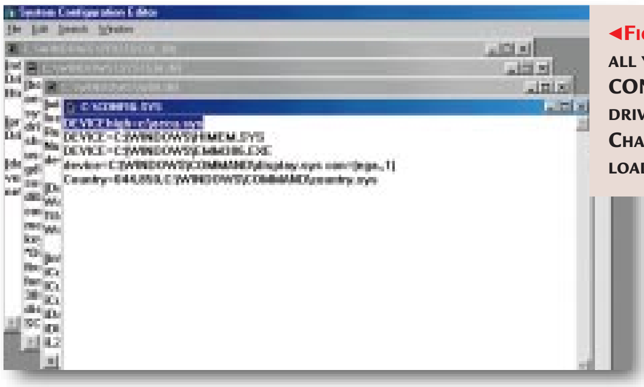
Memory master-class

Roger Gann gives you some memory management super-hints.



System Configuration Editor

C:\CONFIG.SYS

```
[illegible]
DEVICE=C:\WINDOWS\HIMEM.SYS
DEVICE=C:\WINDOWS\EMM386.EXE
device=C:\WINDOWS\COMMAND\display.sys con=(ega,,1)
Country=044,850,C:\WINDOWS\COMMAND\country.sys
```

◀FIG 1 SYSEDIT IS A GREAT TOOL FOR EDITING ALL YOUR STARTUP FILES IN ONE HIT. HERE, THE CONFIG.SYS FILE IS LOADING THE DEVICE DRIVERS 'LOW' INTO CONVENTIONAL MEMORY. CHANGING 'DEVICE=' TO 'DEVICEHIGH=' LOADS THEM INTO UPPER MEMORY

The memory-map which all x86-based PCs employ is based on the original 1981 IBM PC. Driven by an Intel 8088 CPU it could theoretically address 1Mb of RAM. But in practice, only about two-thirds of this (640Kb) is available for use. This is because IBM erred on the side of caution and allocated a large chunk (384Kb) of that memory address space for system ROMs: things like the system BIOS, the display BIOS and the hard disk BIOS. IBM over-anticipated ROM usage of this space and on a typical, modern PC only half, 192Kb or less, is used.

The problem is that because IBM specified the address of the first ROM at 640Kb, the 192Kb of available memory space above this is wasted and cannot be 'seen' by MS-DOS. These chunks of memory are called Upper Memory Blocks.

With a modern PC, it's possible to access all this lost 'Upper Memory', above 640Kb, to load device drivers and other memory-resident programs there. This reduces the hit on conventional memory. It's also possible to load parts of DOS into the 64Kb High Memory Area (HMA) above 1,024Kb, further freeing-up conventional memory. With all these tweaks you can get as much as 627Kb of free conventional memory, which should be enough to satisfy the most awkward game. But how do you achieve this?

The first step to maximising DOS memory is to use the latest version. The best DOSes in this regard are IBM PC DOS 7.0 and Caldera DR-DOS 7.03. MS-DOS 6.2x isn't bad but the first two take memory management, and many other areas of DOS, a step further. All three have an Expanded Memory manager and an Extended Memory manager and a way of optimising your memory usage. In the case of MS-DOS 6.2, these are EMM386 .EXE, HIMEM.SYS and MEMMAKER.EXE. The next step is to audit your startup files and install memory management.

Audit your startup files

Pass a critical eye over your CONFIG.SYS and AUTOEXEC.BAT files. It doesn't matter what sort of DOS or PC you have: it's good practice to take a long, hard look at them to see what fat can be trimmed; not even MemMaker can guess whether or not you need a particular driver or TSR.

Files and BUFFERS are candidates for the chop, especially as they can waste space in the HMA. Ask yourself: do you really need ANSI.SYS, or code page support?

Reduce your STACKS — you can save 2-4Kb of conventional RAM by adding this line to your CONFIG.SYS file. By default, DOS sets aside 2Kb of RAM for 'STACKS'. To turn it off explicitly, add a line like this:

```
STACKS=0,0
```

You may get an error message if you do

```
EXCEPTION ERROR 12
```

which comes from EMM386, or

```
INTERNAL STACK OVERFLOW
```

...in which case put things back the way they were. Note that Windows 3.1x inserts the line STACKS=9,256 during Setup, but doesn't actually need it for it's own operation.

Files and buffers are candidates for the chop

Check your environment size — look at the SHELL statement in your CONFIG.SYS. It might well look something like this:

```
SHELL=COMMAND.COM C:\DOS
 /E:1024 /P
```

The /E:1024 bit sets aside 1Kb for environment space, which might be a bit high. You can tell just how much you actually need by typing SET<CR> and counting all the characters it displays. An easier way is to redirect the environment contents to a file, like this:

```
SET > ENV <CR>
```

Then a simple DIR ENV<CR> will reveal the size of the file and hence your environment. If it's less than 1,024 use EDIT to amend the value in the SHELL statement.

Reduce your file control blocks — this is scraping the barrel but every little helps! Put this line in your CONFIG.SYS — it will save you a vital few bytes:

```
FCBS=1
```

Check your LASTDRIVE statement — more barrel scraping: DOS allocates 88 bytes for each logical drive you specify, so if you have a LASTDRIVE=Z statement you'll be using up 2.3Kb of memory. Amend the statement to the last logical drive you have got.

Tweak DoubleSpace — those with MS-DOS 6.0 and 6.2 will know that DBLSPACE.BIN, the DoubleSpace driver, is large, swallowing 43Kb (6.0) or 52Kb (6.2) of precious memory. It can load into the HMA if space and other HMA users such as BUFFERS, permit. So, keep your BUFFERS statement low, especially if you're using a disk cache like SmartDrive. You can save a further 5Kb by disabling the AutoMount feature. You can also use disk compressors such as Stacker and DoubleSpace to increase the apparent size of RAM disks, therefore preserving real memory.

Install memory management

Load HIMEM.SYS — if you have a 286 or better, you need to load HIMEM .SYS in your CONFIG.SYS, like this:

```
DEVICE=C:\DOS\HIMEM.SYS
```

HIMEM.SYS is an extended memory manager and, if you have more than one megabyte of RAM fitted, it will let you use the HMA, a 64Kb chunk of RAM that sits at 1,024Kb, which can be used by DOS. To do this, add this line to CONFIG.SYS:

```
DOS=HIGH
```

DOS will now load part of COMMAND .COM and BUFFERS in this portion of memory. Third-party programs can shift even more of DOS into HMA — as much as a further 10Kb.

☛ **Use a memory manager** — if you have a 386SX or better, you'll need to load a memory manager. MS-DOS 6.2x comes with EMM386.EXE which is a reasonable choice, considering that it's free, but if your needs are greater you'll have to buy a third-party memory manager like Quarterdeck's QEMM.

Note that EMM386.EXE is loaded in CONFIG.SYS after HIMEM.SYS:

```
DEVICE=C:\DOS\EMM386.EXE
```

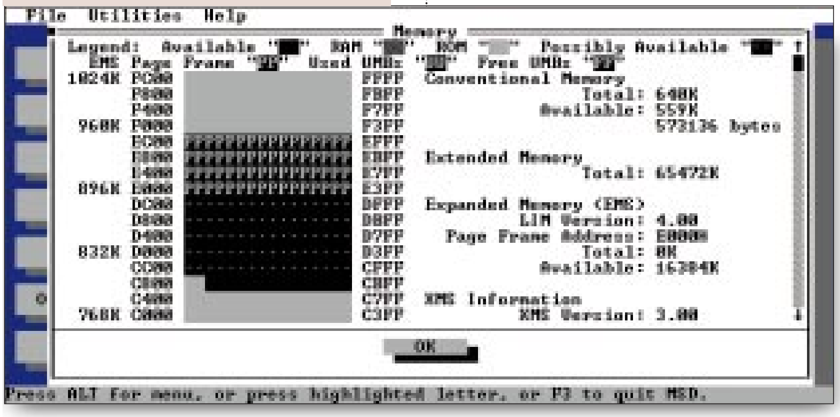
Once this is loaded, you can then access Upper Memory Blocks. To do this, UMB support has to be turned on in CONFIG.SYS, as follows:

```
DOS=UMB
```

You can now load device drivers and TSRs into UMBs using DEVICEHIGH= statements (rather than plain DEVICE=) in CONFIG.SYS and LOADHIGH (or LH) in AUTOEXEC.BAT [Fig 1].

So, using EDIT, amend each line of your startup files where appropriate. You probably won't be able to load everything 'up there' but if it can't fit, it will be loaded in conventional RAM.

▼Fig 2 For manual fine-tuning of your memory optimisation, MSD is a useful tool. Here, the E000 - EFFF block has been taken as an EMS page frame, while the UMB above CC00 is up for grabs



```
 File  Utilities  Help
                               Memory
 Legend:  Available "██"  RAM "██"  ROM "░░"  Possibly Available "██"
  EMS Page Frame "PP"  Used UMBs "██"  Free UMBs "██"
 1024K FC00                   FFFF  Conventional Memory
       F800                   FBFF                Total: 640K
       F400                   F7FF            Available: 559K
  960K F000                   F3FF                       573136 bytes
       EC00 PPPPPPPPPPPPPPPP  EFFF
       E800 PPPPPPPPPPPPPPPP  EBFF  Extended Memory
       E400 PPPPPPPPPPPPPPPP  E7FF                Total: 65472K
  896K E000 PPPPPPPPPPPPPPPP  E3FF
       DC00                   DFFF  Expanded Memory (EMS)
       D800                   DBFF          LIM Version: 4.00
       D400                   D7FF   Page Frame Address: E000H
  832K D000                   D3FF                Total: 0K
       CC00                   CFFF            Available: 16384K
       C800                   CBFF
       C400                   C7FF  XMS Information
  768K C000                   C3FF          XMS Version: 3.00

                                 OK

Press ALT for menu, or press highlighted letter, or F3 to quit MSD.
```

☛ **Tweak EMM386** — there are a number of tweaks you can use on EMM386 — and other memory managers, in principle — to save memory. These take the form of parameter 'switches' on the EMM386 command line.

For instance, the NOHI switch stops EMM386 from loading part of itself into UMB, saving 5Kb of UMB, which might be handy if you need a little more UMB room in order to load a big device driver such as DBLSPACE.BIN.

Another useful switch is NOEMS. This tells EMM386 that you want no expanded memory support whatsoever. This will free the 64Kb of UMB space that was being used as an EMS 'page frame'.

Some modern DOS programs can access Expanded Memory without resorting to a page frame, in which case you can add FRAME=NONE instead. Be warned, though: programs which require LIM v3.2 EMS won't like this option.

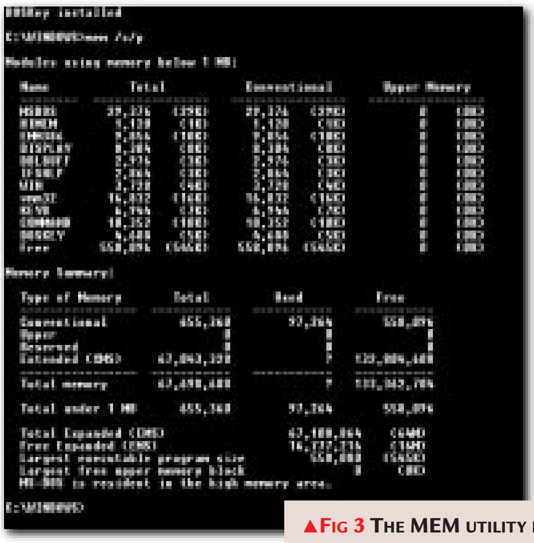
☛ **Use MemMaker** — while it is possible to manually load drivers into specific UMBs, it's not easy with tricky things such as having to edit SYSTEM.INI as well, and figuring out the loading order of files to make best use of the UMBs. Armed with something like MSD [Fig 2], MEM, pencil and paper, the IQ of Einstein and the rest of eternity, you could manually optimise it yourself — but life's too short. MS-DOS 6.2 made things a whole lot easier by including MemMaker which automated the whole process, making it as near painless as possible.

It has two modes: Express, which asks no questions at all, and Custom which allows you some control over the process. Without doubt, if you have MS-DOS 6.2x and need to rearrange your memory, and you want an easy life, then all you need to know is how to run MemMaker. Most users will find MemMaker sophisticated enough for their needs. It can look high and low for potential UMBs. It will, for instance, use the 32Kb of address space used by mono VGA displays, B000-B7FF. So, if you have colour VGA, this space is up for grabs. Often, Windows will take exception to this, in which case you need to add this line to the [386Enh] section of SYSTEM.INI:

```
DEVICE=MONOUMB.386
```

It will also try to use the bottom half of the system ROM space, F000-F7FF. This area of ROM is normally only used during bootup and is not accessed while DOS is running so it, too, is a prime candidate for inclusion.

Not all personal computers like this cheeky stunt but MemMaker will detect if it cannot use this area and will allow you to try again, with more conservative settings. MemMaker will not find every last bit of UMB — it ignores the range C000-C5FF, for instance — but you can probe further by using the MSD diagnostic program to examine the contents of RAM and then manually adjust EMM386.EXE's settings. You can check your progress in detail with the MEM utility [Fig 3].



```
DOSKey installed

C:\WINDOWS>mem /c/p

Modules using memory below 1 MB:

  Name           Total           Conventional      Upper Memory
  --------  ----------------  ----------------  ----------------
  MSDOS       29,376   (29K)    29,376   (29K)         0    (0K)
  HIMEM        1,120    (1K)     1,120    (1K)         0    (0K)
  EMM386       9,856   (10K)     9,856   (10K)         0    (0K)
  DISPLAY      8,304    (8K)     8,304    (8K)         0    (0K)
  DBLBUFF      2,976    (3K)     2,976    (3K)         0    (0K)
  IFSHLP       2,864    (3K)     2,864    (3K)         0    (0K)
  WIN          3,728    (4K)     3,728    (4K)         0    (0K)
  vmm32       16,032   (16K)    16,032   (16K)         0    (0K)
  KEYB         6,944    (7K)     6,944    (7K)         0    (0K)
  COMMAND     10,352   (10K)    10,352   (10K)         0    (0K)
  DOSKEY       5,440    (5K)     5,440    (5K)         0    (0K)
  Free       558,096  (545K)   558,096  (545K)         0    (0K)

Memory Summary:

  Type of Memory       Total         Used          Free
  ----------------  ------------  ------------  ------------
  Conventional          655,360        97,264       558,096
  Upper                       0             0             0
  Reserved                    0             0             0
  Extended (XMS)     67,043,328             ?   132,804,608
  ----------------  ------------  ------------  ------------
  Total memory       67,698,688             ?   133,362,704

  Total under 1 MB      655,360        97,264       558,096

  Total Expanded (EMS)                67,108,864   (64M)
  Free Expanded (EMS)                 16,777,216   (16M)
  Largest executable program size        558,080  (545K)
  Largest free upper memory block              0    (0K)
  MS-DOS is resident in the high memory area.

C:\WINDOWS>
```

▲Fig 3 The MEM utility is a useful tool for analysing memory use — MEM /C/P lists every program in memory and where it's located. In this example, nothing is loaded in Upper Memory

PCW CONTACTS

Send your comments about the 16-bit column to the PCW editorial office (address, p10) or email 16bit@pcw.co.uk