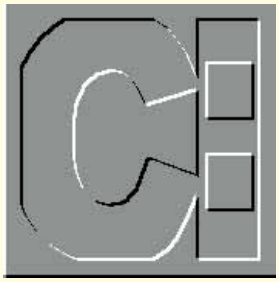


Revealing features

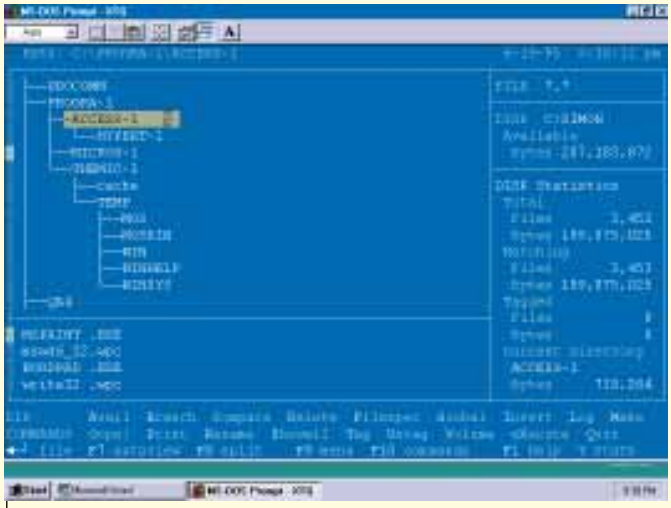
How is Windows 95 is going to handle DOS? Simon Collin explains. Plus, editing and re-booting tricks in pre-DOS 5, and hush... hear that silence?

Over the past few months there has been a mass of articles covering the forthcoming Windows 95 from Microsoft. All the reviews concentrate on the new user front-end, the networking and the long file names; none of them cover how Microsoft is going to handle DOS. Which is a shame, because the MSDOS section of Win95 is packed with new features.

Windows 95 is due to launch at the end of August, so if you're a DOS-aholic and are considering whether it's worth installing Win95, here are the main benefits and drawbacks of Win95 for DOS users.

The first thing you'll notice about the new product is that there's no underlying MSDOS. Instead, Win95 is the operating system, but it still retains support for MSDOS sessions and adds extra functionality. This all gets rather confusing when you first switch on your PC, since Win95 appears to load DOS and execute Config.sys and Autoexec.bat files. During this boot-up sequence, Win95 mimics normal DOS to allow any real-mode TSRs, drivers or other programs to load: these programs often look for DOS and Win95 is providing compatibility. Once they've been loaded, Win95 switches to protected mode and loads the rest of itself.

Now that Windows is up and running, it's time to forego that GUI desktop and get back to the DOS command line! Like Windows 3.1, MSDOS sessions (or VMs — virtual machines — as they are called in '95) start off in a window. There's no support for displayed TrueType fonts under DOS: the result is that as you re-size the window, the fonts change. For example, if you click on the maximise button, the DOS window grows to fit the available desktop, but still displays the bottom status bar and the top window bar (see screenshot below). Since the TrueType fonts are a bit skinny, you can go back to a traditional display by selecting "full screen".



Win95 uses TrueType fonts for text display in a DOS box and adjusts point size dynamically with window size

Now you're up and running, it's worth pointing out one of the problems with '95:long filenames. These are great when the application supports them, but existing DOS applications (or old Windows apps) don't know what to do with them. As you can see from the screenshot, XtreePro is looking at the subdirectories created by '95 and has to truncate the long names with a '~' symbol. It does no harm, but it's annoying.

Other lower-level modifications mean that almost any MSDOS application will now run in a window and, if one crashes, you can kill a DOS VM by closing the window, rather than having to type "exit" (although you can do it this way too).

Microsoft has improved the use of memory, so now protected mode DOS drivers are loaded out of conventional memory altogether. For example, if you are connected to a LAN and have a local CD-ROM drive, '95 will replace these drivers with its own 32-bit protected mode drivers and shunt them out of conventional memory: the result is a saving of around 180kb.

You can run all the same commands as are in MSDOS 6.22 from the '95 command line. You can even get rid of the Windows GUI and only use the command line. Lastly, '95 will run Windows and DOS applications as fast as or better than DOS/Windows 3.1 on a similar base-level PC with an 80386 and 4Mb of RAM.

Oldies but goodies

I have noticed that you are beginning to exclude older users! That is, PCs running pre-DOS 5 which don't have all the features of the current operating system.

I have no particular desire to upgrade, since I have a basic 80286-based PC running its original version of DOS. However, since I wanted to try out some of your memory-saving tricks I soon began to get bored with editing Config.sys and re-booting the PC.

Fig 1 Assembly listing for Reboot.com

```
START   MOV  AX,0040          :set memory segment to 0040
        MOV  DS,AX            :set up segment register
        MOV AX,1234           :load value to create a warm boot
        MOV [72],AX           :into memory location 72
        JMP FFFF:0000         :jump to BIOS init routine to reset PC
```

Fig 2 Assembly code for ASCII code 7

```
MOV DL,07        ;ASCII character code for a beep
MOV AH,02        ;send the character to the
INT 21           ;standard output device
INT 20           ;terminate
```

Fig 3 Creating sounds with the 8253 chip

```
    MOV BX,0400          ;the frequency in Hz you want
    MOV CX,0036          ;duration of sound in multiples of 1/18 of a
                         second
    MOV DX,0012          ;frequency divisor
    DIV BX                ;calculate frequency value
    MOV BX,AX            ;save result
    MOV AL,B6            ;speaker control register setup
    OUT 43,AL            ;send to 8253 chip
    MOV AX,BX            ;restore frequency value to AX
    OUT 42,AL            ;output lower byte of frequency value to 8253
    MOV AL,AH            ;
    OUT 42,AL            ;output upper byte of frequency to 8253
    IN AL,61             ;get Timer 2 output port register value
    OR AL,03             ;turn speaker control bits on
    OUT 61,AL            ;switch speaker on
    PUSH DS              ;start delay count, save segment register
    MOV AX,0040          ;
    MOV DS,AX            ;set segment register to BIOS data area
    MOV BX,[006C]        ;get current timer value
    ADD BX,CX            ;add delay value (set in second line)
OUT  CMP BX,[006C]       ;check for timeout
    JA OUT               ;no timeout, check again
    POP DS               ;end delay, restore segment register
    IN AL,61             ;
    AND AL,FC            ;turn speaker control bits off
    OUT 61,AL            ;switch speaker off
    MOV AH,4C            ;return to DOS
    INT 21
```

Fig 4 Debug script

```
N SOUND.COM
E 0100 BB 00 04 B9 36 00 BA 12
E 0108 00 F7 F3 89 C3 B0 B6 E6
E 0110 43 89 D8 E6 42 88 E0 E6
E 0118 42 E4 61 0C 03 E6 61 1E
E 0120 B8 40 00 8E D8 8B 1E 6C
E 0128 00 01 CB 3B 1E 6C 00 77
E 0130 FA 1F E4 61 24 FC E6 61
E 0138 B4 4C CD 21

RCX
3C
W
Q
```

I have written this simple assembly-language script which carries out a soft reboot. Readers could use it for all sorts of reasons: to get rid of intruders who don't enter a password, to reset a PC and load a different configuration (perhaps with network or device drivers) or to test out new entries in the setup files.

The listing creates a small .COM file called Reboot.com; the assembly listing for it is in *Fig 1*.

You could change this routine to carry out a cold start by changing the value stored in location 0040:0072hex to any value other than 1234hex. The Debug script for Reboot.com is as follows:

```
N REBOOT.COM
E0100 B8 40 00 8E D8 B8 34 12
E0108 A3 72 00 EA 00 00 FF FF

RCX
10
W
Q
```

To create the COM file, type in the debug script as an ASCII file and redirect it to DEBUG to compile this into a COM file.

PCW *As you point out, the newer versions of DOS, specifically 6.x, let you create Config.sys files with multiple configurations that can be chosen at boot-up time (as I've described in previous columns). For anyone with an older version of DOS, your solution is a rather unsubtle but very effective one!*

The sound of silence

I am getting rather bored with the lack of sounds from my PC. Every other PC is now a multimedia monster, but mine remains resolutely silent. Is there any way of creating sounds from DOS?

PCW *The basic PC sound generation hardware, coupled with the DOS functionality in this area, mean that all you're able to get is a feeble "beep" when you issue an ASCII code 7 character. However, even this aural feast isn't supported by DOS, so if you want to add sound to a batch file, you'll need to write a little assembly-language program that will output an ASCII code 7 character. The assembly code is very simple (Fig 2) and the following is its Debug script:*

```
N BEEP.COM
E 0100    B2 07 B4 02 CD 21 CD 20

RCX
8
W
Q
```

This little program creates a file called Beep.com which you can include in any batch file. This is hardly going to set musical hearts racing, so you should turn to the timer chip in your PC for more sophisticated sounds:

Inside your PC is an 8253 timer chip that contains three counters/timers which are used to time events under software control. The third timer, Timer 2, is the interesting one for our purposes, since it's connected to the speaker and can be used for create tones. To produce a "beep" of a different frequency, change the values sent to the 8253 timer and it will do the rest.

Fig 3 is a rather long but complete assembly language program to create sounds with the 8253 chip, and Fig 4 is its Debug script.

This program will produce a tone with a frequency of 1024Hz and sound this for three seconds. You can easily change these values or create a set of COM files that play different notes for warnings, buzzes or other effects.

PCW Contacts

Write care of *PCW* or via email to **scollin@cix.compulink.co.uk** or CompuServe **72241,601**