

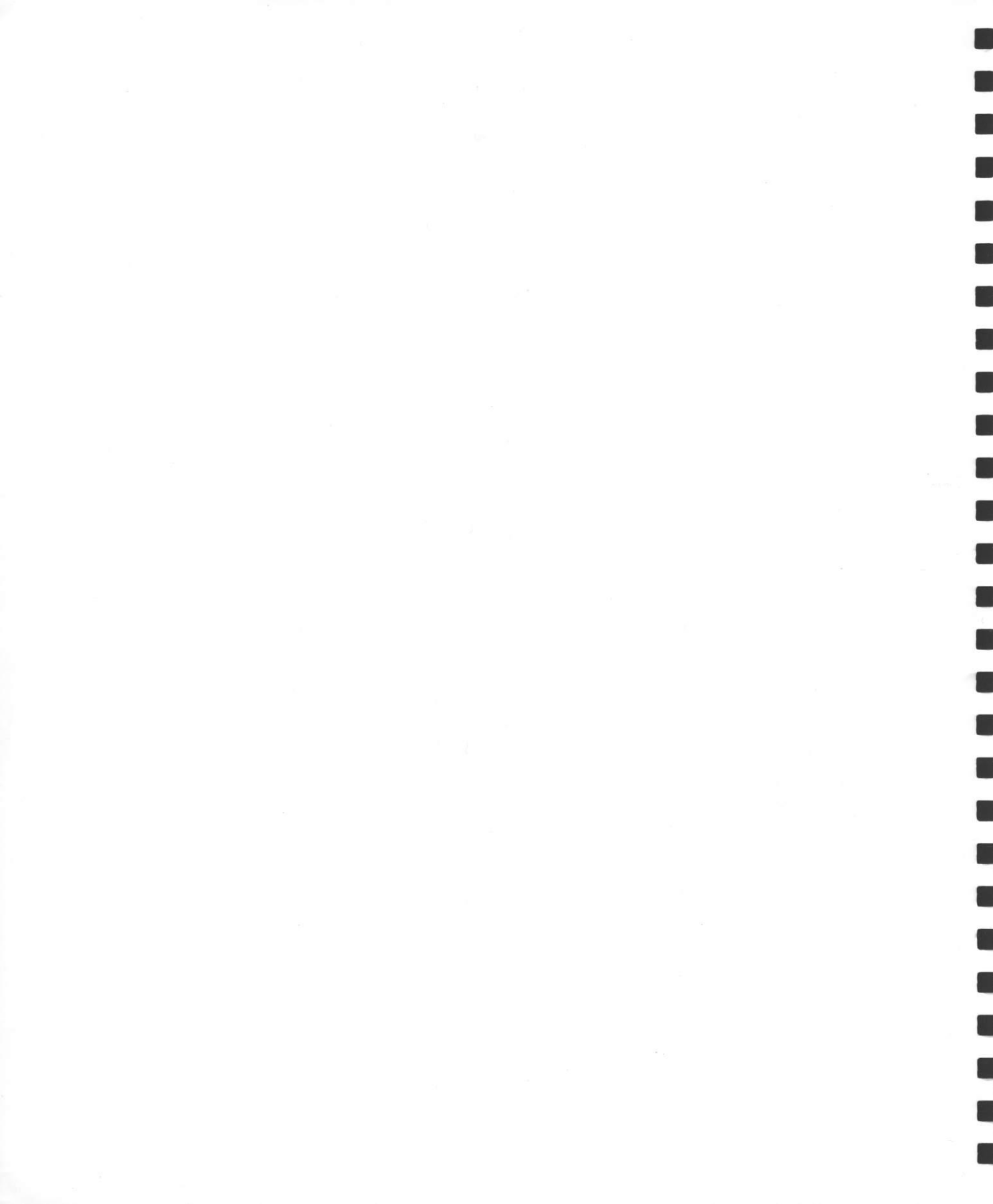
A F U T U R E B O O K

PC PLUS

T H E
**HELP
SCREEN**
C O L L E C T I O N

The Most Comprehensive Compilation
of PC Compatible Hints,
Tips and Useful Programs
Ever Published

EDITED BY MATT NICHOLSON



T H E
**HELP
SCREEN**
C O L L E C T I O N

The most comprehensive Compilation
of PC Compatible Hints,
Tips and Useful programs
Ever published

EDITED BY MATT NICHOLSON

T H E
**HELP
SCREEN**
C O L L E C T I O N

AND
COMPANION DISK

**Published by
Future Publishing
30 Monmouth Street, Bath BA1 2AP**

© 1990 Future Publishing

No part of this book may be copied in any way without prior written permission from the publishers. The disk that is available with this book also contains copyright material, which means you are welcome to take back up copies and adapt the contents for your own use, but can only distribute copies of the programs with the author's permission.

This book was written with *WordStar 4* using a wide range of PC compatibles, ranging from an Amstrad PC1512 to a CompuAdd 386/16. Final page make up was done on an Apple Macintosh IIci running Quark *XPress* version 2.1. The screen shots were taken straight from screen memory and converted to TIFF files using *Pinch and Punch* version 4, written by Iolo Davidson.

Contents

<i>Introduction</i>	<i>5</i>
<i>1. The Basics</i>	<i>7</i>
<i>2. Using MS-DOS</i>	<i>19</i>
<i>3. Batch Files</i>	<i>31</i>
<i>4. Printers.....</i>	<i>43</i>
<i>5. GEM & BASIC 2.....</i>	<i>53</i>
<i>6. Windows</i>	<i>67</i>
<i>7. Using Hard Disks</i>	<i>75</i>
<i>8. Hardware</i>	<i>85</i>
<i>9. Applications.....</i>	<i>103</i>
<i>10. Programming</i>	<i>121</i>
<i>11. Debug Hacks.....</i>	<i>137</i>
<i>12. The Disk.....</i>	<i>147</i>
<i>Index</i>	<i>153</i>

Tips which display this symbol  have an associated file on the disk which accompanies this book. For further details refer to page 148.

Introduction

The magazine *PC PLUS* was launched in September 1986, on the same day that Amstrad launched the PC1512. Just as Amstrad's new machine was set to break the mould of PC computing, so *PC PLUS* was set to break the mould of PC computer magazines.

With the PC1512, Amstrad widened the horizons of the PC compatible beyond the corporate office into the small business and even the home. A whole new breed of low-cost software appeared, together with a whole new breed of PC user. This new type of user wanted information above all else. They might have used a BBC, Commodore or Spectrum before, or the office PC under the guidance of a DP manager, but the complexities of MS-DOS, using hard disks, installing software and so forth were all new. Right from the start *PC PLUS* provided the *Help Screen* section, and right from the start we were deluged in letters. Eventually *Help Screen* settled at around six pages a month but the magazine received, and still receives, enough to fill far more.

The letters ranged from small businesses wanting to set up databases, to novices baffled by the A-prompt, kids wondering how to get the joystick to work, and old hands with a nifty batch file, C program or DEBUG hack. In the earlier issues the peculiarities of the Amstrad PC1512, GEM, DOS Plus and BASIC 2 (all come with the 1512) dominated. Later, as other low cost PCs appeared and the magazine widened its appeal, *WordStar*, *Lotus 1-2-3*, *GW-BASIC*, Pascal, C and even *Windows* took over.

This book is a compilation distilled from three years of *Help Screen*, and inevitably reflects this diversity. The letter and answer format of the magazine has been changed into one of straight 'tips', and the tips have been organised into 11 chapters, ranging from 'The Basics' for those stuck at the A-prompt, to 'DEBUG Hacks' for those who fancy exploring assembler. Many of the earlier tips have been heavily edited to allow for their considerable bias towards the PC1512, but the 'GEM & BASIC 2' chapter is still pretty meaty.

Many of the batch files and utilities from the book have been put on the Companion Disk, an optional extra for those not too keen on creating the utilities with DEBUG and assembler. Also on the disk are three larger utility programs in the shape of a text editor, a file editor and a disk sector editor. These are updated versions of programs

originally appearing on the *PC PLUS SuperDisk*, the cover-mounted disk that comes with the magazine every month. See Chapter 12 for further details.

The tip format of the book has meant that most of the tips have not been credited to the authors of the original letters. Where we have used copyright material we have endeavoured to contact the author, and have put a credit where appropriate. If you feel that copyright material has been used and have not been contacted, please get in touch with the publisher as soon as possible. Our thanks go to all the readers who contributed to *Help Screen* over the years, without whom this book, and indeed a major part of *PC PLUS* itself, would not have been possible.

Thanks are also due to Paul Stephens (present Editor of *PC PLUS*), Simon Williams and Andy Wilton who, together with myself, provided the answers to most of the original letters; Jane Toft, the original Art Editor of *PC PLUS*, for the design and artwork of this book; Tim Kay of Silicon Systems for always being prepared to discuss disk drives; Howard Fisher of Locomotive Software, who faxed us many a BASIC 2 listing; Alan Solomon of the IBM PC User Group for some useful tips; and Chris Lusby Taylor of Digital Research who was always willing to talk about GEM. Thanks also go to Paul Mullen of the Independent PC User Group for his many contributions, particularly when both user group and magazine were starting out. His user group, on (0732) 771512, is well worth joining. Finally, thanks go to Hazel, Jemma and Luke for their support, infinite patience and love.

The book is fairly straightforward to use, but adopts certain conventions used by *PC PLUS* magazine that are worth bearing in mind. Software products are named in *italics*, and file names in CAPITAL letters. MS-DOS commands that are to be typed in and program listings are shown in a **different** typeface. Keyboard keys are indicated by square brackets, so [A] means 'the A key' and [F11] indicates function key F11. When a command is 'entered' it means the command is to be typed as shown, and then [return] or [enter] pressed so the computer can execute it. If a sequence is to be 'typed', don't press [return] unless you are specifically told to. Remember that any prices mentioned include VAT, and were current at the time of publication.

Matt Nicholson

T H E
**HELP
SCREEN**
C O L L E C T I O N



T H E B A S I C S

1. By-passing GEM

It is perhaps unfortunate that Amstrad chose to emphasise GEM so much in the 1512 and 1640 manual, as many beginners have got the impression that GEM is the only way of using a PC; particularly if their previous experience has been with home computers such as the BBC, Commodore 64, Amstrad CPC6128 or Sinclair Spectrum. It is not: the vast majority of programs available for the PC do not need you to bother with GEM *Desktop* at all.

The main difference between micros like the BBC or 6128 and a PC is that the PC requires you to load the 'operating system' from disk, while a home micro usually has it already built in. When you first turn on your machine a message appears saying 'Insert a SYSTEM disk into drive A'. Put in a copy of your red MS-DOS Utility Disk, press a key and away you go – once the A-prompt has appeared you can load a conventional application by entering just one command.

2. File Types

You can call a file anything you like, so long as you don't use illegal characters and it is in the correct format. But there are certain reserved file types used by MS-DOS and GEM that it is wise to steer clear of unless you know what you are doing. There are also a few conventions worth bearing in mind, particularly if other people will be using your disks:

EXECUTABLE FILES

There are basically three types of program files that

can be run directly from the MS-DOS prompt simply by entering the first part of the file name, together with any variable that might be required. They are listed here in order of precedence:

COM files, short for 'command files', have top priority. An example is `APPEND.COM`, to be found on your MS-DOS system disk – it would be run by simply entering `APPEND` followed by the parameters you wish the program to act on.

EXE files, short for 'execute files' are very similar to COM files. Again, there are plenty of examples of these on your MS-DOS system or utility disks.

BAT files, otherwise known as 'batch files'. These allow you to run a whole series of operating system commands automatically, in a batch as it were. A particularly important batch file is `AUTOEXEC.BAT`, which executes automatically when the system is booted up.

SYSTEM FILES

SYS files are to do with the setting up of the operating system itself. The file `CONFIG.SYS` is used to configure the operating system to act in the way you require, and makes use of an assortment of 'device drivers' such as `RAMDRIVE.SYS`, `DRIVER.SYS` and `ANSI.SYS` to set up a particular system.

COMMON FILE TYPES

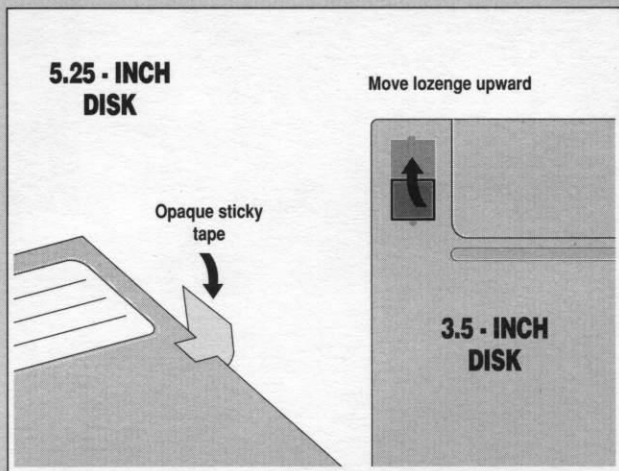
BAS files are usually programs written in the BASIC language, either *GWBasic*, *QuickBasic*, *BASIC 2* or another variety of this language. These programs are run by loading the BASIC interpreter first and running the program from there.

3. Keeping Your Disks Safe From Accidents

When you buy a brand new application program it is always wise to put write-protect tabs (short lengths of opaque tape usually supplied with blank floppy disks) over the notch you will find on each 5.25-inch disk. This means you cannot accidentally erase or change any of the files on your 'master disk'.

You should then follow the installation instructions in the application's manual to the letter, which may well instruct you to make copies of the master disks and use these rather than the master disks themselves. Occasionally you may find that the installation process requires some data to be written to the master disk before the programs can be used – some copy protection systems require this, for example. This means removing the write-protect tab temporarily, but remember to replace it as soon as the installation process is completed.

3.5-inch disks have write-protect tabs built in to the case. To the front-left of the disk (holding the disk as if you were about to insert it into the drive) is a small square opening, and a black plastic tag that can be clicked so as to cover or leave the hole open. The contents of the disk can only be changed if the hole is covered.



BIN files are stored in binary code. Files written in machine code are usually saved with this extension.

BAK, otherwise known as 'backup files'. If you save, for example, a file called STORY.DOC that you have been working on, many applications will rename the previous version of that file STORY.BAK and keep it on disk as a backup in case anything goes wrong.

OVR, or 'overlay files' are special program files that act as an extension to the main program. To save space these are kept on disk and only called into memory when needed.

DOC files, or 'document files' usually hold text stored as straight ASCII or in *WordStar* format. They are often used to hold information about the software on the disk.

\$\$\$ files are used by programs to temporarily store parts of files on disk that are too big to fit in memory all at once. A program should delete temporary files automatically when it has finished with them.

GEM FILE TYPES

GEM uses several file types for its own purposes, as you can see if you look at the directories of the GEM master disks. If you are using GEM it is as well to watch out for these:

APP, or 'application files' hold the application programs themselves.

RSC files are 'resource files' that hold data needed by application programs.

ACC files are concerned with the desktop accessories such as SNAPSHOT or the calculator and clock.

OUT files, or 'output files' are created by GEM software when you wish to output through the GEM program OUTPUT.

IMG, or 'image files' are bit-mapped picture files created by SNAPSHOT or applications such as *GEM Paint*.

LIS, or 'list files' are used to store the print queue list used by OUTPUT.

GEM files are drawings created by applications such as *GEM Draw*, or are linked to bit-mapped images created by the likes of SNAPSHOT.

4. Batch Files

These are special files, with the file extension BAT, which consist of a series of MS-DOS commands. They are executed by simply entering the file name, without the BAT extension.

There are several ways of writing batch files. Perhaps the simplest of all is to use the command `copy con batch.bat` which is a special use of the COPY command that copies what ever you type straight into a file called, in this instance, BATCH.BAT. When you have finished, enter 'Control-Z' by holding down [Ctrl] while tapping [Z], and then pressing [enter].

The new file will then be 'copied'.

You can call a batch file anything you like, though it is advisable not to call it by the same name as any other files on the disk with the extension COM or EXE as these have priority. Perhaps the most useful name is AUTOEXEC.BAT, as this performs a unique function in MS-DOS. Any file with this name is AUTOMATICALLY EXECUTED every time you turn your PC on or re-boot by holding down [Ctrl] and [Alt] and simultaneously pressing [Del].

For example: supposing you have used CONFIG.SYS to set up a RAM disk (called Drive C), and want to set up a disk to automatically copy the program file PROGRAM.COM into RAM disk, transfer control to the RAM disk, and then run the program. On the way you might also set VERIFY ON so that any file copying is automatically verified. You would do this by entering the following on the A-prompt:

```
A>copy con autoexec.bat
VERIFY ON
COPY program.com c:
c:
program
^Z
```

1 File(s) copied

5. Printing files

You can send a file to the printer by first pressing [Ctrl] and [P], which echoes screen display to the printer, and then using the TYPE command. [Ctrl [P] is a toggle, so press it again to stop echoing to the printer.

However a better way is `COPY filename PRN`. The PRN part of this command is a special MS-DOS 'reserved file name' which means the currently selected printer. You won't see the file's contents displayed on the screen, and your print won't contain any surplus A-prompts.

6. Making a Work Disk

When you buy a new software package – a word processor, spreadsheet, database or whatever – you might expect to be able to simply insert your brand new program disks and get on with it. Unfortunately this is not usually the case; indeed it would be most unwise, as these are your only copies of the program and if anything went wrong you would truly be up the creek with no back up.

Commercial software usually comes on non-system disks: disks that do not contain the relevant operating system and so will not work on a PC cold. There are two good reasons for this. First of all, it is wise to make copies of the valuable files they contain: a process known as 'backing up' your master disks. Secondly, most programs need to be 'installed' for your particular

machine.

If you have a hard disk in your PC, installation usually involves simply copying the relevant files to your hard disk and carrying on. If, however, you are restricted to floppy disk drives then you need to install the program onto a separate disk that you can simply plug in and go. The master disks may contain an installation routine that does this for you automatically, possibly a batch file executed by entering **install** at the MS-DOS prompt. However it is possible to tailor your 'Working Disk' to your particular requirements and, if you have a 512 or 640K machine, speed it up considerably.

Normally your program is held on floppy disk and called into memory when loaded. However it is possible to arrange for the program to be held in 'RAM disk' so that it will operate much faster and free the disk drive for your more important data files. This technique is particularly useful if you only have a single-disk machine. This is done by means of three very handy files to be found on the MS-DOS system disk supplied with your machine: **CONFIG.SYS**, which tailors certain aspects of your PC when you boot it up; **RAMDRIVE.SYS**, which sets up a RAM disk; and

AUTOEXEC.BAT which is a special batch file that is automatically executed when you turn on your PC.

The process involves first of all tailoring the file **CONFIG.SYS** so that it sets up a RAM disk in your computer's memory of suitable size, and then writing a file called **AUTOEXEC.BAT** that makes sure that the program files are copied in to the RAM disk when the system is booted up. Once this is done you can insert your new Work Disk and boot up the system (either by turning it on or pressing [Ctrl][Alt] and [Del] simultaneously). Then you can take out your Work Disk, insert your data disk, and get on with running the program. It will run very fast because it is in RAM and doesn't have to be read off disk; and you will have all 360K of your data disk for storing files.

By way of example, let us look at how you might make a special Work Disk for the word processor *WordStar*. The details might vary with different versions, but the principles can be applied to any application:

1. Insert your backup copy of the MS-DOS system disk into the floppy disk drive, and enter the command **Format b:/s** to format the blank disk that is to

10.

7. Making a System Disk

Hard disk owners can have all the software they need at their fingertips thanks to the hard disk's huge capacity. If you are confined to floppy disks you need to put together special 'work disks' tailored to each application's needs, so you can boot straight into your work with the minimum of fuss. You start by making a 'system disk':

FORMAT B: /s/v

This is the first stage in the process. The **FORMAT** command clears the disk, in this case in drive B, of files and ensures that it is ready to accept new data. The **/s** option puts the three files that make up the operating system on the disk, so ensuring that the computer will 'boot up' from it. The **/v** is optional, but if used prompts you for a digital name tag of up to 11 characters which will be displayed every time you ask for a directory.

FORMAT is an 'external' command, which means that the disk in the default drive must contain the program file **FORMAT.EXE**, together with the MS-DOS system files. All these files are to be found on your MS-DOS System and Utilities disk. If you have a twin-drive machine a copy of this System disk should be in the left-hand drive and the new disk in the right-hand drive. If you have a single-drive machine you should swap the disks over when prompted.

The three files that make up the MS-DOS operating system are **IO.SYS**, **MSDOS.SYS** and **COMMAND.COM**. These take up at least 70K, although only the file **COMMAND.COM** is listed on a DIRectory listing as the other two are 'hidden'.

CONFIG.SYS

Whenever your PC is booted up the operating system reads the contents of the file **CONFIG.SYS** and configures itself accordingly. It can be created and altered using any text editor, and consists of a series of discrete instructions,

any of which can be omitted. Remember that changes made to **CONFIG.SYS** only take effect when the system is re-booted by pressing [Ctrl], [Alt] and [Del] simultaneously, or powering up from scratch.

These are the more useful entries for a **CONFIG.SYS** file, the top group generally recommended and the bottom group to be used when required:

Files=20 This sets the number of files that can be open at any time. 20 is a large number, but each entry only increases the size of the operating system by 39 bytes.

Buffers=20 A Buffer is an area of memory set aside to hold data read from a disk that might be needed by the program later. Each buffer takes up 528 bytes, but memory is not at a premium on most machines these days and a value of 20 noticeably speeds operation.

Country=044 This looks after the presentation of various country-dependent information such as time, date and currency. Quite a number of countries are catered for by using the international dialling code. Under version 3.3 or 4.0 of MS-DOS this command requires the file **COUNTRY.SYS** as well.

The option of loading specialised Device Driver files is a valuable one, as it allows MS-DOS to cater for unusual or specialised plotters, scanners, printers or whatever. Two such Device Drivers are supplied with most PCs:

Device=ANSI.SYS The PC screen can be handled in a pretty basic fashion by sending it 'control codes'. As it stands the screen conforms to Teletype specification, but loading the ANSI driver provides enhanced facilities. This command also requires the presence of the driver file **ANSI.SYS** on the system disk, and is necessary if the display is cluttered with superfluous characters. Otherwise it shouldn't be used as it occupies unnecessary space.

become your Work Disk – the /s ensures that the operating system is copied on to the disk too. If you only have one drive forget the b: part of the command.

2. Copy the files KEYBUK.EXE and RAMDRIVE.SYS onto your Work Disk:

```
A>copy keybuk.exe b:
      1 File(s) copied
```

```
A>copy ramdrive.sys b:
      1 File(s) copied
```

3. Remove the MS-DOS system disk from drive A and replace it with the first *WordStar* disk. At this point we have to decide which files are going to be needed on the Work Disk, and copy them over. In this case we also decided to copy over WC.COM as well – a useful little program to be found on most *WordStar* disks that counts the number of words you've written:

```
A>copy *.ovr b:
WSMSG.S.OVR
WSPRINT.OVR
      2 File(s) copied
```

```
A>copy *.exe b:
WS.EXE
WINSTALL.EXE
WSCHANGE.EXE
      3 File(s) copied
```

```
A>copy ws.com b:
      1 File(s) copied
```

4. We now have all the relevant program files on our work disk, so remove the *WordStar* disk from drive A and replace it with your work disk to be. Next we have to decide how big a RAM disk to create, which means deciding which files are going in there. DIR shows that the necessary files, including overlay files with the extension OVR, take up 145086 bytes, so 150K seems in order. This is set up in the file CONFIG.SYS:

```
A>copy con config.sys
files=20
buffers=5
device=ramdrive.sys 150
country=044
^Z
      1 File(s) copied
```

and slows the screen down a fraction.

Device=RAMDRIVE.SYS nnn This driver separates a portion of the available RAM from the rest and treats it as an extra high speed disk drive, usually labelled 'Drive C'. The figure **nnn** is the size of the RAM disk in kilobytes, or on an Amstrad PC1512 or 1640 it can contain the letters **nvr** in which case it is set to the size specified in the non-volatile RAM area. This instruction requires the presence of **RAMDRIVE.SYS** on the system disk.

AUTOEXEC.BAT

The MS-DOS operating system comes with a little language of its own which enables you to create 'batch files', files containing a number of MS-DOS commands which can be executed automatically. Batch files can be called anything but must have the file extension .BAT. A batch file called AUTOEXEC.BAT is a special case though, as it is AUTOMATICALLY EXECUTED whenever the system is booted up. These are the instructions that should be considered for any AUTOEXEC.BAT file, in order of importance:

KEYBUK Installs a resident program that ensures the UK specific keys £, £, @, & and " are mapped to the correct places on the keyboard. On the Amstrad PC1512 and 1640 it also ensures correct handling of the non-standard [Del->] and [Enter], plus the joystick port and the mouse buttons. This command requires the presence of the program file **KEYBUK.EXE** on the system disk. Under version 3.3 or later the command is **KEYB UK**, and both **KEYB.EXE** and **KEYBOARD.SYS** are required.

verify on This ensures that any files copied to disk are checked against the original. It slows down disk access a fraction, but provides warning of any problems.

MOUSE Installs another resident program which allows software to make use of your mouse (some mice work in a different manner, so check your manual

first). It requires the presence of **MOUSE.COM** on the system disk. A waste of space if you're not using the mouse.

prompt \$p\$g The PROMPT command allows you to tailor the MS-DOS 'A-prompt' as you wish. On a floppy drive system you might as well leave it as it comes – simply showing you which is the current drive – but on a hard disk this useful command displays the current directory as well.

GRAPHICS /r loads a printer driver that enables [PrtSc] to print a copy of the screen display if in graphics mode. Only really needed if you are running GEM, where the /r ensures that the background is not printed. May occasionally be required if you want to print screens from other packages, but few other applications run in graphics mode. It requires the file **GRAPHICS.EXE** on the system disk to function.

FILE CHECKLIST

Those in brackets are only necessary if you have added these facilities:

COMMAND.COM	(ANSI.SYS)
CONFIG.SYS	(COUNTRY.SYS)
AUTOEXEC.BAT	(RAMDRIVE.SYS)
KEYBUK.EXE or KEYB.EXE	(KEYBOARD.SYS)
	(MOUSE.COM)
	(GRAPHICS.EXE)

WHAT NEXT

Once you have created your system disk you can copy onto it the files from your application master disks that you need for everyday work. If the last line in the AUTOEXEC.BAT file is the command that invokes the application program, then you will have created a true turnkey work disk: put it in the disk drive, turn on the power and your application will load automatically.

5. Finally we have to create the appropriate AUTOEXEC.BAT so that the required files are automatically copied to our new RAM disk:

```
A>copy con autoexec.bat
ECHO OFF
VERIFY ON
KEYBUK
copy ws*.ovr c: >nul
copy wc.exe c: >nul
copy wc.com c: >nul
c:
path a:
ws
^Z

1 File(s) copied
```

6. Your Work Disk is now ready. To use it, insert it in to drive A and reset your PC. AUTOEXEC.BAT will auto exec and you will be presented with the *WordStar* opening screen. In this case it would be advisable to use WINSTALL to configure your version of the program for your needs, in particular so that it automatically logs on to drive A as this is where your documents will be held.

8. Display Adapters

IBM took a modular approach to much of the design of the PC range, which cushioned it to some extent against future developments. The IBM PC display is actually made up of two components: the monitor itself and a separate Graphics Adapter that fits into the PC's expansion slots and provides much of the electronics needed to drive the display.

The original PC, introduced in 1981, came with

9. MS-DOS Internal Commands

These are the more useful commands 'built into' MS-DOS which don't have to be loaded from disk each time they are used:

FILE SPECIFICATION

The full specification for a file includes a number of parameters indicating its position relative to the current drive and directory:

<drive>: Which drive the file is on. **A** and **B** indicate the floppy drives, while **C** is the hard disk if one is fitted or the RAM disk if not. On a hard disk machine **D** is the RAM disk.

Machines with only one floppy drive still offer both A and B logical drives, the operating system requesting you to insert the appropriate disk for each drive as required.

If no drive is specified, the file is assumed to be on the current drive.

<path>: The names of the directories that you would pass through on the path down the directory tree structure to that holding the file, each separated by a back-slash.

If preceded by a \ it means 'start at the root directory'; otherwise the path is from the current directory. If the path is not specified the file is assumed to be in the current directory.

<filename>.<extension> The filename consists of up to eight characters. In addition it can have an optional 'extension' of up to three characters separated from the first part by a stop.

Extensions can be used with both file and directory names. To avoid confusion, however, it makes sense always to put an extension to a filename, but never to a directory name.

These parameters can be made up from most of the characters available from the keyboard, including dashes and underlines. Spaces cannot be used, and neither can the following characters:

< > = | / [] , . \ + { } ! * ? " ' ^

The characters ? and * are called 'wildcards', and can be used in the filename

and the extension with many MS-DOS commands to capture a range of files in a given directory.

? can be used to represent one valid character in its position;

* represents any number of characters to the right of its position.

DIRECTORIES

The directory structure provides a useful way of organising a large number of files. Absolutely vital if you have a hard disk:

MD <path> Make a new sub-directory within the current directory.

CD <path> Change current directory to that specified.

CD or **CD <drive>**: display the name of current directory, or that of the specified drive.

CD .. Takes you to the parent directory – the one immediately above the current directory.

**CD ** Takes you to the root directory.

RD <path> Removes specified directory, provided it is empty. The directory can be emptied quickly using **DEL <path>**, although this will not remove the sub-directory itself.

PATH <drive>:<path>;<path>;<path> ... Tells MS-DOS which directories to search for external command files, in addition to searching the current directory.

COPY

COPY <sourcefile> /A/B <targetfile> /A/B/V

Copies the source file(s) to the target file(s) (CP/M users note: it works the other way round from PIP). Either source or target can be by default rather than specified, so:

COPY a:letter b: copies the file called 'letter' to drive B;

either a Mono Display Adapter (MDA) coupled to a black and white monitor that was capable of a fairly high resolution text display; or a Colour Graphics Adapter (CGA) coupled to a colour monitor that could display text and graphics, but at a lower resolution and in a limited range of colours. These look pretty primitive these days, but thanks to the PC's design most PC owners can take advantage of technological advances simply by plugging in new graphic adapters capable of driving higher resolution monitors as they appear. Unfortunately one of the machines on which this is not really feasible is the Amstrad PC1512.

In 1984, the year the AT was announced, IBM also launched the Enhanced Graphics Adapter (EGA). This was fully compatible with the earlier adapters but could provide up to 16 colour text and graphic displays on suitable monitors. Then in 1987 IBM launched the Video Graphics Array (VGA) as part of its PS/2 range, which provided even higher resolutions while still

maintaining compatibility with previous adapters. VGA was actually built into the main circuit board of the PS/2 machines, but other companies soon produced VGA adapters that could be plugged into the expansion slots of PC and AT compatibles.

Each of these adapters comes with a character set which means that the text from text-based programs is automatically displayed at the higher resolutions the adapter offers. Graphics-based programs such as publishing, drawing and graphing packages require special software drivers to take advantage of these modes, however these are usually incorporated into the package or provided by a graphics environment such as Digital Research's *GEM* or Microsoft's *Windows*.

CGA

Many clones can emulate IBM's CGA display card, which stands for Color Graphics Adaptor (American

COPY letter b: has same effect if the current drive is A;
COPY a:letter has same effect if current drive is B.
Wildcards can be used as required.

/V means that the transfer is checked and repeated if necessary. Has the same effect as **VERIFY ON**.

/A designates the file as an ASCII file. If used with the Source, the copy will take effect up to the first EOF character (ASCII 26 or ^Z). If with the Target, then an EOF character will be added to the end of the file.

/B designates the file as a BINARY file. This means that EOF markers are ignored, so **COPY** checks the directory entry to find out how long the file is and simply copies that number of bytes.

COPY <file1>+<file2>+<file3> <targetfile>

Joins the contents of the three files together and puts them into one target file. If the target file is omitted a new FILE1 is created containing the contents of all three, so use carefully!

Valid filenames include the devices **CON** (the keyboard and screen), **PRN** (the printer port), **AUX** (the serial port), or **NUL** (a dummy device):

COPY CON <filespec>

Ideal for entering short files, this copies anything you type on the keyboard into the file specified. Terminate the file by entering [Ctrl][z] on a fresh line.

COPY <filespec>+CON

Anything you type is added to the end of the file specified. Terminated with ^Z as above.

COPY <filespec> /B CON

Displays the contents of the file on the screen.

COPY <filespec> PRN

Copies the file specified to the printer.

COPY <filespec> AUX /A

Copies the file through the serial interface in the manner set by the **MODE** command, terminating the file with an EOF marker.

OTHER COMMANDS

DIR <filespec> /P /W

Displays the file directory of the specified drive or path, or the current directory if these are not specified. Wildcards can be used. So, for example, **DIR B:** displays all files on drive B while **DIR *.EXE** displays all files in the current directory with the extension EXE.

/P displays the directory a 'page' at a time.

/W displays the directory in a 'wide format' with five files per line.

TYPE <filespec>

Displays the contents of the file specified on the screen. Wildcards are not permitted. Key [Ctrl][S] to pause or continue display, [Ctrl][C] to exit.

DEL <filespec>

Deletes files specified (**ERASE** can also be used). Wildcards can be used, but with care! It is a good idea to **DIR** with the same wildcards first.

REN <oldname> <newname>

Changes the name of a file. Wildcards can be used to change a number of files at the same time.

DATE <dd-mm-yy>

TIME <hh:mm:ss>

Sets the MS-DOS clock. On most PCs this is maintained by the batteries even when you switch off, so it should not need re-setting until you replace the batteries.

CLS

Clears the screen and returns the MS-DOS prompt. Can be used to return to default colours.

spelling because IBM is an American company). This can work in two different modes, either producing a 300 by 200 dot picture in four colours, or a 640 by 200 dot picture with one foreground colour. There are three different palettes for the four-colour mode: Cyan, magenta, black and white (the most common); red, green, yellow and black; or cyan, red, white and black (the most palatable).

Some PC clone makers put the same CGA circuitry into both colour and mono machines, so both the Amstrad PC1512 MD and CD models have CGA fitted. The mono monitor simply shows the colours as shades of grey, just like a monochrome TV set showing a colour transmission. Any software described as requiring CGA will run on a monochrome PC1512.

The only possible drawback is that some colour combinations which look good on colour monitors don't look so good as shades of grey, and in some cases it can be difficult or indeed impossible to work out what is going on. Some business software lets you choose your own colour set-up, or has a specific 'mono' setting, but this isn't normally true of games.

The Amstrad PC1512 also has a special 'Amstrad' display mode offering a 640 by 200 dot picture with 16 colours, but this is not compatible with any standard IBM display adaptor. Only software which has been specifically written for the PC1512 will run in this mode, and software which only uses this mode won't run at all on any other PC clone. The situation for PC1512 owners is therefore that four colours are the limit for all but a small selection of software. Only the few programs which specifically say 'Amstrad PC1512 16-colour mode' will give the wider palette.

AMSTRAD PC1640

All PC1640's are fitted with an Internal Graphics Adaptor (IGA) which is in fact a third-party product, the Paradise display board, hard-wired into the machine. This is able to mimic the operation of many different types of display adaptor – in fact almost the full IBM PC range including CGA, EGA and the Hercules monochrome card. However the modes which you can use on your particular 1640 are controlled by the type of monitor you have fitted.

The PC1640 is available with three monitors:

10. Useful Books

For the first time user, *Running MS-DOS – 4th Edition* by Van Wolverton, published by Microsoft Press at £19.95, provides an excellent tutorial taking you from booting up through batch files and beyond. *MS-DOS Commands – Quick Reference Guide*, also by Van Wolverton from Microsoft Press, is a slim handy reference book at only £4.95.

monochrome (MD), standard-resolution colour (CD) and enhanced (high-resolution) colour (ECD). The emulation options for each monitor are as follows:

MD – Hercules, Monochrome EGA

CD – CGA

ECD – CGA, Colour EGA

These are physical restrictions. If for example you try to run Hercules monochrome emulation on an ECD monitor you get no display at all, due to the different signal frequencies used by the two devices.

Obviously, the IGA has to be told which display adaptor to emulate – it can't pop out and take a look at the label! The main way of doing this is with the DIP switch on the back of the 1640 system unit, which sets the 'default' mode, i.e. the one selected whenever you switch the machine on or reboot MS-DOS. The settings to use are the ones described on pages 11 and 12 of the PC1640 User Instructions manual, and give Hercules, CGA and colour EGA respectively for the MD, CD and ECD monitors. Your PC1640 should have these already set by your dealer on delivery – if not, complain!

Occasionally, owners of MD and ECD monitors may want to switch to the alternative display emulations which will work on their machines – Monochrome EGA for MD, CGA for ECD. This is where the DISPLAY program supplied with the 1640 comes in – it's a way of overriding the mode set by the DIP switch on the back. The Amstrad manual (page 438) lists a bewildering set of possible modes, but thankfully you don't need to use most of them. The most relevant modes are:

MD monitor – MDHERC1 (the default), MDMONO (monochrome EGA)

ECD monitor – CGA, ECD350

The format is **DISPLAY ECD350**, for example.

Even this is not often required, however, as most modern programs switch the graphics adaptor into the correct mode themselves. Games are a good example. Some even say 'supports CGA and EGA adaptors,' when what they really mean is that they switch the EGA back to CGA mode before starting. Occasionally though, games and other programs don't do the switching, which means chaos when they try to write CGA graphics to an adaptor which thinks it's an EGA, and that's when you need to run DISPLAY first.

So the PC1640 with monochrome MD monitor uses the Hercules display mode, unlike the PC1512 which uses the same CGA-compatible mode for both colour and monochrome. This caused some confusion to PC1512 owners upgrading to the 1640, who found they could no longer run CGA programs. If you have a monochrome PC1640 and wish to display CGA graphics you need a CGA monitor, such as the CD. This is the ideal solution as Amstrad was rather unconventional with the 1512 and 1640 in putting the

power supply for the whole system in the monitor, rather than in the system unit (the 2000 range is thankfully more conventional here). Alternatively any cheap colour CGA monitor will do, though you would have to keep your existing monitor under the desk for its power supply.

There is another, rather cheaper, possibility; a special program which lets the PC simulate a CGA card while actually running in Hercules mode. There are various versions, one being called *SIMCGA*, available from user groups and public domain software libraries for around £5. This doesn't work on all software though, and may reduce the apparent speed of programs.

HERCULES

For business text and graphics work a Hercules mono rig is decidedly better than a CGA with standard colour monitor. First there's the quality of the text display. The Hercules adaptor uses a 9x14 pixel text matrix, compared with the CGA's 8x8. This gives much sharper, more solid text of the same quality as an EGA display. You don't get any colours of course, but the Hercules does support a variety of effects such as half intensity, flashing and even underline, and virtually all business software runs very successfully using these options.

Secondly, there's the graphics. A Hercules card with mono monitor will normally give a graphics resolution of up to 720x350 pixels – higher even than an EGA system. And whereas games software for the Hercules is in very short supply, there's no problem at all with business software.

In fact virtually all business graphics programs include Hercules support, and in most cases give a much better display than when in CGA mode. Both Microsoft *Windows* and Digital Research's *GEM* work fine giving really crisp, fine-detailed graphics displays. Again you don't get colour, but with business graphics it's the amount of information not the colour scheme which counts, and the higher the resolution the more information you can impart in a single graph.

Finally, a decent monochrome monitor will give a clearer, sharper display than all but the very best colour screens due to the absence of the shadow-mask, phosphor colour triads and complicated circuitry. If you're using your PC for long periods of intensive text or data entry, then this, combined with the 9x14 text matrix, really makes a difference.

So if you're considering a PC for word-processing, database, spreadsheets, accounting or even Computer-Aided Design or Desktop Publishing, don't be put off by the 'Hercules-only' label on monochrome systems. It's true that an EGA setup gives you high-quality output and colour as well, but the difference in price can be considerable.

EGA

Although known as a high-resolution colour graphics card, the EGA was in fact also the first IBM adaptor which allowed monochrome monitors to display pixel graphics. This was a bit of an afterthought on IBM's part, and EGA monochrome was both pre-dated and overshadowed by the much more popular Hercules standard. What's slightly confusing is that in text mode, Hercules and mono EGA look exactly the same – it's only the graphics modes which differ.

11. Using CONFIG.SYS

When you first turn on your PC and 'boot-up' MS-DOS the operating system immediately looks for two files before turning its attention to you. These files are called *CONFIG.SYS* and *AUTOEXEC.BAT*.

The first of these files contains information that tells the operating system how you want it *CONFIGured* – and using it enables you to tailor your PC as you require. To see what we mean, 'boot up' your PC with your MS-DOS system disk and enter the following at the A-prompt:

```
A>TYPE CONFIG.SYS
```

something like this should appear:

```
files=20
buffers=5
device=ramdrive.sys 100
country=044
```

The command *TYPE* shows you what is in the file *CONFIG.SYS*. The *FILES* and *BUFFERS* entries should be left alone – what is interesting is the last two lines. *DEVICE* is used to specify a 'device driver', which is a program that tells the operating system how to use the keyboard, screen, memory or whatever.

In this case a RAM disk has been set up, the size of which has been specified as 100K. You could also have a line reading *device=ansi.sys*. *ANSI.SYS* is a screen driver that enables you to produce special graphic effects, should you wish.

The line starting *COUNTRY* specifies certain minor modifications to the keyboard that tailor it towards the needs of a particular country. The code required is in fact the international dialling code of the country in question.

The simplest way to change the contents of *CONFIG.SYS* is to load it into a word processor. Remember though that if you are using a version *WordStar* it should be loaded as a 'non-document'. Edit it to read as you require and save it. Remember though that any changes you make to *CONFIG.SYS* won't have any effect until you re-boot the computer, either with the on/off switch or by pressing [Ctrl], [Alt] and [Del] (the one at the bottom left of the keyboard) at the same time.

If you haven't got a word processor, you can always

change or create short files from MS-DOS by using the command:

A>COPY CON filename

This copies from the keyboard (CONsole) to the named file. Once you have typed the required contents, enter [Ctrl] [Z] on a fresh line and hit [Return]. This copies the contents on to the disk. Beware though – this command creates a new file, it does not copy into an existing one.

12. Floppy Disk Types

The different types of disk and disk drives used by the various types of IBM compatible can be confusing, thanks to the many different ways that a 5.25-inch disk has been formatted over the years, only a few of which have been used by IBM compatible machines.

A computer disk consists of a thin sheet of plastic coated on both sides with a magnetic material similar to that found on audio or video cassettes. Data is written onto these surfaces by read/write heads, very similar to the record/replay head on a cassette deck. However before data can be stored on the disk surfaces they have to be magnetically 'formatted', a process analogous to painting the white lines on a car park so cars know where to park.

The disk surface is first divided into a number of concentric rings, called 'tracks', which are further subdivided into 'sectors'. Any read or write operation involves at least one sector so these represent the smallest amount of data that can be read or written to the disk in one go. Both sides of a disk can be formatted if the disk drive has two read/write heads – one on each side.

The radius on which the first sector of each track

starts is marked by a hole through the disk which lines up with the index hole in the sleeve. A 'soft sector' disk can have as many sectors as the formatting software dictates. A 'hard sector' disk has a hole marking the start of each sector, so the number of sectors per track is dictated by the disk. Hard sectoring can pack in more sectors per track, but is not used on micro computers these days.

The format is important because disks can only be read by disk drive systems that can interpret the format. However soft-sectored disks can accept any format imposed on them, although formats that impose a high density of data require disks coated to particularly high tolerances. A disk that can be safely formatted to hold 1.2M of data will have no problems looking after 360K; but a disk that is only designed to hold 360K is bound to have problems storing 1.2M. The formatting process should identify problem areas on the disk surface and mark them as 'bad sectors' so the operating system knows not to store data there. However it is bound to miss some which is probably why you are having problems with some files.

Early PC floppy drive systems used a number of different formats before settling on the 360K format we all know and love. Up until recently 5.25-inch disks were always formatted with 40 tracks on a side, but early models only formatted one side of the disk, and only used eight sectors per track rather than the nine used now. Each sector has always held 512 bytes of data, so simple arithmetic gives capacities of 160K or 180K for single-sided formats with eight or nine sectors per track, and 320K or 360K for the corresponding double-sided formats.

The IBM AT introduced a new 5.25-inch format where the number of tracks on each side was doubled to 80 and the number of sectors per track increased to 15, giving a total capacity of 1.2M (work it out!). The PS/2 range introduced 3.5-inch 'microfloppy' diskettes, although non-compatible manufacturers had been using them for some time. These are also double sided and formatted with 80 tracks per side and either 9 or 18 sectors per track, giving a total capacity of either 720K or 1.44M.

The storage capacity of a floppy disk is described by its manufacturer in a number of ways. First of all disks can be 'single sided' or 'double sided' depending on whether one or both sides are guaranteed suitable for storing data – in practice many single sided disks can be formatted on both sides, although this is a dangerous practice as you run the risk of losing data.

The density at which data can be stored is rather a loose term and can cause much confusion. For 5.25-inch disks 'double density' describes those suitable for 40 tracks per side and up to nine sectors per track – the standard 360K format, in other words. This is usually qualified by the term '48 tpi' (tracks per inch). If

Disk Types and Prices

Diameter	Description	Formatted capacity	Reasonable price per disk
5.25"	Double Sided Double Density 48tpi	360K	80p
5.25"	Double Sided Quad Density 96tpi	not suitable	
5.25"	Double Sided High Density 96tpi	1.2M	£1.60
3.5"	Double Sided Double Density 135tpi	720K	£1.70
3.5"	Double Sided High Density 135tpi	1.44M	£3.60

Prices are per disk for well-known brands bought as a box of 10 and include VAT. These are not rock-bottom prices but might require shopping around, and larger orders should reduce the price per disk further.

you measure the length of the slot on the disk sleeve, through which the drive head reads the disk surface, you will find it is just over an inch long so should be well capable of taking 40 tracks.

So a disk marked '96 tpi' should be good for 80 tracks per side, and suitable for a 1.2M format? Unfortunately it is not as simple as that. A 'quad density' disk is indeed capable of taking twice as many tracks as a 'double density' disk, but is still restricted to around nine sectors per track. This means it is suitable for formatting to a capacity of 720K, but neither the PC nor the AT drives use disks formatted to this capacity. What is needed for a 1.2M format is a 'high density' disk that has a coating guaranteed for the required 15 sectors per track.

The story is different again for 3.5-inch disks, which inherently require a higher density rating because of their smaller size. Here 'double density' disks are suitable for formatting to 720K, while 'high density' are required for 1.44M. Because of the smaller diameter these disks are commonly rated at 135 tpi.

The table gives an indication of the sort of price you should expect to pay for floppy disks at the time this book was published. If you are asked to pay well over the odds for a disk it is probably because it is designed for the high density formats offered by some specialist disk drives. It will of course format quite happily in an AT or even PC drive, but is an unnecessary expense.

If you have a large number of double sided double density disks left over from your PC1512 days you can still use your AT to suitably format them. Use the command `format a:/4` on a 1.2M drive to format a disk to 360K capacity. Use `format a:/s/4` to format as a bootable system disk.

13. Binary, Decimal and Hex

Your computer's memory, or RAM, is made up of 524,288 (for a 512K model) or 655,360 (for a 640K model) discrete 'addresses', each of which is capable of holding an eight-bit 'byte' of information. This means that each address is itself made up of eight discrete locations, each capable of containing a '0' or a '1'. The whole address can hold a binary number from zero to 11111111, which is equivalent to the number 255 in our decimal notation.

Binary notation is sometimes called 'base-2', because each digit can be in one of two states, 0 or 1. Decimal notation is base-10, because each digit can be in one of ten states, 0 through to 9.

Binary notation is somewhat cumbersome, as even quite small numbers need lengthy strings of 1's and 0's; but decimal is not that useful as a way of expressing the contents of computer memory as there is no simple relationship between base-2 and base-10 arithmetic. A more direct relationship is to be found with hexadecimal numbers ('hex' for short) to base-16, with the 16 states symbolised using 0 through to 9 as with decimal, plus the letters A through to F to represent the hex equivalent of 10 to 15.

A lengthy binary number can be made more legible by splitting it into groups of four bits – in much the same way as we split decimal numbers into thousands and millions using commas (as in 655,360 for example). 1111 in binary is equivalent to 15 in decimal, or **Fh** in hex (note the notation used to indicate hexadecimal numbers); 10h is equivalent to 16 in decimal, and 0001 0000 in binary: you can see that hex is better matched to binary than the decimal

14. Lost memory

Many people are puzzled to find that BASIC programs, or text documents created by many word processors, are limited to 64K in length, despite having 512K to 640K of memory in their machine. This is down to one of the classic drawbacks of the IBM PC design, or more accurately the architecture of the Intel 8086 microprocessor: segmented memory.

The simplest way to check the overall memory fitted to a machine is to enter `chkdisk` at the MS-DOS prompt (after putting your MS-DOS boot disk in drive A if you do not have a hard disk machine). This will show something like '524288 bytes total memory' on a 512K machine, or '655360' on a 640K system. Alternatively the boot-up message usually tells you how much memory you've got. If you are running MS-DOS version 4 you can use the rather quicker `mem` command which just shows memory use and so doesn't need to read the disk.

Assuming the machine is OK, why the problem with these programs? It is because the amount of memory reported by, say, the BASIC command `FRE`, is limited not by the total size of your RAM, but

by the size of a single 'segment' of memory. The 8086 chip sees memory in 'chunks' of 64K, called segments. Although these segments aren't rigid compartments, it does make programming a lot easier (and often makes programs run a lot quicker) if all the data used is held within a single 64K segment.

As a result you constantly come up against the situation where a program is limited because it reserves just one segment for data. The `FRE` command returns the amount of space left within the single segment used to hold your program; it is unaffected by the number of 'empty' segments floating around elsewhere in the machine as these are not available to you.

If you have just expanded from a 512K machine to 640K, this can be frustrating. However, you can use the extra memory for is to increase the size of a RAM disk, and to use the GEM snapshot facility to the full. Many programs will make full use of the extra memory, especially word processors and spreadsheets; and you can also load memory resident 'pop-up' programs such as *Sidekick* into the extra space.

T H E
**HELP
SCREEN**
C O L L E C T I O N

19.



U S I N G M S - D O S

16. Losing COMMAND.COM

On a floppy drive machine some applications put up an 'Insert a disk with \COMMAND.COM in Drive A' message when you try to return to the operating system. This involves removing the program disk and inserting the operating system disk before you are allowed to exit back to the A-prompt. Usually this is caused by the program overwriting COMMAND.COM during operation. COMMAND.COM is the operating system's command line interpreter, so no A-prompt without it.

Fortunately there is a way of avoiding this problem, provided you have set up a RAM disk big enough to hold COMMAND.COM. Simply include the following lines in your AUTOEXEC.BAT to keep COMMAND.COM safe from harm and tell MS-DOS where it is:

```
IF NOT EXIST c:\command.com copy \command.com c:\
SET comspec=c:\command.com
```

17. Fatal Stack Errors

Certain programs sometimes give a 'Fatal Stack Error' or similar message, usually after you have done a couple of hours' work and then try to save it to disk! As the message implies the error is fatal and there is nothing you can do except re-boot the machine.

This message is not usually caused by bugs in the

software being run: it occurs because Microsoft decided to save a bit of memory when writing MS-DOS version 3.2 by cutting down on the stack space reserved. The solution is to add an additional line to your CONFIG.SYS file which says **stacks=64,128**. This undocumented command restores the stack space to the value used in earlier versions of DOS.

Even with this increased memory you can still eventually generate the error on some Amstrad PC1512's (but not for some odd reason on all of them) by turning the volume control down to zero and leaning a book on the keyboard so that it depresses a key.

18. Enough Buffers

Many people are not aware of the influence the BUFFERS parameter in the CONFIG.SYS file has over processing speed. With this in mind one of our readers (a C.J. Drouin) set up a series of tests, based mainly on file copying, each time altering the BUFFERS parameter. This produced the following results:

BUFFERS=1	24.11 seconds
BUFFERS=5	13.90 seconds
BUFFERS=10	11.54 seconds
BUFFERS=20	9.39 seconds

Anything above **BUFFERS=20** yields little difference in the processing time, but with each buffer only taking up 0.5k of memory. the trade off is well worth it.

19. MS-DOS External Commands

These are utility programs that are supplied with MS-DOS, but which have to be loaded from disk each time they are used.

DISKCOPY A: B:

Makes a track-by-track copy of the 'source' disk in drive A onto the 'target' disk in drive B, and formats the target disk if necessary. Hard disk users with just one floppy can use this format from the C-prompt; MS-DOS will use the same drive for both A and B disks, prompting you for the appropriate disk when required.

Diskcopy on its own uses the default drive for both source and target disks - again prompting when required.

MORE < filespec

Displays the contents of the specified file one screenful at a time, scrolling when you press [return]. The < 're-directs' the input of MORE to be the file specified. Wild cards can't be used.

command | MORE

Filters the screen output of the command sequence so it is displayed one screenful at a time, scrolling when you press [return]. The | 'pipes' the output of the command through the filter.

XCOPY sourcefile targetfile /options

Similar to COPY, but copies directory structures too and offers more options. Target can be omitted (it defaults to your current drive/directory), but you must specify at least drive, directory or filename of Source. *,* is the default filename for Source and Target. These are the more useful options:

- /A Only copies files that have the Archive attribute set (this attribute is used by some programs, most notably BACKUP, to mark files as already 'archived').

- /M As above, but turns off the Archive attribute afterwards.

- /S Copies files from sub-directories within the specified directory as well, retaining the directory structure. If /E is also specified this includes empty sub-directories.

- /W Waits for a key press before executing the command, allowing you to change disks if source file is not on same disk as XCOPY.EXE.

- /P Prompts for confirmation on each file before copying.

- /V Verifies each copy - has same effect as VERIFY ON.

- /D: dd-mm-yy Only copies files that have been changed on or after the specified date (date format depends on COUNTRY command). Should be last in option list, and remember that this only works if you your clock is set properly!

20. Write Protecting Files

It is often a good idea to write-protect essential MS-DOS files like AUTOEXEC.BAT, CONFIG.SYS and COMMAND.COM, particularly as some installation routines insist on altering your configuration files without telling you. These three files will usually be found in the root directory (although it is possible for them to live in another directory). To write-protect them use the commands:

```
ATTRIB +R AUTOEXEC.BAT
ATTRIB +R *.SYS
ATTRIB +R COMMAND.COM
```

If you do later need to modify any of these files you can remove the read-only attribute by using the command `ATTRIB -R filename`.

21. Editing at the DOS Prompt

How many times have you typed a command at the DOS prompt and got it slightly wrong? That mistyped filename or forgotten backslash won't be half so infuriating if you learn how to edit and re-use your last command.

There are two different ways to bring up the last command you entered: you can either restore it in its entirety by hitting [F3], or you can bring it up a few characters at a time using the left and right cursor keys.

PRINT filespec1 filespec2... /options

Sets up a 'print queue' for background printing of the specified files. Printing is slower than usual, but once the queue is set up you can get on with a 'foreground' task. The filespecs can contain wildcards.

The first time PRINT is run the options specify how the command is to be used:

/D:PRN Sets the output device to that connected to the printer port. If you have a serial printer use **/D:AUX**.

/B:4096 Sets a 4K print buffer. The default, if this option not used, is 512 bytes.

/Q:32 The number of files that can be in the queue. The default is 10 and the maximum 32.

Once the queue has been set up it can be modified by specifying one of these options after each filespec:

/C Removes this and following files (until a **/P** is encountered) from the queue.

/P Add this and following files (until a **/C** is encountered) from the queue.

The queue can be cancelled and the paper advanced to the next page with **PRINT /T**.

Suppose you type `COYP A:FILE.TXT PRN` when you mean `COPY A:FILE.TXT PRN`. MS-DOS'll give you a 'Bad command or filename' message and dump you back at the prompt again, because it doesn't know of a command called `COYP`. You could hit [F3], use the delete key to backspace as far as the error and then type the rest of the line properly, but you might just as well re-type the whole line from scratch. A better method is as follows:

1. Hit the right cursor key twice to bring up `CO`;
2. Type the `P` and `Y` in the right order this time;
3. Hit [F3] to bring up the rest of the original line.

Once you've got the hang of that, try using [Ins] and [Del]. These get you round situations where you've got missing or unwanted characters in the original line. For instance, to change `COPY A:FILE.TXT PRN` into the command `COPY A:\FILE.TXT PRN`, you could:

1. Use the right cursor key to bring up `COPY A:`
2. Hit the [Ins] key;
3. Type in a backslash [`\`];
4. Hit [F3] to restore the rest of the line.

Notice the effect [Ins] had: normally the backslash would have replaced the `F` of `FILE.TXT`, rather than squeezing in before it as here. If you have a Microsoft-compatible mouse installed on your system, things are even easier. Moving the mouse to the right has the same effect as rapidly hitting the right cursor key, so you can bring up long sections of a line with one quick sweep of the little beast.

22. Directory Shorthand

If you do a DIR of a hard disk subdirectory you will find the following two entries at the very top. The first, the single stop, represents the current directory while the second, the double stop, represents the parent directory – the one above the current directory on the directory tree:

Volume in drive C has no label
Directory of C:\NW\EMPTY

```
.          <DIR>      25-08-88   3:18p
..         <DIR>      25-08-88   3:18p
```

The double dot is commonly used with the Change Directory command, as in `CD ..`, but less well known is that these directory entries can be used in other ways too:

copy . b: copies all the files in the current directory to the disk in drive B;

del . deletes all the files in the current directory;

dir .. shows a list of files in the parent directory. In much the same way, the backslash \ indicates the root directory, so **del** is equivalent to **del*.***.

23. Using CHKDSK and RECOVER

The way files are stored on disk is complex, and can go wrong. In particular the 'File Allocation Table' at the start of the disk holds information about the files without which MS-DOS cannot access the data – even though the files themselves may be in perfect condition. The CHKDSK utility, together with RECOVER, can repair such damage, but should be used with great care.

Files or disks are only recovered in the sense that they are made readable by MS-DOS. They usually need further surgery as chunks may be missing. Program files can rarely be recovered enough to be usable.

CHKDSK *filespec* /F/V

This performs several functions:

Reports on the allocation of memory on the disk specified (or the current disk) and in RAM. In particular it indicates the amount used by hidden files, and the amount of free space on both disk and RAM.

Indicates the presence of 'lost clusters', or data on the disk that MS-DOS cannot link to specific files. If the /F option is specified these are converted into files called **FILEnnnn.CHK** which MS-DOS can read.

Indicates the presence of 'bad sectors', or areas of the disk that cannot be used to store data. There shouldn't be any on a floppy disk, but up to one or even two percent of a hard disk is acceptable. Any more and the disk should be re-formatted.

filespec is optional, and can contain wildcards. When used the files are checked for 'non-contiguous blocks'. These occur when files have been deleted and copied many times. They do not usually cause problems, but if there are a large number the files affected should be copied to a fresh disk. **CHKDSK *.*** is the best way of finding out how many files are affected.

/F Errors are only actually fixed when this option is used. This may cause some data loss, depending on the problem, but may be the only solution. When this option is specified CHKDSK can make some attempt to repair a damaged File Allocation Table, as indicated by error messages like 'Disk Error reading FAT'.

/V displays the name of each directory entry as it is checked, including hidden files. Possibly the best way of finding out what is on a disk.

RECOVER *filespec*

A very dangerous utility that recovers the file specified

if it contains bad sectors by reading it sector by sector, skipping the bad sectors and noting them in the directory area to prevent future use. Only use on data files as it cannot recover lost data, just render the file readable by MS-DOS. You have to fill in the missing data by hand.

If you type the command RECOVER without giving a filename you are telling MS-DOS to completely wipe-out all the entries in a directory, naming files **FILEnnnn.REC** as it goes, regardless of their original contents. Try it on a floppy disk full of data that you don't want to see what this means.

Imagine what would happen if you used it on a hard disk! Especially since all your sub-directories become files and any files that were in them attempt to go back into the root directory. Since there is a maximum limit to the number of files that there are in a root directory (usually 512 on a hard disk) the loss of data could be catastrophic! RECOVER is best deleted off your system altogether.

24. Filename Priority

When you enter a command at the MS-DOS prompt, such as **myfile**, MS-DOS first checks the current directory for a file called MYFILE.COM, then for a file called MYFILE.EXE and finally for a file called MYFILE.BAT. If a file called MYFILE.COM exists it is automatically run, which means that any files called MYFILE.EXE or MYFILE.BAT are ignored. As MS-DOS also ignores any extension in the command line, even if you were to enter **myfile.bat** it would still result in MYFILE.COM being executed – there is just no way of running MYFILE.BAT while either MYFILE.COM or MYFILE.EXE are in the same directory.

The situation is further complicated by the PATH command. If PATH is set to, say:

C:\;C:\MSDOS;C:\PROGS

Then, if neither MYFILE.COM, EXE or BAT can be found in the current directory; MS-DOS will carry out the same search in the root directory, followed by the \MSDOS directory and the \PROGS directory before giving up and reporting 'Bad command or file name'. This means that the program \MSDOS\MYFILE.BAT would be loaded in preference to the program \PROGS\MYFILE.COM. Funny old world, isn't it?

25. Checking RAM Disk Size

For people with no memory, this micro-tip might be useful. If you can never remember how big a RAM drive you've set, and using DIR when the RAM drive is empty provides no answer, put the following line in your AUTOEXEC.BAT file:

MD C:\SHOWSIZE

This creates a subdirectory called SHOWSIZE, which

is enough to make DIR show you how much free space there is. As a side effect, it also records the time you last booted the system, should that ever be of the slightest interest.

26. Listing Sub-directories

A directory name follows the same format as a filename under MS-DOS – eight characters for the name itself and three for the extension – but most people forget about the extension altogether. The command `dir *` will list all extension-less entries in the current directory so, provided you always give files some sort of an extension and never give sub-directories an extension, what you get is a nice clean list of subdirectories.

An alternative is to use the command

```
dir | find "<"
```

as this displays only those lines of the directory listing that contain the left angle bracket <. All subdirectory entries contain this as part of their <DIR> indicator. However, since the < symbol is used for input redirection it can't form part of a filename, so it won't crop up in any other directory entries. There are two points to note here.

First, thanks to the quotes surrounding it, MS-DOS doesn't treat < as a redirection symbol. Second, a subdirectory listing produced with either method will still contain entries for . (this directory) and .. (the parent of this directory). One solution is to use the line:

```
dir | find "<" | find /v "."
```

This further filters the directory listing to exclude entries containing the . symbol. Ironically enough this ends up ditching subdirectories with extensions, one of the problems with that `dir *` method we started with.

27. Problems with Concatenation

A problem can often occur when files are concatenated (joined together) using a command such as `copy file1+file2+file3`, which would create a new FILE1 containing the contents of all three files. When the new file is viewed with certain text editors, or using the MS-DOS command TYPE, it can look as though the concatenation has not occurred, although in reality it has.

The problem is caused by the fact that some text editors puts an End of File marker (ASCII code 26, or Control-Z) at the end of the file when the file is saved, while MS-DOS usually does not. This is because CP/M, the eight bit operating system on which a lot of MS-DOS was based, always marked the end of a file

with a Control-Z marker. MS-DOS is not so fussy as it makes more use of the note it keeps in the directory itself telling it the length of the file.

However the TYPE command does take notice of the EOF marker, and stops TYPEing when it encounters one. Try entering this instead:

```
COPY file1 /B CON
```

The option /B labels the file as a 'binary' file, so any EOF markers are ignored. If you want to print the file, try the command:

```
COPY file1 /B PRN
```

29. Mystery Files

A useful command for putting your disk directories into alphabetic order is `dir|sort`, but enter this command and you will find it has an odd side effect: two extra files show up in the directory that weren't there under DIR alone, and they're stamped with today's date and time.

These mystery files are a product not of SORT itself, but rather of the DOS piping facility invoked by |. Piping is really just a special type of input/output redirection.

Redirection is a way of changing where a program gets its input from or sends its output to. Input redirection with the < symbol can be used to make a program read input from a file rather than from the keyboard. Output redirection with > can be used to make a program send its output to a file or printer, rather than to the screen. So `debug < listing.txt` runs DEBUG, feeding it the contents of the file LISTING.TXT as if you'd typed them in at the keyboard. The command `dir >contents.txt` copies a listing of the current directory into the file CONTENTS.TXT.

28. Deleting Sub-directories

This can be a tedious business as first you have to empty the sub-directory with `del green\*.*` and then `rd green` to actually kill the thing off. It's all a bit of a slog if you've got five or six directories to wipe out.

The smart way to do this kind of job is with the FOR command. Type the following at the MS-DOS prompt to erase all the files in sub-directories GREEN, WHITE, BLACK, BROWN and RED:

```
FOR %p IN (green white black brown red) DO del %p\*.*
```

Now you can remove the sub-directories with:

```
FOR %p IN (green white black brown red) DO rd %p
```

These may look like fiddly lines to type in, but bear in mind that the second line is almost identical to the first, so with a touch of [F3] and a little backspacing you can avoid typing the FOR...DO bit a second time.

This powerful DOS facility has spawned the class of programs known as filters, which are designed to have their input and output redirected, and are very rarely used any other way. SORT is a good example of a filter, sorting its input alphabetically, line by line. This is not a lot of use if its input comes from the keyboard, but is very handy indeed if you redirect the input from a text file. Thus

```
sort <file.txt
```

displays the contents of the file FILE.TXT on screen in sorted form, while

```
sort <garbled.txt >tidy.txt
```

sorts the contents of the file GARBLED.TXT into TIDY.TXT.

That's all very well for simple tasks, but try using this technique for producing a sorted directory listing. One way would be to use two separate commands as:

```
dir >temp.txt
```

```
sort <temp.txt
```

This feeds the output of DIR into the file TEMP.TXT, and then uses this as the input for SORT. A more convenient method is to use 'piping', a form of redirection that connects the output of one program to the input of another. We can replace our earlier attempt:

```
dir | sort
```

Piping was originally conceived for multi-tasking operating systems such as Unix, where the two operations, **dir** and **sort** in this case, could actually be carried out concurrently. MS-DOS can only start the **sort** operation after the **dir** has finished, so it feeds the output of DIR into a temporary file, and then uses that file as the input for SORT. There are two important differences though: first, it chooses the temporary file name itself so as not to clash with a file of yours; and second, it automatically deletes the temporary file afterwards.

That explains what those two files are – temporary storage files for the information that's being piped – and why they always have the current date and time. What it doesn't explain is why MS-DOS should create two files when it ought only to need one.

30. Mixed Drive Types

One of the problems of combining disk drives capable of taking two different types of floppy disk is that you can no longer use the command **diskcopy a: b:** to copy the contents of a disk from one drive to the other. This is because DISKCOPY copies data byte by byte as it is stored on the disk, and the directory and FAT structures, which MS-DOS uses to find files on the disk surface, occupy different areas of the disk between, say, a 360K 5.25-inch disk and a 720K 3.5-inch disk.

You can, however, use the XCOPY utility as in **xcopy a: b: /s**

to copy the entire contents of the disk across, subdirectories and all (that's what the /s bit does). This will only work if MS-DOS can find the XCOPY program in the current directory or on the directory path.

31. Losing Long Files

Take a diskette, put it into drive A and create a subdirectory on it (called TEMP, for the sake of argument). Copy some files into it, including one large file of 100k or so. Then log into the subdirectory using **CD \TEMP** and type **COPY A:\TEMP**. DOS will attempt to copy all the files, but when you do a DIR you'll find that the large file is now only just over 64K in length.

So what's happening? Normally MS-DOS won't allow you to copy a file on top of itself, but for some reason the particular syntax above isn't trapped. COPY uses a 64K buffer – if you copy a larger file you can see it taking 64K chunks of the file – so the problem arises as follows.

DOS reads the first 64K from the file, and then re-opens the file to write to it. Having written that 64K, it then tries to read the next 64K. By that time however the file has been opened and the directory entry changed, so DOS can't read any more of the file.

The cure for the problem is hard to find. The data is still on the disk, and as long as you don't copy anything at all to any part of the disk, it should be possible to get it back, provided you know what you are doing. It's an easy mistake to make, though!

32. Directory Structure

A floppy disk consists of a very thin piece of Mylar, a very tough kind of plastic, covered in a thin magnetic coating. Before the computer can use such a disk to store data it has to organise the magnetic layer into a format that it can recognise and operate with. This is done when you run the MS-DOS utility FORMAT.

The computer divides the magnetic material into a number of very narrow 'tracks', 40 on each side of a 360K disk, fitted into the 0.83 inch width of the coated area. The tracks are numbered 0 to 39. The tracks are then sub-divided into 'sectors'. Each track of a 360K disk is divided into 9 sectors, numbered 0 to 8, so that there are a total of 360 sectors on each side of the disk. Each sector stores 512 bytes of data, hence the capacity of the disk.

However all that capacity is not available to you, as the computer reserves 6K for its own use. The first sector (0) is reserved for the 'boot sector', which contains details of the disk format and a short program to 'boot' the operating system. Sectors 1 to 4 are used for the 'file allocation table', or FAT for short, which

keeps a record of what areas of the disk are used for storing each file (this area actually holds two copies of the FAT in case one gets corrupted or damaged). The next seven sectors, 5 to 11, are used by the 'root directory' table.

This doesn't contain the actual files but rather the address where each one starts on the disk, its attributes, its length and one or two other details. This information takes up 32 bytes per file, and as the root directory table occupies just seven sectors this means you can only have 112 files in the root directory of a floppy disk, as $512 \times 7 / 32 = 112$. You can only have 512 files in the root of a hard disk for basically the same reason.

You can get round the problem by using sub-directories, which the computer treats as a special kind of file storing the directory table for that sub-directory, using exactly the same format as the root directory. Sub-directories are stored in the main data area of the disk, which means they do not suffer the 112 file limitation.

33. Formatting Problems

It appears that some ATs with mixed floppy disk drives, specifically a 1.2Mbyte 5.25-inch as drive A and a 720K 3.5-inch as drive B, generate an error when the command `format b:` is used. Try `format b: /n:9` which specifies nine sectors per track, instead.

34. MS-DOS Version 3.3

These are the main differences between version 3.2 and version 3.3 of MS-DOS, as they affect the user (you can check which version you're running by entering `ver`):

- The main reason for a new version was to support the 1.44Mbyte high density 3.5-inch floppy drives used by IBM's PS2 range.
- The command `FASTOPEN` was added. This tracks the location on hard disk of frequently used files and directories, and is perhaps the biggest innovation for hard disk users.

Normally, if you ask MS-DOS to find a file in a sub-directory it first checks the root directory for the appropriate sub-directory entry, then it looks in the sub-directory file to find out where the file is stored (assuming it doesn't have to find another sub-directory as it would with, say `C:\WORK\LETTERS\WRIT.001`), then it can go to the file itself. This is why your hard disk sometimes sounds as though it is doing a tap-dance!

The silly thing about MS-DOS is that it even does it if you refer to a file in the current directory. The only

reason that this is not ridiculously slow is because MS-DOS keeps the most recently accessed disk sectors in memory in buffers, as set by lines like `buffers=20` in your `CONFIG.SYS`. Provided you have a reasonable number of buffers you will notice that the first time you do a `DIR` command MS-DOS will access the disk, but if you repeat the command the disk light will not come on – MS-DOS is reading the directory from its buffers.

But disk sector buffers are an inefficient way to store directories: the buffering process is being done by the low level part of MS-DOS which has no idea whether that sector contains a directory entry that is going to be frequently accessed, or a data file which will just be read once. So if you copy a moderately large data file (even 50 buffers is only 25K of storage) all your buffers get filled up with this file and MS-DOS has to read the directory entries again.

`FASTOPEN` sets out to cure this by holding the directory entries for the most recently accessed sub-directories and files. It must be run once (usually in the `AUTOEXEC.BAT` file) with a line like:

```
fastopen c:=100 d:=50
```

This specifies the number of entries to hold for each hard disk (each entry takes 35 bytes). It can only be used on fixed disks, as changing disks would cause it too many problems, but then fixed disks are more likely to have deeply nested sub-directories.

- Any command in a batch file starting with the `@` character is not echoed to screen. This means you can get rid of that irritating display of 'ECHO OFF' at the start of a batch file by using `@echo off` instead.

- The `CALL` command provides proper nesting of batch files, without the messy business of loading a separate copy of `COMMAND.COM` each time. If you have a file called `BATCH1.BAT` containing:

```
call batch2
call batch3
```

and so on, each `CALL` will execute each specified batch file in turn and then return to `BATCH1` – quite an improvement for modular batch writers.

- Code Pages have been introduced as a system to provide different character sets for use in different countries (see elsewhere for full details).

- `KEYB` is the MS-DOS 3.3 replacement for `KEYBUK` and its other national equivalents. It uses a file called `KEYBOARD.SYS` to hold details of all the different possible keyboard layouts, each being selected by the letters after the command name – in our case `UK`. If you like to keep your root directory uncluttered and not have `KEYBOARD.SYS` in the root directory, you will

get an error message unless you also add the full path and filename to the end of the command line:

keyb uk, 437, c:\msdos\keyboard.sys

As 437 is the default code page, this command can be shortened to:

keyb uk, , c:\msdos\keyboard.sys

- The APPEND command does for data files what the PATH command does for program files, however it seems only to work with commands or programs which read a file and never write to it. For example it works with the TYPE command but not with RPED, DEL or even COPY. If you want to use this command you will have to experiment to see which of your programs it works with.

- Since early days MS-DOS has imposed a limit of 32 Mbytes on hard disk size – if you wanted to use a larger disk you had trick MS-DOS into thinking that the disk was in fact two separate drives. However the FDISK program of version 3.3 allows you to define more than one MS-DOS partition on a disk. Each partition still can't be more than 32 Mbytes long (Version 4.0 removes that limit too) but the remainder

of the disk can then be defined as if it were a separate disk. See the 'Using Hard Disks' chapter for further details.

- Version 3.3 includes completely re-written BACKUP and RESTORE utilities, which appear to be incompatible with those of version 3.2. This means that the only way to restore files backed up under version 3.2 is to make up a special version 3.2 boot disk, complete with the version 3.2 RESTORE program, and boot the machine from that to restore the files. Once restored they are perfectly readable under version 3.3. Thankfully versions 3.3 of these programs are considerably faster than the older versions.

36. Upgrading to MS-DOS 3.3

On the Amstrad PC1512 and PC1640 several problems arise if you decide to upgrade from MS-DOS version 3.2 to 3.3. You will find that the mouse buttons, the extra keys [Del->] and [Enter], and the joystick socket attached to the keyboard will not work. This is because they are all non-standard features, the mouse buttons also being routed through the keyboard so that they appear to the computer to be extra keys. They therefore won't work without the KEYBUK.EXE program supplied with Amstrad's MS-DOS 3.2. That

35. Using Code Pages

This is a very simple idea which someone at IBM managed to make very complicated in practice. Code pages were introduced with MS-DOS version 3.3 as a means of providing different character sets for use in different countries.

Before the IBM PC different countries each used their own national variations of the ASCII character set. IBM, however, tried to design an extended character set that encompassed all the international variations. Unfortunately they forgot some characters that, although not very widely used elsewhere, were essential to countries like Portugal, Denmark and Norway, leading these countries to start modifying the hardware character sets on their computers.

Now IBM has introduced a way of doing this in software called Code Pages. The old standard character set has become known as Code Page 437, although it remains the default; while the most common alternative is code page 850, a new multilingual character set which replaces some of the less used double box lines and also most Greek letters with various accented characters.

The idea is that if someone sends you a disk file created using a different character set, such as Norwegian, you can switch the code page to the correct character set and read the file as it would appear on a Norwegian PC. There are five code pages supported in all:

437	United States
850	Multilingual
860	Portuguese

863	French Canadian
865	Nordic

Version 4.00 adds the option of choosing keyboard layout by specifying a Keyboard ID. Most countries have one specific keyboard layout, so this parameter need not be specified. A few countries, however, can use one of two layouts, in which case the Keyboard ID can be specified to select which layout is to be used. If it is not specified the layout defaults to the first in the list below, so in practice you rarely need worry about Keyboard ID:

France	120,189
Italy	141,142
United Kingdom	168,166

Screen controllers and printers normally hold character sets in ROM, so you can only use code pages if your hardware supports software-downloadable characters. When you switch to a code page other than the standard one, the software drivers DISPLAY.SYS and PRINTER.SYS download details of the altered characters to a suitable display adapter or printer.

At present only EGA and VGA screens can support downloadable characters. Ordinary monochrome screens do not, while CGA screens only support it in graphics mode. The manuals suggest that the only printers to support code pages are IBM's but code pages can also be

KEYBUK could be used on MS-DOS 3.3 except that it says 'Wrong Version' when run.

Many MS-DOS utilities check version and won't run on any other version. This of course is very wise if hard disk formats have changed between versions (as they did between 2.1 and 3.0). In this case though the check can safely be patched using DEBUG. First, though we have to rename it, as DEBUG won't write out an EXE file. Enter this exactly as it is written:

```
copy keybuk.exe keybuk33
debug keybuk33
e b92
lf
w
q
ren keybuk33 keybuk33.exe
```

This routine changes the value of the byte at address 0B92 from hex 14 to 1F. KEYBUK33 can then be run instead of KEYB in your AUTOEXEC.BAT file.

37. Corrupted Disk Directories

Has this ever happened to you? You are using something like *WordStar* to edit a document, and your data disk contains many text files, several of them

important. After one editing operation – the creation, entry and saving of, say, VITAL.DOC – all this changes. A directory of the disk now shows VITAL.DOC, other files that didn't belong there, and none of the original documents. According to *Norton Utilities*, *PC Tools* or *File Rescue Plus* there are very few erased files on the disk, and they don't include any of the original documents. Needless to say, there is no backup of the vital data.

So what's happened? Aside from VITAL.DOC, all the files shown by DIR as being on the corrupted disk actually belong on another floppy disk. What's more, these files aren't actually on there at all. TYPE them and they come out as scrambled versions of the lost documents! What has happened is that the FAT (File Allocation Table) and root directory, vital areas which between them describe and map all files on the disk, have been overwritten.

Overwriting is almost always caused by changing floppy when your PC isn't expecting you to. When MS-DOS wants to alter a few selected bytes of information on a floppy, it doesn't directly write them to disk. Instead it copies the entire disk sector into memory; alters the values of the selected bytes in memory; and then writes the memory copy back out to disk. This final writing process completely replaces the original information in the sector.

used with some Epson models and presumably any printer which supports downloadable characters. The printer driver supplied with MS-DOS version 3.3 does not support 24 pin printers. The font information is held in Code-Page Information (CPI) files as follows:

EGA.CPI	EGA or VGA display
LCD.CPI	IBM Convertible LCD display
4201.CPI	IBM Proprinter, Proprinter XL or compatible
4208.CPI	IBM Proprinter X and XL24 (version 4.0 only)
5202.CPI	IBM Quietwriter III or compatible

To set up your system to use Code Pages requires some jiggery-pokery. Let's set up a PC to use the Nordic code page 865 with keyboard layout 155; the country code is 047 and the keyboard code is NO (gleaned from your MS-DOS manual). We start on the CONFIG.SYS file by including the lines:

```
country=047,865,c:\dos\country.sys
device=c:\dos\display.sys con=(ega,865)
```

Next comes the AUTOEXEC.BAT which needs to include:

```
nlsfunc c:\dos\country.sys
```

The 'National Language Support Function' command allows the Change Code Page command CHCP to be used, and specifies the file that contains country-specific information. The file need not be specified if COUNTRY.SYS is in the root directory.

```
mode con cp prep=((850 865) c:\dos\ega.cpi)
mode con cp select=865
```

The first MODE command prepares two code pages, 850 and 865, for use with CONSOLE, specifying EGA.CPI for the font information. The second command then selects the required code page for the console.

```
keyb no,865,c:\dos\keyboard.sys /id:155
```

Finally we run the keyboard driver specifying the keyboard code, code page, the file containing the keyboard layouts, and the keyboard ID.

The system is now set up as required, using code page 865. With this configuration you could use the command chcp 850 to switch to the multilingual code page, as this has already been prepared in the AUTOEXEC.BAT.

However, unless you have an EGA screen, a suitable printer and a pressing need to swap disk files with a Norwegian, we suggest that you forget all about code pages! In this case most of the above commands can be ignored, although your CONFIG.SYS still requires the line:

```
country=044,,c:\dos\country.sys
```

To specify the country code, while AUTOEXEC.BAT requires the line:

```
keyb uk,,c:\dos\keyboard.sys
```

Note the double commas where the code page number should be. As the code page has not been specified, the default 437 is used. If both COUNTRY.SYS, KEYB.EXE and KEYBOARD.SYS are in the root directory, you could just use good old country=044 and keyb uk as always.

There's a big assumption here, which is that you haven't swapped floppies in between. If you have, the last stage of the process destroys all existing information in the sector concerned and replaces it with data from the disk originally in the drive. If the sector concerned is a FAT or directory sector, this could mean that vital pointers to files are replaced by pointers that refer to files on a different disk entirely.

This kind of disaster happens something like this. VITAL.DOC is created with the first floppy in the drive, and the crucial sectors had already been read into memory. During editing, this floppy is replaced with the data disk we're concerned with. Finally VITAL.DOC is saved, so DOS alters its memory copy of the FAT and directory sectors, which come from the first disk, to reflect the existence of a new file. It then writes them out to disk, but it's now the wrong disk.

There's usually very little actual loss of data during an accident like this, but the loss of those vital pointers indicating where files were causes serious problems. In most cases the files have to be pieced back together by hand using a low-level recovery tool and some knowledge of what was in the files in the first place.

The same thing can happen when you're doing quite trivial tasks at the MS-DOS prompt. Suppose you have two disks, one write-protected and one not. If you try modifying the write-enabled disk, perhaps just to erase a few files, you get a 'write protect error' message, and you'll immediately realise that you've got the wrong disk in the drive. What's your first instinct? Swap disks and hit [R] to retry? It'd be a natural thing to do, but it would also kiss your disk's FAT and root directory goodbye.

MS-DOS only discovers the write-protect problem when the last stage in the modification process fails to work. Hitting [R] should make it go right back to the beginning and re-read the relevant sectors into memory, but sadly it doesn't. Instead it just re-runs the last stage, which this time succeeds, and overwrites the relevant sectors of the second disk with those from the first. Disaster has well and truly struck.

The moral of this tale is to think carefully before swapping floppy disks. *WordStar* has special commands for 'logging on' to a new drive or directory, and these should always be used if you change disks. As for MS-DOS, it's probably safer to press [A] for abort and start again!

One last word of warning. If this happens to you and you want to try and recover some of the files, make sure that the first thing you do is DISKCOPY the corrupted disk, and work on the copy rather than the original; that way if anything goes wrong you can always start again. Note also that the *Norton Disk Doctor*, the automatic trouble-shooter packaged with the latest version of *Norton Utilities*, would not be able to handle this sort of situation and could make things worse in its attempts to sort the disk out.

38. Undeleting Files

It is possible to recover files that have been accidentally erased using DEBUG, providing they are short enough to have been stored in just one 'cluster'. Longer files cannot easily be recovered due to the tortuous method that MS-DOS uses for keeping track of subsequent clusters; however there are plenty of file recovery utilities available, some even in the public domain. What we provide here is a method of restoring a deleted file that is less than one cluster long.

This technique also looks at the way the file directory works, and can be used to change quite a few of the file's attributes. Proceed with caution, though, as it is possible to do a lot of damage!

Files are stored on disk in groups of sectors called 'clusters'. The number of sectors per cluster depends on the disk format used.

Format	Sectors/ cluster	Cluster size (bytes)
5.25" 360K	2	1024
5.25" 1.2M	1	512
3.5" 720K	2	1024
3.5" 1.44M	1	512
Hard disk	4 or 8	2048 or 4096

When a file is created the following information is stored in the directory:

Offset	Description	Size (bytes)
0	filename	8
2	file extension	3
11	file attribute	1
12	reserved	10
22	time created	2
24	date created	2
26	starting cluster	2
28	file size	4

The addresses of the clusters that the file occupies is stored in a special area call the File Allocation Table (FAT). The starting cluster entry in the directory is also a pointer to the offset in the FAT. When DOS deletes a file the first character of the filename is changed to E5 hex, and the file's entry in the FAT is zeroed to indicate that the clusters are free for use.

FAT entries are extremely complex and restoration is really beyond the scope of DEBUG. However the first cluster can be recovered fairly easily, provided no disk write operations have been performed after the accidental deletion which might have overwritten the actual cluster:

1. First of all, make a backup copy of your original disk using the DISKCOPY command. This step is not

at location 01A0.

The DEBUG command **E** is used to enter data into memory. Enter **E 01A0** and the computer will respond by displaying the address and the byte it contains: **1A00:01A0 E5** (the **1A00** is the absolute address of the data segment being used in this example – it may be different in your case, but don't worry about it). To change this byte all you need to do is to enter the ASCII equivalent of the letter required. For example for 'A' type **41**, for 'B' type **42**, and so on.

If you cannot remember what the original letter was, no matter; any letter will do for the recovery. The **RENAME** command can be used to change the name later. When [Return] is pressed, the byte in memory location **01A0** is immediately replaced by the number

entered at the keyboard (**45** in our example).

Now that the repair has been made, the last step is to put the data back onto the disk using the DEBUG write command **W**:

W 100 0 5 7

Notice that the syntax of this instruction is almost identical to that of the (L)oad command except that now the data is being written onto the disk instead of read from it. The task should now be complete. Exit DEBUG by entering **Q**, and confirm that the file has now been restored with the MS-DOS **DIR**ectory command. ●

T H E
**HELP
SCREEN**
C O L L E C T I O N

31.



B A T C H F I L E S

39. The Batch Language

MS-DOS supports a simple programming language which allows you to automate many DOS procedures. A Batch File consists of a series of commands, each on a separate line, which are executed sequentially when it is run.

A Batch File is a simple text file, saved as **filename.BAT** and run simply by entering **filename** (without the extension) at the MS-DOS prompt. If saved with the special filename **AUTOEXEC.BAT** the Batch File is executed automatically after the machine is switched on or reset. Execution can be interrupted by pressing [Ctrl][c].

There are four types of command that can be used in a Batch file:

1. Internal MS-DOS commands.
2. A call to any program with the file type COM or EXE, provided the path is specified so the Batch File can find it.
3. A call to any other Batch File, although after execution control is returned to the DOS prompt, not the originating Batch File.
4. A Batch Command (all of which are internal to MS-DOS):

ECHO OFF stops the commands being displayed as they are executed.

Some commands, such as COPY, still display messages on the screen after ECHO has been set off, as in '1 file(s) copied'. These reports can be suppressed by re-directing output to the NUL device, as in:

```
copy a:file.* b: >nul
```

ECHO message Displays the message on the screen.

ECHO. Sends a blank line to the screen.

PAUSE Displays 'Press a key when ready...' and waits for you to do so. Allows the user to change disks, for example.

REM message Only displayed if Echo is on. Used by the writer for explanatory REMarks.

FOR %%p IN (a b c...) DO command

The *command* is executed several times, first with the parameter %%p set to the value a, then with it set to b, and so on. The parameter can then be used as part of the command, so:

FOR %%a IN (file1.* file2.*) DO COPY %%a C:
 is equivalent to executing two separate commands:
COPY file1.* C: followed by **COPY file2.* C:**

The parameter can be used to represent filenames, parts of filenames, calls to a program, or indeed any string of characters. If the values are filenames then paths cannot be supplied, so the files concerned must be in the current directory.

GOTO label

commands

:label

Transfers control to the line following the label, skipping the *commands* in between.

COMMAND /c batch2

This allows you to run *batch2* from within another batch file, and return control to the originating batch file afterwards. It does however load a second copy of COMMAND.COM off disk which takes time, and space in RAM.

IF EXIST filename command

the *command* is executed if *filename* exists on the current directory of the specified drive. Useful for testing if the right disk has been inserted after a PAUSE command. Wildcards are permitted.

IF ERRORLEVEL n command

Some programs, and MS-DOS utilities, return an exit code. The *command* is executed if the errorlevel is n or higher. When testing for a variety of errorlevels using consecutive IF statements, make sure they are tested in descending order.

IF "string1"="string2" command

Executed if the two strings are the same. The strings can consist of environment parameters, replaceable parameters or any group of characters. "" signifies an empty string.

Using an **IF NOT** instead of an **IF** in the above statements causes execution of the *command* only when the condition is false.

32.

40. A Simple Typewriter



An expensive word processor is all very well for writing *War and Peace II - Peace and Quiet*, but it is a bit excessive for writing short memos to your boss reminding him about your imminent rise: it takes longer to load the word processor than it does to write the memo.

What follows is a short batch file that turns your PC and printer into an electronic typewriter. Enter it in a file called TW.BAT and you can summon it by entering TW. It has disadvantages, in that it doesn't word-wrap and you have to get every line right before sending it to the printer - but it's free, so who's complaining?

```
ECHO OFF
CLS
ECHO Now in Typewriter mode - Ctrl-Z and [return]
ECHO to end and print text
ECHO -----
COPY CON PRN >NUL
CLS
```


REPLACEABLE VARIABLES

Parameters can be passed to a batch file from the outside by means of 'replacement variables'. MS-DOS allows for ten variables, called %0 to %9, to be passed in this way. So, for example, ARCHIVE.BAT could contain these two lines:

```
copy a:%1.* b:%1.bak >nul
copy a:%2.* b:%2.bak >nul
```

Running ARCHIVE.BAT with the line `archive beef pork` substitutes `beef` for %1 and `pork` for %2. the dummy variable %0 would be replaced with the word `archive` itself.

SHIFT

Discards the contents of the lowest replacement variable and moves the rest down one, so %1 becomes %0, %2 becomes %1 and so on. Can be used to cycle variables through a command, or as a way of exceeding the limit of ten replaceable variables by shifting an eleventh into %9.

For example, a rather more useful ARCHIVE.BAT could contain the lines:

```
:loop
copy a:%1 c:\bak >nul
shift
if "%1"="" goto end
:goto loop
:end
```

This can take any number of specified files as parameters and copy them from drive A into the \BAK directory of drive C. Note the use of the IF command to test for the end of the list.

Parameters can also be transferred to programs called by the batch file, so if one of the lines of WRITE.BAT is `ws %1`, then entering `write novel.doc` would execute `ws novel.doc`, and load your novel into *WordStar* ready for you to start work. This only works with programs that can accept parameters in this way.

ENVIRONMENT VARIABLES

These are more like the variables you find in Basic or whatever, in that they are stored in an allocated area of RAM. They are created with:

```
SET name=value
```

where *name* is the selected mnemonic and *value* is the string, pathname or filename that defines its current value. The *name* is always put into upper-case.

`SET name=` deletes the *name* from the environment.

`SET` on its own displays a list of the variables.

Certain environment variables have a special meaning to MS-DOS:

PATH tells MS-DOS where to look for a command file.

PROMPT is the current MS-DOS prompt string.

COMSPEC tells MS-DOS where it can find **COMMAND.COM**. For example, the GEM batch files

copy **COMMAND.COM** into the Ram drive C and then:

```
SET comspec=c:\command.com
```

Many programs over-write the area in memory used to store the command processor, so this is a useful way of protecting it.

41. Free Multi-Choice Utility

If you are a proud owner of *WordStar 1512*, then you may not know it, but you are also the proud owner of a handy little utility that you can use in your batch files. Hidden on the Installation Disk, Disk 1, is a file called **CHOOSE.COM**, occupying about 256 bytes. It does not appear to be used by the main program itself, but is used in the installation batch file to give those friendly multiple choice screens.

To use this program in your own batch files you must put a copy of **CHOOSE.COM** onto the disk that contains the batch file, otherwise it won't work. To demonstrate how it works, here is a little batch file that makes use of **CHOOSE.COM**:

```
ECHO OFF
:START
CLS
ECHO Select option:
ECHO.
ECHO [1] Red
ECHO [2] Blue
ECHO [3] Green
ECHO [X] Exit the program
ECHO.
ECHO Press appropriate numeric key...
choose X123
if errorlevel 4 goto GREEN
if errorlevel 3 goto BLUE
if errorlevel 2 goto RED
REM otherwise 1 is assumed
goto QUIT
:RED
ECHO You have a chosen red.
pause
goto START
:BLUE
ECHO You have chosen blue.
pause
goto START
:GREEN
ECHO You have chosen green.
pause
goto START
:QUIT
```

Store the program with whatever name you like, providing it has the extension **BAT** (for batch file). If you call it **COLOUR.BAT** it can be run by entering

COLOUR at the MS-DOS prompt.

A pretty useless program, admittedly, but it does demonstrate several useful batch file techniques. The program CHOOSE.COM is run on the line reading **choose X123**. The utility returns an exit code of 1, 2, 3 or 4 depending on which of the four parameters are selected – in this case the code 1 is returned if X is selected, and 4 if 3 is selected. You can use whatever characters you wish. The error code is detected by the lines **if errorlevel 3 goto BLUE** and so forth. This directs the batch file to jump to the flag :BLUE.

Remember that the program CHOOSE.COM is the copyright of the author of *WordStar 1512*. If you own this program you are at liberty to use it as you want, but you must not use it in any commercial fashion without the permission of the author. You could, however, use CHOICE.COM which you can create using DEBUG – see a later chapter in this book.

42. Multiple Work Disks

Those trying to run a large application on a machine fitted only with floppy disk drives may find it impossible to get all the files needed onto a single work disk: quite a common problem if you have a 640K machine and only 360K floppy disk capacity!

The solution is two working disks. The first, which we will call the SYSTEM disk, should contain the MS-DOS operating system, the files CONFIG.SYS, RAMDRIVE.SYS, KEYBUK.EXE and any other utilities you might need. The second, the PROGRAM disk, should contain the application files required to run the package. In addition, both disks should contain a copy of the following file labelled AUTOEXEC.BAT:

```
ECHO OFF
KEYBUK
VERIFY ON
ECHO Now put PROGRAM disk in drive A
:TEST
PAUSE
IF EXIST program.exe GOTO OK
ECHO Wrong disk!!
GOTO TEST
:OK
COPY program.exe C: >NUL
COPY AUTOEXEC.BAT C: >NUL
C:
ECHO Now put DATA disk in drive A
PAUSE
program
```

So that this batch file has a general application, we have labelled the program files simply **program.exe**. You should substitute the names of the files needed to run the problematic application, and adapt the batch

file should more than one file be needed.

This batch file is automatically executed when you boot up from your SYSTEM disk. The guts of its operation lie in the first PAUSE command, and the preceding screen prompt, which gives you time to swap disks in the middle of the batch file's execution. It is vital that identical copies of the batch file exist on both disks, as the computer refers back to the disk for each instruction. If it were not to find the batch file at all it would get very confused and let you know in no uncertain terms.

Note the test that is made to check that you have actually swapped disks. The file **program.exe** should be any file unique to the PROGRAM disk. The main EXE or COM file would do nicely.

Once you have swapped disks the batch file copies the relevant program files to the RAM disk, drive C. We have shown just one COPY command line, but you will probably need more. Then it copies itself to the RAM disk, passes command to the RAM disk, prompts you for the DATA disk (another disk on which you store the documents you are working on) and automatically executes the program – again, use whatever command is required to run your application here. It is of course vital that your CONFIG.SYS file should contain the line **device=ramdrive.sys nnn**, where nnn is the space required (in kilobytes) for all the program files together with the batch file itself.

This batch file is just an example, and can be adapted in a number of ways. For instance, there is a lot of wasted disk space on the SYSTEM disk that could be used to load memory resident programs such as *Sidekick* or *PC Outline*. These could be stored on your SYSTEM disk and run from the batch file before the disks are swapped over.

This is a handy technique, but always check that the program itself cannot be configured to run RAM resident anyway. This would mean that you wouldn't need to have the program files in the RAM disk at all as the program would be loaded completely into memory right from the start, making a RAM disk a waste of space. *NewWord* and *WordStar* can be configured in this way, as is shown later in the 'Applications' section of this book.



43. Finding Across Files

Here are two variations of a simple batch file that takes the same command parameters as the DOS command FIND, but accepts wild cards for the file specification. The first handles badly matched files specifications elegantly, The second is of the quick one-liner variety:

Variation One:

```
ECHO OFF
if exist %2 goto found
```

```
ECHO No files match specification
goto end
:found
for %a in (%2) do find %1 %a
:end
Variation Two:
if exist %2 for %a in (%2) do find %1 %a
```



44. Selective Delete

The following batch file, called DELB.BAT deletes all the files in a directory except those specified. It is used in the following format and allows as many files to be specified as the user wishes:

```
DELB file1 file2 file3...
```

If any of the files specified do not exist the batch file will echo the filename to the screen, leaving the decision to continue or abort entirely to the user:

```
ECHO OFF
IF "%1" == "" GOTO fin
MD DELB
:begin
IF "%1" == "" GOTO endprog
IF NOT EXIST %1 GOTO notexist
CD DELB
COPY ..\%1 >NUL
CD ..
:return
SHIFT
GOTO begin
:notexist
ECHO %1 does not exist.
GOTO return
:endprog
CD DELB
DIR
CD ..
DEL *.*
CD DELB
COPY *.* .. >NUL
DEL *.*
CD ..
RD DELB
ECHO Deletion Complete!
:fin
```

45. Checking Machine Details

There are a couple of useful little routines that come with the 1512, which are used by the installation batch files to provide information about the machine and could be useful in any batch file. On Disk 2 (the blue GEM startup disk) in the directory \AUTO is DRVNT.COM. This is used by the batch file AUTOCOPY.BAT (in the same directory) to return the

number of drives on the machine. TYPE the batch file to see how it works.

The 1640 comes with a routine called DISPCCHK.COM (in the \GEMSTART directory of Disk 1) which returns the display mode selected by the dip switches. It is used in by the AUTOEXEC.BAT file in Disk 1's root directory.

46. Batch File Menu System

It is perfectly feasible to create a menu system using just the MS-DOS batch system. First of all, create a batch file called MENU.BAT as follows:

```
CLS
ECHO 1) Wordstar
ECHO 2) Supercalc
ECHO 3) Mirror II
```

For each choice on the menu, create further batch files as 1.BAT, 2.BAT and 3.BAT. The batch file for 1.BAT might look something like:

```
CD WORDS
WORDS
CD \
MENU
```

On running MENU.BAT the screen clears and the menu is displayed. The drive prompt appears at the end of the menu, at which point you simply enter the desired number which then runs the required menu selection. When you quit the selected application, the main menu reappears. Put the command `menu` at the end of your AUTOEXEC.BAT file and you will automatically be presented with your menu as soon as you turn on your PC. The main disadvantage of this system is that it is rather slow, as it is continually loading batch files off disk. Each batch file takes up disk space too, which could cause problems on a based on floppy disks only. One way of speeding up at least the initial display of the menu is to put the menu text into a text file, called say MENU.TXT, and then put the line `type menu.txt` at the end of AUTOEXEC.BAT. This would have much the same effect but the TYPE command is rather quicker than ECHOing successive lines of text. An alternative would be to create the file MENU.BAT containing:

```
echo off
cls
type menu.txt
```

All in all this system is better suited to a hard disk machine though.

An alternative is to use a short utility such as CHOICE.COM that returns an errorlevel to the batch file which depends on the key that has been pressed. The advantage of this technique is that the whole menu system can be built up from one batch file, and COM files are considerably faster to execute than batch files.

CHOICE.COM is a utility that can be generated using DEBUG. To find out how to create it for yourself, see elsewhere in this book.

47. Setting RAM Disk Size

On an Amstrad PC1512 or 1640 the size of any RAM disk can be set from a utility called NVR (executed by entering `nvr` at the MS-DOS prompt). This program allows a number of parameters that are stored in the batter-backed 'non-volatile RAM' to be set from a simple menu system.

If the line `device=ramdrive.sys nvr` is present in your system's CONFIG.SYS then the size of the RAM disk is set to that held in non-volatile RAM. However, if you look in the sub-directory \GEMDESK of Master Disk 5 of a hard-disk PC1512 you will find a utility called NVRPAT2.COM. Type the file GEM.BAT and you will see what is happening. After transferring control to the GEMDESK sub-directory, the batch file runs NVRPAT2.COM. If this program returns an errorlevel of one, the message 'RAM Disk reduced to zero - effective next reboot' is echoed to the screen.

What NVRPAT2 does is read the non-volatile RAM area to see what size RAM disk has been set. If it is not already set to zero it sets the NVR for a RAM disk size of zero, and sets the DOS variable `errorlevel` to one. Despite the message it is only actually set to zero when you reboot your machine and the operating system encounters the appropriate line in your CONFIG.SYS file. This tells it to set the RAM disk to the size specified in the NVR, which NVRPAT2 has set to zero.

48. Checking the Syntax

One particular way use for the FOR command is to test the validity of parameters and control execution accordingly. For example, suppose you have written a batch file TEST.BAT which you want to operate in one of two ways, depending on whether it is executed with `test r` or with `test s`. The following could be handy:

```
FOR %a IN (r s R S) DO IF "%1"=="%a" GOTO %1
ECHO error message...
GOTO exit
:R
ECHO 'r' commands executed...
GOTO exit
:S
ECHO 's' commands executed...
:exit
```

What happens here is that the IF statement that makes up the second half of the FOR statement is executed four times, testing for the two valid parameters `r` and `s` in both upper and lower case. If a match is found control is transferred to the appropriate label using the GOTO command, which thankfully is not case sensitive). If not then an error message is displayed.

49. Free PPC Utility

There is a useful little utility to be found on the system disk of an Amstrad PPC portable, called DATETEST. It is used in the machine's AUTOEXEC.BAT to check whether the date is later than 1980. If it is it assumes that the real time clock is working and returns an errorlevel of 1, which causes the IF statement following to jump over the DATE and TIME commands. If it is not, then there might be something wrong with the batteries:

```
datetest
if errorlevel 1 goto signon
date
time
:signon
```

50. Restoring the Path

Some application programs require a particular path to be set before they are loaded, but it would be useful to be able to restore the old path when it is terminated. MS-DOS stores the path setting in an environment variable called PATH, so the following does the trick:

```
SET OLDPATH=%PATH%
SET PATH=c:\;c:newpath
ECHO Load program here
SET PATH=%OLDPATH%
```

This sets up a new variable OLDPATH which stores the value of the original path while the application is running. Notice that the percent marks effectively mean 'replace with the contents of this variable'.

51. Terminating Batch Files

A useful control code is code 3 or 'Ctrl-C', used to terminate execution of a batch file. The usual way of offering the user a chance to terminate a batch file is something like:

```
ECHO Press a key to continue
ECHO or Ctrl-C to terminate
PAUSE
```

Pressing [Ctrl-C] at this point generates the message 'Terminate batch job (Y/N)?' at which point you would press [N] to return to the MS-DOS prompt. The following technique smartens this up somewhat, and reduces the number of keystrokes required of the user:

```
ECHO ^C | DATE > NUL
```

This single line pipes the output of `ECHO ^C`, i.e. the value 03 represented by ^C, into the DATE program, while the output of DATE is re-directed to the NUL device.

To understand how this works, create a batch file that contains the single command **date** on one of its lines. When the batch file is run and execution reaches this command the present date is displayed, followed by an invitation to enter a new date. Instead you press [Ctrl-C] which interrupts execution of the batch file with the message 'Terminate batch job (Y/N)?'. This is the essence of the technique, and all the fuss about the date can be lost by re-direction to NUL, leaving just the termination message. The user has only to press [Y] or [N] to terminate or continue with the batch file. We use the **DATE** command simply because it takes user input and doesn't do any damage; **TIME** would have done equally well.

All well and good, but try creating this batch file using **COPY** and you run into problems: as soon as you press [Ctrl-C] the **COPY** command is aborted and all is lost. This is a tricky problem which can only satisfactorily be overcome by using a word processor that allows direct entry of the first 32 ASCII characters. *WordStar* owners, for example, can generate such characters by pressing [Ctrl-P] followed by the appropriate letter (the [C] in this case), but must remember to edit in 'Non-Document' mode. Others allow you to hold down [Alt] and tap out the value as a three-digit number on the numeric keypad, 003 in this case. If you are not one of the lucky ones you will have to resort to the MS-DOS utility **DEBUG**; not a very satisfactory solution.

52. Providing Instructions

With a little ingenuity replaceable variables can be extremely versatile. For example, if you are creating batch files that you use only occasionally, or creating them for other people, it might be useful to provide instructions. You could of course simply put a series of **ECHO** statements at the start of the file, but this would be annoying for people who don't need them. The following technique allows users to call up instructions when required:

```
ECHO OFF
IF NOT "%1"=="?" GOTO main
ECHO This batch routine shows
ECHO how entering '%0 ?' can
ECHO be used to provide help
GOTO exit
:main
ECHO main routine...
:exit
```

This means that entering **batch** with its correct parameters gets on with the executing main batch file, while entering **batch ?** lists the instructions. Note the use of %0 which is replaced with the name of the batch

file itself. This means you are free to change the name of the batch file without compromising the instructions.

54. Changing Screen Colours

The **ANSI.SYS** driver, which provides an 'easy' way to change screen colours, requires you to send it an escape character (CHR\$(27) in BASIC parlance) as the start of a command. This is not easy as many text editors won't let you type an ESC character, so it's hard to build the necessary ANSI string into a BAT file.

However, you can do it fairly easily using the **PROMPT** command. This causes MS-DOS to change the system prompt from **A>** or **C>** to the new string you specify. This string is displayed on the screen every time MS-DOS looks for a command, so if it includes the necessary ANSI codes they will be picked up and acted on by **ANSI.SYS**, whose job it is to monitor all screen output looking for the magic escape sequences.

Including the characters **\$e** in the **PROMPT** string generates an escape character, so for example putting **PROMPT \$e[33;44m** would set the screen to yellow text on a blue background. A full list of the ANSI codes is usually to be found in your computer manual. A batch file using this technique would look something like this:

```
PROMPT $e[33;44m
ECHO ON
ECHO OFF
PROMPT
```

The **ECHO** statements are needed in order to give the **PROMPT** string a chance to be displayed, while **PROMPT** on its own sets the prompt back to **A>**. Remember that you must have the line **DEVICE=ANSI.SYS** in your **CONFIG.SYS** file to make these escape codes work.

Incidentally, because **ANSI.SYS** monitors all keyboard and screen activity, it can slow the machine down somewhat – real speed freaks tend to discard it.

53. Fancy Prompts

The famous **A**-prompt performs a function, it must be admitted. It tells you that you are logged on to Disk Drive A, but that's about all. But you can tailor the prompt to do a lot more than that – to tell you the time, which sub-directory you are logged in to, and a variety of other information. This is a handy prompt that you might like to include in your **AUTOEXEC.BAT**, or put it into a separate batch file labelled **NEWPRMPT.BAT**; but it requires **ANSI.SYS** to be loaded:

```
prompt $e[s$e[1;1H$e[K $e[7m $d $t $p $e[0m$e[u$n$g$e[0;59;"dir /w";13p
```

NEWPRMPT.BAT should be entered as a single line. Note that there are three spaces between the **\$d** and **\$t**, and three between **\$t** and **\$p**.

Once you've created this batch file enter **NEWPRMPT** and stand back in amazement. Try pressing [F1] too...

55. Generating Control Codes

Control codes – those ASCII characters with a code between zero and 31 – can perform a variety of useful functions when sent to computer screen or printer. However getting them into a batch or text file is not always easy, as MS-DOS and many application programs use many of the more useful codes for their own purposes. The ESC character ASCII code 27, used by ANSI.SYS and most printers for special effects, is particularly problematic. Here, however, are a number of methods that have proved successful:

1. If you hold down [Alt], hit three number keys in the numeric keypad area and release [Alt], the ASCII character relating to that number will be generated. So to type ESC, key '027' on the numbers pad whilst holding down [Alt].

Some word processors, including *PC Write*, allow any ASCII character to be generated in this fashion. Others, unfortunately including RPED, EDLIN and *WordStar*, interpret many characters generated this way exactly as if you had pressed the equivalent key on the keyboard. MS-DOS does the same, so this does not work with files generated using the COPY CON technique.

2. It is possible to generate ASCII characters with a value less than 32 from within *WordStar* by holding down [Ctrl], tapping [P] (which brings up the 'Print Control Menu'), and then tapping the alphabetic equivalent of the control code: [A] for one, [B] for two, [Z] for 26 (the end-of-file marker) and [] (open square bracket) for 27 or the ESC character. Go into Non-Document mode and press [Ctrl] [P] (which gets you the print menu) and then [Esc]. You should see ^[on screen. This certainly works with *WordStar 4*, but earlier versions only allow the generation of a limited number of control codes – you will have to experiment!

ANSI codes have a [in front of them so to clear the screen, for example, use the method above to generate the ESCAPE code, followed by a [and 2J. You now have ^[[2J on the screen. Save this, TYPE it from the A-prompt and the screen will clear.

Screen modes can be changed in the same way with ^[[=11 and ^[[=31 to toggle between 80 and 40 column text. Escape codes can even be used to redefine the function keys (see the 'Batch File' section for details).

Typing in long lines of such characters quickly becomes tedious. Instead you can use *SuperCalc* in replicate or cell repeat mode. When you are happy with your screen, turn off the borders and print it to disk. This gives a PRN file which is 100 percent ASCII and can be loaded into an ASCII text editor or Non-document mode *WordStar*. *SuperCalc 4* also has a help screen which

shows the graphics characters useful for making menus. Reach this screen by typing any letter and then pressing [F1]. You need to page through the first few screens to see the graphics symbols.

3. In a similar fashion it is possible to generate many of the control codes within the MS-DOS line editor EDLIN by holding down [Ctrl], pressing [V] and then the equivalent alphabet key.

4. A rather neat way of getting the ESC character into a text file is to redirect the output of the MS-DOS command CLS into a text file by entering the following at the MS-DOS prompt:

```
cls > cls.txt
```

This creates a four-byte file called CLS.TXT which, if you examine it with EDLIN, reads ^[[2J. The first ^[is in fact the ESC character, so you can simply use your word processor or text editor to edit the file CLS.TXT as you wish, using copy block commands to duplicate the ESC character whenever you need it.

5. One way of putting the ESC character into a file is to create the file using a straightforward text editor, marking the places where you want ESC with some other little-used character (say ~). Then use DEBUG to replace these with ASCII code 27.

Using DEBUG, first enter **DEBUG filename** at the MS-DOS prompt. This loads the file into memory ready for you to get to work. If you now enter **d100** at the DEBUG prompt it dumps the first 128 bytes of your file to screen. Use the instruction **s,100,ffff,7E** To find the address of any occurrences of the ~ character – this instruction means 'search from address 100 to address ffff (well beyond the end of the file) for any occurrences of hex 7E'. DEBUG will list the address of each occurrence. To change these to, say, the ESC character, you need to enter **E address**. This displays the contents of that address – 7E – followed by a stop. Enter **1B** (Hex for ESC) and the change is made. When you have finished enter **w** to write the changes to disk, and **q** to quit DEBUG.

56. Defining Function Keys

PROMPT is a command intended to let you redefine the system prompt (A>, C> or whatever) so that it's friendlier or more informative. It has other uses too: in particular, you can use it in conjunction with ANSI.SYS to redefine your PC's function keys.

Before you start, however, you must have ANSI.SYS installed. This means adding the line **device=ansi.sys** to the CONFIG.SYS file on your boot disk, copying the file ANSI.SYS onto the disk if it's not already there, and then rebooting the machine.

Now to the actual key redefinition. You use a

PROMPT command of the form

PROMPT \$e[0;n;"string"p

where **n** is a number identifying the function key you want to redefine and **string** is the series of characters you want the function key to produce from now on. You can make the function keys produce different strings when [shift], [alt] or [ctrl] are held down, simply by using different values of **n** as follows:

Key	Normal	Shift	Ctrl	Alt
F1	59	84	94	104
F2	60	85	95	105
F3	61	86	96	106
F4	62	87	97	107
F5	63	88	98	108
F6	64	89	99	109
F7	65	90	100	110
F8	66	91	101	111
F9	67	92	102	112
F10	68	93	103	113

This means you can redefine [F1] to produce "Hello World" by typing in

PROMPT \$e[0;59;"Hello World"p

As a more useful example you could redefine [Alt] [F10] to produce "DIR A:*.DOC" using

PROMPT \$e[0;113;"dir a:*.doc"p

You can restore the original key definitions with a command in the form

PROMPT \$e[0;n;0;np

In other words, you could undo the two example definitions with

PROMPT \$e[0;59;0;59p

PROMPT \$e[0;113;0;113p

By now you will have noticed that all these fancy PROMPT commands have the unfortunate side effect of making the system prompt disappear. You can get the prompt back with a normal PROMPT command such as

PROMPT \$p\$g

If using the PROMPT command in a batch file, it is important to remember that the PROMPT method works by sending control characters to the screen: an earlier ECHO OFF command will block these and prevent the command from working properly.

An alternative way of sending the appropriate sequence to screen is to use the batch file ECHO command. The problem here is entering the ESC character into the batch file, but this can be done by using some other character, such as a ? query character, and then using DEBUG to alter all occurrences of the query character (hex 3F) to the ESC character hex 1B (see 'Generating Control Codes' elsewhere in this section).

For example, the following batch file redefines [Alt] plus the function keys to perform some routine tasks:

ECHO OFF

ECHO ?[0;104;"chkdsk";13p

ECHO ?[0;105;"format a: ";13p

ECHO ?[0;106;"backup c:*. * a: /s";13p

ECHO ?[0;107;"restore a: c: /s";13p

ECHO ?[0;108;"tree /f";13p

The 13 at the end of each line is the ASCII code for a

Here is a batch file that does the whole thing for you. First of all, enter the following as a batch file called FKEY.BAT:

```
ECHO OFF
set oldprmt=%prompt%
prompt
set key=%1
set code=32
if not %2==/e goto skip
set code=13
shift
:skip
set string=%2 %3 %4 %5 %6 %7 %8 %9
fkey1 1 59 2 60 3 61 4 62 5 63 6 64 7 65 8 66 9 67 10 68 11
```

Then enter this file as FKEY1.BAT:

```
ECHO OFF
shift
:loop
if %key%==%0 goto found
shift
shift
if not %0==11 goto loop
echo Invalid: key number must be in range 1-10
goto done
:found
rem code if in %1
set key=
prompt $e[0;%1;"%string%";%code%p
ECHO ON
```

```
ECHO OFF
:done
rem remove environment strings set by file
set code=
set key=
set string=
prompt %oldprmt%
set oldprmt=
```

To use, type **fkey n /e <string>** where **n** is the number of the function key to redefine, **/e** is optional and if included will cause the new command to be executed automatically, and **<string>** is the new definition.

For example, to set F1 to display the directory of drive A just type:

fkey 1 /e dir a:

This batch file requires CONFIG.SYS to contain the line **DEVICE=ANSI.SYS** if it is to work properly.



carriage return, so that the command generated by the function key is generated immediately. The only problem with this approach is that you can't TYPE the batch file properly once you've converted the queries – the escape codes just feed through the ANSI driver and are interpreted accordingly. A cleaner way is to use this TYPE feature to your advantage, by putting all the escape sequences into a small text file on their own. That way you only need `type redefine.txt` (or whatever) in your AUTOEXEC.BAT to redefine the keys.

57. Customising Pause

The PAUSE command always generates the line 'Strike a key when ready...' and waits for you to press a key. However, you might prefer to display a different message. This can be done by re-directing PAUSE to NUL, losing the standard display for a new message displayed by a previous ECHO command:

```
ECHO Insert a new disk and press a key
ECHO to continue, or [Ctrl-C] to quit:
PAUSE >NUL
```

58. Graphics from the Keyboard

The ANSI screen driver can be used to redefine [Alt] plus a given key to produce a graphics character on screen. To do this the screen driver has to be loaded, by including the line `device=ansi.sys` in your CONFIG.SYS file, and copying ANSI.SYS into the root directory of your boot disk. Once ANSI is installed, you have to send an escape sequence to the screen for each key redefinition you want to do.

Each sequence consists of ESC (ASCII code 27 in decimal or 1B in hex), a left square bracket '[', two numbers identifying a key, the character code for the graphics character, and a lower case letter 'p' to finish off with. The key numbers for the main letter keys are as follows:

[Alt] plus:	Key numbers:
Q	0;16

59. Useful Books

If you're really into batch files then we thoroughly recommend *Supercharging MS-DOS* by Van Wolverton (£17.95 from Microsoft Press). Van's just wild about menus! This book contains a full description of a menu system created under MS-DOS with batch files and ANSI.SYS. Also from Microsoft Press, but only costing £5.95, is *MS-DOS Batch Files: Quick Reference Guide* by Kris Jamsa.

W	0;17
E	0;18
R	0;19
T	0;20
Y	0;21
U	0;22
I	0;23
O	0;24
P	0;25

A	0;30
S	0;31
D	0;32
F	0;33
G	0;34
H	0;35
J	0;36
K	0;37
L	0;38

Z	0;44
X	0;45
C	0;46
V	0;47
B	0;48
N	0;49
M	0;50

In case you hadn't figured it out already, the strange order here is because the key codes are based on keyboard layout rather than position in the alphabet. This also explains the numbering gaps after P and L.

To specify the graphic you want your chosen key combination to produce, simply use its character code: so 176 (hex B0) would give you a shaded block, while 201 (hex C9) would give you the top left corner of a double-lined box.

To string all this together, if you want [Alt] [Q] to produce a shaded block, you need to send the ESC character followed by:

```
[0;16;176p
```

See 'Generating Control Codes' elsewhere in this section for how to get an ESC character into a batch file, or alternatively you could simply enter:

```
prompt $e[0;16;176p
```

At the MS-DOS prompt.

60. Environment Space

The PROMPT, PATH and SET commands all take up space in the area MS-DOS sets aside for 'environment variables'. If you make prodigious use of these you may well encounter a 'Out of environment space' error message. For MS-DOS version 3.1 or later the space can be increased by using the line:

```
shell=command.com /p /e:n
```

in your CONFIG.SYS file, where *n* is a number indicating how much memory space you want to allocate.

Under version 3.2, *n* is simply the number of bytes you want allocated, and can be anything up to 32768 (32K in other words). The default is 127 bytes, so

`shell=command.com /p /e:256` might be a reasonable setting if you have these problems.

Under version 3.1, *n* is the environment size in 16 byte units, and you can allocate anything from 160 bytes (with *n* set to 10) to 992 bytes (*n* set to 62). ●

The 8-bit ASCII Code

Dec	Hex	Ch	Ctrl	Mem	Meaning	Dec	Hex	Ch	Ctrl	Mem	Meaning	Dec	Hex	Ch	Ctrl	Mem	Meaning
0	00h		^@	NUL	Null	11	0Bh	♂	^K	VT	Vertical tab	22	16h	□	^V	SYN	Synchronous mark
1	01h	␣	^A	SOH	Start of heading	12	0Ch	♀	^L	FF	Form feed	23	17h	↑	^W	ETB	End transmission of block
2	02h	␣	^B	STX	Start of text	13	0Dh	♪	^M	CR	Carriage return	24	18h	↑	^X	CAN	Cancel
3	03h	▼	^C	ETX	End of text	14	0Eh	♪	^N	SO	Shift out	25	19h	↓	^Y	EM	End of medium
4	04h	♦	^D	EOT	End of transmission	15	0Fh	✱	^O	SI	Shift in	26	1Ah	→	^Z	SS	End of file (EOF)
5	05h	▲	^E	ENQ	Enquire	16	10h	▶	^P	DLE	Data link escape	27	1Bh	←	^[	ESC	Escape
6	06h	▲	^F	ACK	Acknowledge	17	11h	◀	^Q	DC1	Device control 1, or XON	28	1Ch	⌞	^_	FS	File separator
7	07h	•	^G	BEL	Bell	18	12h	↑	^R	DC2	Device control 2	29	1Dh	↔	^]	GS	Group separator
8	08h	■	^H	BS	Backspace	19	13h		^S	DC3	Device control 3, or XOFF	30	1Eh	▲	^^	RS	Record separator
9	09h	○	^I	HT	Horizontal tab	20	14h	¶	^T	DC4	Device control 4	31	1Fh	▼	^_	US	Unit separator
10	0Ah	␣	^J	LF	Line feed	21	15h	§	^U	NAK	Negative acknowledge						

Dec	Hex	Chr	Dec	Hex	Chr	Dec	Hex	Chr	Dec	Hex	Chr	Dec	Hex	Chr	Dec	Hex	Chr
32	20h		64	40h	@	96	60h	'	128	80h	Ç	160	A0h	á	192	C0h	Ⓐ
33	21h	!	65	41h	A	97	61h	a	129	81h	ü	161	A1h	í	193	C1h	Ⓑ
34	22h	"	66	42h	B	98	62h	b	130	82h	é	162	A2h	ó	194	C2h	Ⓒ
35	23h	#	67	43h	C	99	63h	c	131	83h	â	163	A3h	ú	195	C3h	Ⓓ
36	24h	\$	68	44h	D	100	64h	d	132	84h	ä	164	A4h	ñ	196	C4h	Ⓔ
37	25h	%	69	45h	E	101	65h	e	133	85h	å	165	A5h	Ñ	197	C5h	Ⓕ
38	26h	&	70	46h	F	102	66h	f	134	86h	ä	166	A6h	ª	198	C6h	Ⓖ
39	27h	'	71	47h	G	103	67h	g	135	87h	ç	167	A7h	º	199	C7h	Ⓗ
40	28h	(	72	48h	H	104	68h	h	136	88h	ê	168	A8h	¿	200	C8h	Ⓖ
41	29h	)	73	49h	I	105	69h	i	137	89h	ë	169	A9h	⌈	201	C9h	Ⓖ
42	2Ah	*	74	4Ah	J	106	6Ah	j	138	8Ah	è	170	AAh	⌋	202	CAh	Ⓖ
43	2Bh	+	75	4Bh	K	107	6Bh	k	139	8Bh	ï	171	ABh	½	203	CBh	Ⓖ
44	2Ch	,	76	4Ch	L	108	6Ch	l	140	8Ch	î	172	ACH	¼	204	CCh	Ⓖ
45	2Dh	-	77	4Dh	M	109	6Dh	m	141	8Dh	ì	173	ADh	ı	205	CDh	=
46	2Eh	.	78	4Eh	N	110	6Eh	n	142	8Eh	Ë	174	A Eh	“	206	C Eh	≠
47	2Fh	/	79	4Fh	O	111	6Fh	o	143	8Fh	À	175	AFh	”	207	CFh	≠

Dec	Hex	Chr	Dec	Hex	Chr	Dec	Hex	Chr	Dec	Hex	Chr	Dec	Hex	Chr	Dec	Hex	Chr
48	30h	0	80	50h	P	112	70h	p	144	90h	É	176	B0h	⌘	208	D0h	Ⓐ
49	31h	1	81	51h	Q	113	71h	q	145	91h	æ	177	B1h	⌘	209	D1h	Ⓐ
50	32h	2	82	52h	R	114	72h	r	146	92h	Æ	178	B2h	⌘	210	D2h	Ⓐ
51	33h	3	83	53h	S	115	73h	s	147	93h	ø	179	B3h	⌘	211	D3h	Ⓐ
52	34h	4	84	54h	T	116	74h	t	148	94h	ð	180	B4h	⌘	212	D4h	Ⓐ
53	35h	5	85	55h	U	117	75h	u	149	95h	ò	181	B5h	⌘	213	D5h	Ⓐ
54	36h	6	86	56h	V	118	76h	v	150	96h	û	182	B6h	⌘	214	D6h	Ⓐ
55	37h	7	87	57h	W	119	77h	w	151	97h	ù	183	B7h	⌘	215	D7h	Ⓐ
56	38h	8	88	58h	X	120	78h	x	152	98h	ÿ	184	B8h	⌘	216	D8h	Ⓐ
57	39h	9	89	59h	Y	121	79h	y	153	99h	Ö	185	B9h	⌘	217	D9h	Ⓐ
58	3Ah	:	90	5Ah	Z	122	7Ah	z	154	9Ah	Ü	186	BAh	⌘	218	DAh	Ⓐ
59	3Bh	;	91	5Bh	[	123	7Bh	{	155	9Bh	ç	187	BBh	⌘	219	DBh	Ⓐ
60	3Ch	<	92	5Ch	\	124	7Ch		156	9Ch	£	188	BCh	⌘	220	DCh	Ⓐ
61	3Dh	=	93	5Dh	]	125	7Dh	}	157	9Dh	¥	189	BDh	⌘	221	DDh	Ⓐ
62	3Eh	>	94	5Eh	^	126	7Eh	~	158	9Eh	₹	190	BEh	⌘	222	DEh	Ⓐ
63	3Fh	?	95	5Fh	_	127	7Fh	Δ	159	9Fh	ƒ	191	BFh	⌘	223	DFh	Ⓐ

T H E
**HELP
SCREEN**
C O L L E C T I O N

43.



P R I N T E R S

62. Pound Sign Problems

It is very rare to see the £-sign on a computer screen, though few have trouble with the \$ symbol – another sign of America's efforts to dominate the world? But enough of politics. The usual solution is to use the 'hash' character # instead.

Fortunately it is fairly easy to get your printer to print out the £-sign, simply by telling it to print out a £ whenever it sees a #. Unfortunately this does mean problems if you want to print a # as well, but that's computers for you. With accounting packages the important thing is getting the £ down on paper – what you see on the screen is not so important.

The details vary from printer to printer, but basically to get a £-sign you set the DIP switches to select the 'UK' character set. If you have got the Amstrad DMP3000 printer you will find the details on page 30 of its manual: you switch the first three switches of DIP switch bank one to OFF, OFF and ON respectively.

Some software, such as GEM actually displays and prints a £-sign without problem.

63. Canon PW-1080

Epson is the IBM of the printer world: its printers have set an industry standard. This means that if your printer is not mentioned on the installation list of your software, always give it a go pretending it's an Epson as it might just work. The Canon PW-1080 is Epson-compatible, so if it is not available in its own right

probably the best driver to install is the Epson FX80. You could also experiment with the LX80 or MX80 to see if you get better results.

65. Surplus Line Feeds

When a printer gets to the end of a line it has to perform two separate functions, just like a typewriter, if it is to leave the print head in the right position to print the next line. It has to go back to the beginning of the line ('carriage return' or CR), and it has to move down to the next line ('line feed' or LF).

A printer performs these actions in response to specific codes sent to the computer: ASCII code 13 for CR and ASCII code 10 for line feed. Most software arranges for both codes to be sent together; however some software only sends the CR, expecting the printer to add the LF off its own bat. Most printer manufacturers allow for both eventualities with a small 'dip' switch, usually round the back or under a flap somewhere (see the manual for details), that determines whether a line feed is added whenever a carriage return is received.

If this switch is set wrong the result is either no line feeds at all, with your whole document ending up crammed into one messy line at the top of the page, or a surplus blank line between each line of text. If you are having problems experiment with changing the position of this switch. If this doesn't work then you may be able to change the number of line feeds from the software. *NewWord* and *WordStar*, for example,

64. Changing Paper Length

Some applications feed through a new sheet of paper by counting lines, and can be set for a change in paper size simply by specifying a new line count. Others feed paper by sending an ASCII code 12, the Form Feed character, which means that changes in paper length have to be specified at the printer end of things.

Most printers have switches for setting paper length as either 11 or 12 inches but can only be switched to, say, 70 lines per page from software. If the printer is Epson compatible the required code is 1B 43 46. To send these codes to your printer, proceed as follows:

1. Create a short text file called 70LINES.PRN using the COPY command:

```
copy con 70lines.prn
*****
^Z
```

The actual character used doesn't matter, so long as it creates a file greater than four characters long.

2. Use DEBUG to alter the file as follows:

```
debug 70lines.prn
e100          - Starts data entry.
1B 43 46 1A   - Note the 1A - this is the Ctrl-Z
               end-of-file mark.
w             - Write file to disk.
q             - Exit from DEBUG.
```

3. You now have a file called 70LINES.PRN containing the required control codes. This can be sent to the printer with something like:

```
copy 70lines.prn prn
```

The only side effect of this technique is that a form feed will be printed after configuration. This costs you a sheet of paper each time you configure the printer. If this bothers you, find a control code sequence that will 'absorb' the form feed without doing anything harmful, such as the 'Set Horizontal Tab' sequence Hex 1B 65 00 nn. If you insert the first three of these codes before the EOF marker the form feed will be captured as the nn code.

includes special printer drivers that do not send line-feed with carriage returns.

A rather more drastic solution, if you are suffering from too many line feeds, is to disable pin 14 of the Centronics cable. Earthing pin 14 tells the printer to line-feed after each carriage return, so disabling it can solve the problem. This can be done either by disconnecting the wire from the pin inside the D-connector, or taping over the connectors at the printer end of the lead.



66. Sending Form Feeds

Most printers interpret ASCII code 12 as a 'form feed' command, automatically moving on to the top of a new sheet of paper when it is received. Fortunately this control code can actually be entered from the MS-DOS prompt directly, simply by holding down [Ctrl] and tapping [L]. The following batch file, called FF.BAT, can be useful:

```
ECHO OFF
ECHO ^L >PRN
```

Displaying this batch file using the MS-DOS TYPE command reveals the ^L to have been replaced by the biological symbol representing ASCII code 12.

67. Customising Print Drivers

A very common problem area is that of matching non-Epson printers to the print drivers supplied with an application. A case in point is what *PC PLUS* call the 'stripy tiger' problem, which gets its name from the effect generated when a nearly Epson compatible printer is used to print out the drawing of a tiger supplied as a sample with *GEM Paint*.

What follows is a technique for using DEBUG to customise the Epson print driver supplied with GEM so that it better suits the problem printer. The principles behind the technique can be used with a wide range of printers and applications, but first a bit of background – and dig out the Control Codes section of your printer manual, as you will be needing it.

GEM programs, and many other graphics programs, use purpose-built printer drivers to ensure that the correct control codes are sent to your printer. Most programs include an Epson driver because the Epson range has become a standard, but unfortunately when it comes to printing graphics you soon discover that the only true Epson compatible is made by Epson – the compatibility starts breaking down.

Printer control codes are known as 'Escape codes' because they usually start with ASCII character number 27; the ESC character (1B in hexadecimal).

The Stripy Tiger Effect



■ What happens when the GEM print driver is used with the wrong printer – the line spacing is set wrong

The command that is at the root of the 'stripy tiger' syndrome is `ESC '3' n`, which for an Epson printer sets the line spacing to $n/216$ th of an inch for future line feeds; but for many other makes sets the line spacing to a rather wider $n/180$ th of an inch. It is this discrepancy that causes the stripes.

Attempts to send the correct code prior to loading GEM are fruitless, because the GEM printer driver simply comes along and re-sets the printer to the wrong code. The only answer is to get yourself another printer driver from Digital Research – or to get into the Epson driver supplied and change it to suit your printer.

In hexadecimal notation the command `ESC '3' n` translates into `1B 33 nn`. Most Epson drivers have `nn` set to about 15 or 17, which corresponds to around 12 for a $1/180$ th inch setting. All we have to do is look for the hex string `1B 33 17` (or thereabouts) in the program code, change the 17 to 12 and the job is done. 12 might not be the right value for your printer, so some experimentation might be necessary here.

But how to find `1B 33 17` in a program which may be several kilobytes long, and which of the many GEM files should we look in? The answer is to use DEBUG, on the MS-DOS boot disk, to find the necessary code; while the printer driver is the file `EPSMONH6.SYS` in the `GEMSYS` folder. The following procedure is for a twin drive machine, so if you have a single drive or hard disk fitted to your system you will have to change it accordingly:

1. Playing around with DEBUG can be dangerous, so always use a copy of whatever disk you are going to use it with. DEBUG can quite literally screw up your hard disk without you knowing, so BACK IT UP FIRST.

2. Put the MS-DOS disk containing the DEBUG utility in Drive A, and the disk containing EPSMONH6.SYS in Drive B. Enter:

DEBUG B:EPSMONH6.SYS

The drives will run for a few seconds and the A-prompt will change to the hyphen which is the DEBUG prompt.

3. Enter **D 100** and the screen will show byte-by-byte the contents of the file in question. Down the left are shown the addresses of the left-most byte; the addresses at which DEBUG has placed the file in memory, starting with 100. The central area shows the actual hex contents of the first 128 bytes of the file. The right hand area is taken up with an ASCII representation of the contents.

4. You can also see that it is not going to be easy to find **1B 33 17** just by looking, so enter:

S,100,FFFF,1B 33

This tells DEBUG to search the file for the code **1B 33**, starting at address 100 and ending at FFFF – well past the end of the program file.

5. Almost immediately two addresses appear; something like 9031 and 9037. Enter **D 9030** and there you are: on the top line starting at address 9031 is **1B 33 17**; and at 9037 is **1B 33 01**. Do not change this second entry because GEM makes two passes when printing each line; the second pass advancing the paper by the smallest fraction, so if you increase the 01 the problem will only get worse.

6. Now that you have found the offending value you can change it. The 17 is at location 9033, so enter **E 9033** and you will see the address followed by the contents – 17 – followed by a stop. Enter 12 (which will appear after the stop), and the 17 will change to a 12. Prove it by entering **D 9030** again. All that is required now is to enter **W** (for Write), and the file will be written back to disk with the new value in place.

7. Enter the command **Q** to Quit DEBUG, boot up GEM, try printing the Tiger and you will soon find out if 12 is the right value for your printer. If not, repeat with another value.

Sometimes the Tiger may not be appear as chubby as he is on screen; he should be roughly circular. The solution to this problem lies in the printer options set for GEM Output. Select 'Printer' from the Options Menu and you are given the choice of 'Full Scale' or 'Best Fit'. It will probably default to 'Full Scale', so change this to 'Best Fit'; and while you're at it choose 'No' for the Final Form Feed as this saves paper while you experiment.

Save the new option, try printing the Tiger and hopefully he will appear with a chubby face and all his stripes where they should be. Make the new option the Default option, and when you are satisfied with the changes put the new EPSMONH6.SYS file on your working disk.

In the search above we looked for the two codes **1B 33** rather than the full **1B 33 17**. This is because the program might use some other number to define the line spacing. Another useful code to search for when trying to find printer control instructions is **ESC 'L'** (**1B 4C**) which prepares the printer for 1280 dots per line dump, or **ESC 'K'** (**1B 4B**) which sets 640 dots per line. If you find one of these then you have almost certainly found the area to explore for other codes.

46.

```
-s,100,ffff,1b 33

1460:9031
1460:9037
-d 9030

1460:9030 0D 1B 33 17 0A 05 0D 1B-33 01 0A 04 1B 4C 00 00 ..3....3...L..
1460:9040 02 1B 32 00 00 00 00 01-00 07 FE FE FD FD FB FB ..2.....
1460:9050 01 00 02 00 04 00 35 0A-32 0A 3B 0A 38 0A 42 0A .....5.2.;8.B.
1460:9060 3E 0A 02 1B 47 02 1B 48-02 1B 34 02 1B 35 03 1B >...G..H..4..5..
1460:9070 2D 01 03 1B 2D 00 03 1B-4A 3F FF FF F0 FF C0 C0 -...-...J?.....
1460:9080 18 FF 00 FF 91 F1 00 00-0F 00 00 00 00 00 FC 3F .....?
1460:9090 04 20 04 20 74 2E 54 2E-54 2E 54 2E 54 2E 74 2E . . t.T.T.T.T.t.
1460:90A0 04 20 04 20 FC 3F 00 00-00 00 13 00 FF FF 00 80 . . .?.....
-e 9033

1460:9033 17.12

-d 9030

1460:9030 0D 1B 33 12 0A 05 0D 1B-33 01 0A 04 1B 4C 00 00 ..3....3...L..
1460:9040 02 1B 32 00 00 00 00 01-00 07 FE FE FD FD FB FB ..2.....
1460:9050 01 00 02 00 04 00 35 0A-32 0A 3B 0A 38 0A 42 0A .....5.2.;8.B.
1460:9060 3E 0A 02 1B 47 02 1B 48-02 1B 34 02 1B 35 03 1B >...G..H..4..5..
1460:9070 2D 01 03 1B 2D 00 03 1B-4A 3F FF FF F0 FF C0 C0 -...-...J?.....
1460:9080 18 FF 00 FF 91 F1 00 00-0F 00 00 00 00 00 FC 3F .....?
1460:9090 04 20 04 20 74 2E 54 2E-54 2E 54 2E 54 2E 74 2E . . t.T.T.T.T.t.
1460:90A0 04 20 04 20 FC 3F 00 00-00 00 13 00 FF FF 00 80 . . .?.....
-w
-q
```

■ Customising GEM's EPSMONH6.SYS print driver using DEBUG – shown above is what you should enter and what you should see on screen

Finding the printer configuration areas in other application programs calls for a little detective work, but can be rewarding. Probably the best way to proceed is to find a printer driver that seems to be fairly compatible, and proceed from there. It Remember that it is best to make just one change to the code at a time, save it to disk and try it out: resist the temptation to make major changes all at once, as you will never be able to tell what effected what. It goes without saying that you should always work with a copy – never with your master disks.

68. More Stripes

Stripes in pictures printed out from your printer could have nothing to do with the printer driver software at all. Examine the picture closely and you may find there is in fact something in those stripes, but very faint. This usually means that one of the nine pins of the dot-matrix print head is a little weak, which (thanks to the microscopic size of these critters) probably means you need a new print head.

69. Loose Printer Chips

Does your printer occasionally emit strange bleeping noises and refuses to print, getting worse and worse until it won't work at all? Inside the Epson FX80 (and other Epson models of this vintage) is a large, custom-built IC chip – much bigger than the 8086 inside a PC. With all the usual banging about of print heads and platens, and especially if the printer has been moved around, this chip can work a bit loose in its socket. Result: funny bleeps and stoppages, getting worse as the the chip works looser.

The answer is to disconnect the mains, remove the top cover (four screws and the platen-wind knob), then gently press the chip back into its socket. You can't miss it – it's surrounded by a black plastic collar – but if in doubt just press all the chips gently home.

It's up to you whether you want to try this, but it certainly worked for us. If your printer's still under guarantee it would be best to take it to your dealers and let them do the pressing, though.

70. Bold Type in WordStar 1512

The printer drivers supplied with *WordStar 1512* are unable to produce bold type in NLQ mode on Epson printers, as all use coding for double-strike rather than emphasis. This is all very well for earlier Epson models but in NLQ mode the bold appearance of double strike is lost. This problem can be is solved with the aid of a disk editor program.

Once you have installed a working version of *Wordstar 1512* from the numerous disks supplied, you

should have a directory called \1512\PROGRAM on your hard disk, or a floppy called '1512 WP'. This contains three files which represent the printer definitions, called PRINTER1.PDF, PRINTER2.PDF and PRINTER3.PDF.

To change from double-strike to emphasis you must alter the escape codes stored in these files. Double strike is toggled on with ESC G and off with ESC H, while emphasis is represented by ESC E and ESC F. The task is to locate the ASCII characters 'G' and 'H' and change them to 'E' and 'F'.

Examine each file and in sector 7 of the code (the last sector of each file) and you will find:

ASCII representations of part of sectors 6 and 7 of PRINTERx.PDF

```
....^J.^[A&c,^[S
^A.^[T.^[S^@.^[T
.^H.^[5.^[4.ITAL
IC OFF.ITALIC ON
```

Change the

```
.^[G.^[H.^[-1.^[
-0.^[E.^[F.^[4.^
[5.^L.^M.^M^J.^M
```

→

G and H

```
.^[E.^[F.^[-1.^[
-0.^[E.^[F.^[4.^
[5.^L.^M.^M^J.^M
```

71. Cut Sheet Feeders

Most software packages assume A4 paper to have 66 lines, but unfortunately the mechanics of an Epson printer fitted with a sheet feeder require that there is a margin at the top of each page, and the paper out switch forces another at the bottom. The result is that only 58 lines are actually available to print on, which can cause endless problems.

The cure is simple. Just tell your software that the page length is 58 and all should be hunky-dory.



72. The Problem with Pound Signs

The 'Case of the missing Pound Sign' illustrates perfectly that computer technology does not proceed in a logical fashion, but rather as a series of historical developments that 'seemed like a good idea at the time'.

It all started with the American Standard Code for Information Interchange (ASCII for short), introduced in 1963. This was a standard laid down for the mapping of seven-bit binary numbers onto the characters required to represent text – letters, numbers, punctuation and a number of control codes. It is a standard that has become accepted throughout the world, despite the fact that, from our point of view at least, it has one profound disadvantage: the symbol used to express currency is of course the good old dollar (\$) – the sterling pound sign (£) is nowhere to be seen.

ASCII is a seven-bit code because many early computers used the eighth bit as a parity check. Micro computers these days use the full eight bits for data storage, and when IBM launched its PC in 1982 it decided to make use of the remaining places (decimal 128 through to 255, or **80h** to **FFh**) for what it called 'foreign' symbols. Our £-sign was accordingly allocated position 156, or **9Ch**.

IBM was of course interested in the international market for its new baby, and so had to adapt its design for the specific needs of each country, while maintaining compatibility. The American keyboard does not have a £-sign on it at all – above the '3' is a # (used by the Americans to symbolise 'number'), while our # key is replaced with a '. In this country most compatibles have a £-key right next to the \$-key, while PC-DOS version 3.0 onwards (MS-DOS version 3.2 onwards) provides a program which, when run, ensures that the £-key sends the code **9Ch** to represent the £ sign, rather than **23h** which is the ASCII code for #. Other countries are supported either, in the case of MS-DOS version 3.2, with their own program (KEYBUK for the United Kingdom, KEYBGR for Germany, KEYBFR for France, and so on) or, for version 3.3 onwards, by supplying the KEYB program with appropriate parameters.

This program should be run automatically by using a suitable text editor to add the line **keybuk (keyb uk** for MS-DOS version 3.3 and on) in to your AUTOEXEC.BAT file. You can check if it has been executed by typing [Shift] [3] (the £ key) at the MS-DOS prompt. If you get a £-sign all is well, if you get a # then it hasn't been run.

Also provided for domesticating your PC is the configuration command COUNTRY. Available in MS-DOS version 3.0 on, the line **country=044** should be included in your CONFIG.SYS file to again set various national parameters. The code is in fact the international dialling code, and a wide range of countries are catered for.

Meanwhile, on the printer front, the Japanese company Epson was rapidly becoming the standard-bearer. Unfortunately it decided to go a different route to IBM and use the codes 161 to 254 (**A1h** to **FEh**) for italic characters. To cater for the international market it used some of the printer's DIP switches (tiny banks of switches hidden round the back and used to configure the printer to particular needs) to select foreign characters. When these are set for the UK the effect is to print a £-sign whenever ASCII code **23h** (a hash) is received.

This is rather unfortunate for your PC: having dutifully run KEYBUK and set the DIP switches of your printer for the UK, pressing the £-sign key sends **9Ch**, which is ignored completely by the printer. To get a £-sign on your printer, you have to press the #-sign on

the keyboard – hardly satisfactory.

IBM were aware of this problem when they launched the PC, and so also launched their own version of one of the Epson printer range as the 'IBM Graphics Printer'. The main difference between the two models is that the IBM model interprets character codes above 128 in the same way as the IBM PC, so sending the printer **9Ch** would actually result in a £-sign on the paper.

Epson could not very well ignore this, so decided to offer customers the best of both worlds. Accordingly, other DIP switches were used to allow you to select between the Epson character set and the IBM. Just to complicate things, IBM had provided two separate character sets, and only the IBM Character Set 2 caters for £-signs. These DIP switches also override those that interpret the # code as a £ – with the IBM set selected, code **23h** is printed as a #.

Meanwhile, word processor developers were desperately trying to cater for the whims of both computer and printer manufacturers while still providing users with a friendly way of getting what they wanted on both screen and paper. The £-sign was dealt a heavy blow when MicroPro's *WordStar* package became a standard. Being an American package it totally ignored the requirements of the UK; furthermore, thanks to its seven-bit character code, it was unable to make use of IBM's extended ASCII table. Pressing the £-key under *NewWord 2* or *3*, for example, simply blinks the screen. To get a pound sign you have to put up with the # on screen, and set the printer to the Epson character set so that this gets printed out as a £.

Thankfully most word processors come with a range of 'printer drivers': programs that sit between the word processor and the printer, making sure that what is sent to the printer gets printed in the fashion required. Every word processor these days provides drivers for most Epson models and the IBM Graphics Printer at the very least, and many for a very wide range of other types. *WordStar 4*, for example, does allow the display of a £-sign on screen, although it is actually stored in memory as three separate characters: **1B 9C 1C** (hex). Selecting the appropriate print driver ensures that this sequence is interpreted as a £-sign by the printer.

So, to get a pound sign on screen and printer, proceed as follows:

1. Put the line **KEYBUK** in your AUTOEXEC.BAT file, and make sure the file KEYBUK.EXE is on your boot disk (**keyb uk** and KEYB.EXE for MS-DOS version 3.3 and above).
2. Put the line **country=044** in your CONFIG.SYS file.

3. Boot up your word processor, or the program provided with it for configuration.

4. Select an appropriate printer driver from the menu. If you have an Epson printer, you will have to choose between the appropriate model or 'IBM Graphics Printer' by experiment. Otherwise, select the model nearest to yours (daisywheels present their own problems, as usual!)

5. Make sure the DIP switches of your printer are set to match – in particular, that they are set to 'IBM character set 2' if you are using a printer compatible with the IBM Graphics Printer.

6. Check what is happening by quitting the word processor altogether and seeing what you get when you send characters to the printer direct from MS-DOS. To do this, make sure the printer is connected and on-line and then press [Ctrl] [p]. This tells your PC to send whatever is displayed on screen to the printer as well. Now enter a # and see what you get on the printer. If the # is interpreted as a £, then you know the printer is printing the Epson character set. Next enter a £. If this is printed as a £ then the printer is set to IBM character set 2.

7. Another check is to directly send your printer ASCII codes 9C and 23 (hex), which are the IBM extended ASCII codes for a pound and hash sign respectively, by entering the following commands to create a batch file that sends them directly. It requires the MS-DOS utility DEBUG in the current directory or on the current path:

```
copy con ukpnd.bat
ECHO xx > PRN
[Ctrl Z]
```

```
debug ukpnd.bat
e105
9C 23
w
q
```

Note that it is very important that the two x's are the 6th and 7th character in the batch file, as shown above. When entering 9C 23 you are directly entering hex values over these x's. The present value will be shown (78 hex) and you type the new value. The space between the two values moves to the next memory location, i.e. the second 'x'. Run the batch file by entering ukpnd at the MS-DOS prompt and see what you get at the printer.

8. Be patient – you should certainly be able to get a £ on paper, although some word processors might not be

capable of showing one on screen.

9. If all else fails, try running the GRAFTABL utility (supplied on your MS-DOS disk) before running the word processor. It's possible that your word processor is running in the Graphics display mode, in which case the dot patterns for codes above 128 are not automatically held in memory (for some reason known only to IBM).

73. Colour Screen Dumps

The MS-DOS utility GRAPHICS has options for producing colour output on a variety of printers. The Epson EX800, for example, is a 9-pin printer using a four colour (red, green, blue, black) ribbon, and the nearest thing to this supported by GRAPHICS is the IBM Personal Computer Color Printer with RGB ribbon. To print in colour, try:

```
graphics color4
```

If you now load any graphics display software and press [Shift][PrtSc], then the screen display will, with luck, be printed in colour. Note though that this only applies to screens which use the CGA (colour graphics adaptor) graphics modes – not ordinary text displays, even if they include coloured text.

74. DIP Switches

The default settings of your printer, which take effect when you switch on, are usually set by DIP switches – tiny switches grouped together somewhere awkward. Unfortunately there are no standards, but certain functions are worth checking right away. Some can be overridden by control codes in the document. Consult the manual to find out which switch is which, and remember to turn the printer off before changing them:

International character set – usually three switches which together allow the selection of eight different nationalities. Set for the UK to have a £ printed in place of a # symbol. The Epson control code `ESC R <n>` overrides the switch setting.

IBM/Epson character set – determines whether the printer reacts to the characters 128 to 255 (decimal) by printing out the IBM graphics set or italics instead. Selecting for IBM graphics usually overrides the international setting, but allows [Shift] [PrtSc] printouts to include box graphics. As with the above, `ESC t <n>` overrides the switch.

Draft/NLQ typeface – sets print quality to draft or Near Letter Quality. `ESC x <n>` overrides.

Zero character – selects for either a slashed or unslashed zero.

Automatic linefeed – decides whether paper is automatically advanced one line when a CR (ASCII code 13) is received, or whether software has to send its own LF (ASCII code 10). The latter is the normal setting for a PC. If you are having problems with line spacing, check this switch.

Page length – selects between 11 or 12 inch paper, so determining the length of paper fed when a FF character is received (ASCII code 12) or the Form Feed button is pressed. **ESC C <n>** overrides.

Skip perforation – causes the printer to automatically advance four lines at the end of each page. Switch off if your software is already looking after page format.

Cut-sheet feeder – makes the necessary adjustments when a feeder is fitted.

75. Epson Control Codes

Most printer manufacturers gave standardised to a greater or lesser degree on a series of 'escape codes' defined by Epson. When these sequences are sent to the printer, as data along with your document, the printer doesn't print them but instead interprets them as laid out below. They are called escape codes because they are almost all preceded by the non-printing ASCII control code 27, or 'ESC'. Very few non-Epson printers respond properly to all the escape codes, but many respond to a useful set of them.

TYPE STYLES

code	hex	meaning
ESC SI	1B 0F	Condensed print On

DC2	12	Condensed print Off
ESC SO	1B 0E	Double width On
DC4	14	Double width Off
ESC 4	1B 34	Italic print On
ESC 5	1B 35	Italic print Off
ESC E	1B 45	Bold print On
ESC F	1B 46	Bold print Off
ESC G	1B 47	Double strike On
ESC H	1B 48	Double strike Off
ESC M	1B 4D	12-pitch (elite) On
ESC P	1B 50	10-pitch (pica) On
ESC S SOH	1B 53 1	Subscript On
ESC S NUL	1B 53 0	Superscript On
ESC T	1B 54	Sub/Superscript Off
ESC - SOH	1B 2D 1	Underline On
ESC - NUL	1B 2D 0	Underline Off
ESC W SOH	1B 57 1	Double width On
ESC W NUL	1B 57 0	Double width Off
ESC p SOH	1B 70 1	Proportional On
ESC p NUL	1B 70 0	Proportional Off
ESC x SOH	1B 78 1	NLQ On
ESC x NUL	1B 78 0	NLQ Off
ESC ! <n> allows selection of a whole range of styles in one go. See Epson printer manual for more details.		

LINE SPACING

Most 9-pin dot matrix printers have a resolution of 72 dots per inch, and print by default six lines per inch. This leaves a margin between each line which can be reduced to zero by requesting eight lines per inch.

Epson control codes allow control of line spacing in intervals either of 1/72 inch, or a third of this at 1/216 inch. Some printers interpret these codes as 1/60 inch and 1/180 inch respectively, which can cause confusion:

ESC 0	1B 30	1/8 inch spacing
ESC 1	1B 31	7/72 inch spacing
ESC 2	1B 32	1/6 inch spacing
ESC 3 <n>	1B 33 n	n/216 inch spacing
ESC A <n>	1B 41 n	n/72 inch spacing

CHARACTER DEFINITION

ESC t SOH	1B 74 1	IBM graphics set printed
ESC t NUL	1B 74 0	Epson italic set printed

OPERATING COMMANDS

BEL	07	Sound beep
BS	08	Back one space
HT	09	Move to next horizontal tab position
LF	0A	Line feed
VT	0B	Move to next vertical tab
FF	0C	Form feed (new page)
CR	0D	Carriage return
CAN	18	Clear print buffer
ESC <	1B 3C	Print one line left-right
ESC @	1B 40	Reset printer to defaults

76. Epson Codes for setting Page Format

ESC l <n>	1B 6C n	Left margin to <n>
ESC Q <n>	1B 51 n	Right margin to <n>
ESC C <n>	1B 43 n	Page length <n> lines
ESC C NUL <n>	1B 43 0 n	Page length <n> inches
ESC N <n>	1B 4E n	Bottom margin <n> lines
ESC O	1B 4F	Cancel bottom margin
ESC a <n>	1B 61 n	Set justification mode: n=0 left, 1 centred, 2 right, 3 full
ESC D <n...n> NUL	1B 44 n...n 0	Set up to 32 horizontal tab positions
ESC e NUL n	1B 65 0 n	Set horizontal tab spacing
ESC B <n...n> NUL	1B 42 n...n 0	Set up to 16 vertical tab positions
ESC b and ESC /		Allows setting up of eight vertical tab channels – see an Epson printer manual for details.

77. Revitalising Ribbons

You can re-ink a dead printer ribbon with the endorsing ink normally used for office rubber stamp pads. The procedure is simple if potentially messy. Squeeze a few drops of ink into a thin polythene bag (as used for sandwiches), drop in the old ribbon and carefully massage the ink down the whole length of the ribbon. Don't use too much ink! It may be impractical for long concertina ribbons in a cassette.

Alternatively, you can use a can of WD40 to redistribute the ink over the ribbon surface. Remove the cassette cover and spray the ribbon, then allow the WD 40 to dry leaving the ink redistributed over the ribbon surface. The ribbon is now ready to use again.

78. Laser versus 24-Pin Printers

When it comes to graphic resolution most 24-pin dot matrix printers can outperform a laser printer, frequently offering 360 by 360 dpi (dots per inch) as opposed to the 300 dpi available from the standard laser printer. This is not actually as useful as it sounds, though: laser output normally looks a good deal better, for two reasons that aren't immediately obvious.

For one thing, dot-matrix printers are mechanical things. The print head's tiny metal rods or 'pins' mark the paper by striking a ribbon against it, and this is inevitably a messy business. The individual dots end up as blobs rather than perfect squares or circles, and can easily be slightly out of position.

A laser printer, by contrast, works by a combination of optics and electrostatics, without the need for any impact on the paper. The mechanism is very similar to a photocopier. In a photocopier, light reflected off the original artwork strikes a rotating drum, altering the static charge on its surface. The drum then picks up an ink-like substance called 'toner' which is used to form the finished image. This is where the static electricity comes into play: the parts of the drum that were struck by the reflected light repulse the toner while the unlit parts attract it. Finally the drum rolls over the surface of a blank sheet of paper, transferring the toner onto it. The patterns of toner on the paper match those on the drum, which in turn match the patterns of light and dark on the original.

Laser printing is essentially the same process, except that the drum is charged not by light reflected off the original – there isn't one – but by the beam of a computer-controlled laser. Because this is optical rather than mechanical, it can be very accurate indeed.

There's also another problem if you're trying to get high-quality images out of a dot-matrix printer, as most applications software can't use the high resolution of a 24-pin printer effectively. It's easy enough to see why: at 360 dpi you can cram over 100,000 dots into a

square inch of paper, which is equivalent to around 14K of information. A sheet of A4 is around 100 square inches in area, so any application trying to specify how that sheet is to be printed, dot by dot, would need to use at least 1 Mbyte of memory.

But that's not all: just wait until you try sending 1 Mbyte of information down a cable to the printer. At normal data transmission speeds this would take a long, long time. You're up against this bottleneck whatever form your data takes, unless you're using the printer's built-in fonts.

Laser printers usually get round the problem by extending the idea of built-in fonts, so that the PC only has to send text information and short graphics commands. This does call for a fair bit of processing power actually inside the printer, and indeed some of the more expensive laser printers are actually very powerful computers in their own right. This accounts to some extent for the high cost of laser printers, and for the vast difference in price between simple and sophisticated ones.

79. Feeding Single Sheets

Many dot matrix printers can't print to the bottom of a single sheet of paper because of the relative positions of the print head and the friction feed mechanism. One solution is to use two sheets of those yellow self-adhesive notelets from 3M to stick two or more single sheets together, and then print right to the bottom of the first sheet. When the notelets are removed there is no trace of adhesive left on the sheets, and the notelets can be re-used a number of times.

80. Paper Length

11-inch paper gives 66 lines on a page at six lines per inch, 88 at eight lines per inch.

True A4 paper is 11.69 inches long, accommodating just over 70 lines at six lines per inch.

12-inch paper gives 72 line on a page at six lines per inch, 96 lines at eight lines per inch.

81. Mysterious Printing Problems

Do you suffer from mysterious problems when using a parallel printer, typical symptoms being perfect printing from some applications but error messages from others and when you try to print direct from MS-DOS? If so it is worth checking the printer cable. It would appear that sloppily made cables may connect the PC's printer port error line input (pin 15) to one of the printer's ground connections, rather than to the relevant error line output (pin 32). Since the error line is active low, this is equivalent to a permanent printer error.

82. Controlling Your Printer

Usually the handling of escape codes is handled automatically once you select the appropriate print driver for your application software. But occasionally, particularly if you are printing files direct from the MS-DOS prompt, it can be useful to be able send escape codes direct. The following batch file can be used to create any number of effects:

```
ECHO OFF
ECHO Switching to italic print
ECHO (escape codes) >PRN
```

Where *escape codes* are the codes necessary to switch the printer to, in this case, italic print. For most Epson-compatible models this would be **ESC 4**, or **1B 34** in hexadecimal numbers. Note that the figure translates to the ASCII code for the alphanumeric character '4', and is not treated as a numeric value. See the 'Using MS-DOS' chapter for how to get ESC into a text file.

83. Configuring Software

All printers use control codes to allow applications to tell the printer what they want it to do. A few of these codes, such as those for carriage return, line feed, formfeed, tab and backspace, date back to early mainframe computer terminals and follow the standard ASCII codes. However, different manufacturers cannot agree whether a carriage return code should be followed by a line feed code or not, so it has been left to printer manufacturers to implement particular features as they develop new models. Fortunately complete chaos has been avoided by the existence of three de-facto standards, one for each of the main types of printer: the daisywheel following codes originally set by Diablo and Qume; the dot-matrix following Epson; and the laser printer following those used by Hewlett Packard's LaserJet.

Configuration means telling your application package which codes it should use to get each feature and, of course, the more features that the program uses, the more complex the configuration requirements. Some packages allow you to specifically specify all the codes that your printer uses, which can mean a considerable amount of consulting obscure printer manuals and experimentation. However these days it is usual for them to allow you to choose your printer from a list. This is where problems usually start for no matter how long the list, your model never seems to be mentioned!

One reason for this is that some time has probably gone by since the software was written, and new models of printer are appearing all the time. Another

problem is that Japanese printer manufacturers have a terrible habit of giving different names to almost identical models in the US and in Europe. The best selling Star NL-10 printer, for example, was called the NX-10 in the USA.

Epson made its reputation with a model called the MX-80, which has had various revisions. The latest is called the Type III in Europe but 'MX-80 with Grafrax' in most US applications. This model led to three further models: the RX-80, using very similar control codes to its predecessor; the FX-80, with many additional features including an elite font (12cpi); and the IBM Graphics printer. This last, made for IBM by Epson, has a character set which matched that of the PC, including all the graphics characters. Unfortunately though it used identical control codes to the old MX-80 and so lacked many features introduced on the FX-80.

Modern printers, invariably described as 'FX-80 compatible', use codes based on the FX-80 but usually with the addition of an NLQ mode and an IBM compatible character set. If users want to use these features, selecting FX80 from the printer configuration list is not usually enough.

The FX80 itself was eventually superseded by models called – in the UK at least – the FX-85, FX-86, FX-800 and FX-850 (models starting with a 10 instead of 8 are wide carriage versions of the same printers, using the same control codes). Only the FX-86 onwards provide the IBM character set, but if we look at the printer list provided by *Wordstar 4* for example, we find nothing listed after the FX-85! This printer can't use the IBM graphics characters, while the IBM Graphics Printer cannot print most of the Epson's special styles. If you want both you need to find a printer listed which has IBM Graphics and the additional font styles, such as the IBM Proprinter.

If you have a 24 pin printer beware: although it obeys many of the same control codes as the FX-80, it is not compatible. In particular the vertical movement codes move the paper in increments of 1/60th of an inch instead of 1/72nd. In this case you need to look for names like Epson LQ800 or LQ1500.

If you are not certain which driver to use, prepare a test file using many of the printer's features and then print it out using a variety of different drivers. If you are still uncertain, a good way of finding out exactly what codes your printer is sending is to use 'Hex Dump mode'. Refer to your printer manual to get this: usually you have to turn the printer off and then hold down one or more buttons whilst turning the power back on. The printer then prints out the hex values of each code it receives instead of acting upon them, which reveals the codes that your printer is receiving. Compare these with those in your printer manual, and you may be able to work out why your printer switches out of NLQ mode every time that you start to print a letter! ●

T H E
**HELP
SCREEN**
C O L L E C T I O N

53.



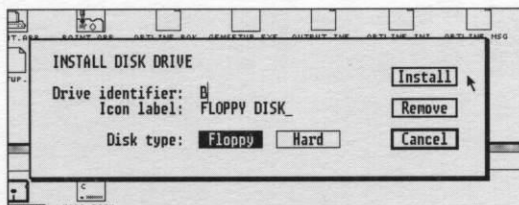
G E M & B A S I C 2

84. Configuring The Desktop

You don't have to use GEM *Desktop* as it comes. Instead you can customise it to give the facilities you want:

1. INSTALLING DISK DRIVES

Click on the top-left corner of one of the two Desktop windows until you are at the top level of the directory tree – the window labelled 'Disk Drives:'. Make sure the drives shown correspond to the drives on your machine, remembering that the hard disk icon is used to represent a RAM disk as well.



To remove an icon:

- click once on the offending icon to highlight it,
- pull down the **Options** menu,
- select 'Install Disk Drive' – an **INSTALL DISK DRIVE** dialogue box will appear,
- click once on 'Remove' to remove that icon from the Desktop.

To add an icon:

- click once on any of the existing icons to highlight it,
- again, select 'Install Disk Drive' from the **Options** menu.
- The cursor will be at **Drive Identifier** on the dialogue box, so type the drive letter [A], [B], [C] or whatever, followed by [return].
- The cursor moves to **Icon Label**, so enter a suitable label for the icon that is to represent the new drive.
- Select the type of icon by clicking on **Floppy** or **Hard**,
- click on **Install** to have the new drive appear in the window.

2. CONFIGURING APPLICATIONS

Application programs, whether GEM or straight MS-DOS, can be configured so that double-clicking on a document automatically loads the appropriate program – provided the document is in the same folder as the application, or can be found through a previously set PATH parameter. You can also select alternative icons to better represent the nature of application and document:

- Click once on the required EXE, COM or APP program file to highlight the application.

- Select 'Configure Application' from the **Options** menu, or press [Alt][A] – a **Configure Application** dialogue box will appear.

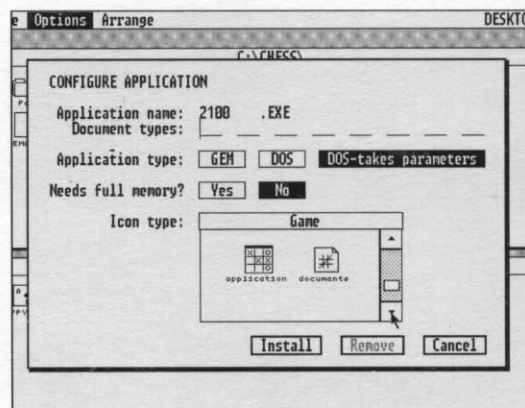
- The cursor will be at **Document Types**. Enter the filename extensions of the document files you want associated with the application being configured: TXT and DOC for a word processor, for example, or DBF and DAT for a database, perhaps. Press [Tab] after each one.

- Set **Application Type** to **GEM** if the **Application Name** has the extension APP. If it is COM or EXE, set it to **DOS** unless you wish a further dialogue box to come up every time you run the application, requesting further parameters.

- Set **Needs Full Memory** to **No** initially. Set it to **Yes** if after leaving the application there are problems re-loading the Desktop.

- Use the scroll bar on the right of the **Icon Type** window to select matching icons to represent the application and associated document. There are quite a range available, but if you want to create your own you will need the 'Icon Editor' which is just a small part of the *GEM Programmer's Toolkit* – contact Digital Research for further details.

- Click on **Install** to return to the Desktop.



85. GEM Paint on one disk

The disks supplied with the Amstrad PC1512 required users to swap three disks before they could start using *GEM Paint*. The following allows owners of twin floppy drive machines to get *Desktop*, *GEM Paint* and *Output* on to one disk so only the GEM start-up disk needs to be loaded first. There's no need to write batch files and the Clock, Calc and Snapshot utilities are all included. Drive B is used for the image files.

Using either GEM desktop or the operating system proceed as follows with the first of two empty formatted disks (no system files):

1. Open folders or make directories called GEMSYS, GEMAPPS and GEMDESK.
2. In the GEMAPPS folder (or directory) open (or make) another called PATTERNS.
3. Into the GEMSYS directory copy all files from the GEMSYS directory of Master Disk Three and the two OUTPUT files and one DEFAULT file from GEMSYS on Master Disk Two.
4. Into GEMDESK copy all files in GEMDESK on Disk Three.
5. Into the Root directory copy the two PAINT files from GEMAPPS on Disk Four.
6. If you have already generated some pattern files copy them into the PATTERNS directory in the GEMAPPS folder.
7. For the second (images) disk open one folder called IMAGES and copy into it the contents of the IMAGES folder on Disk Four.

This completes setting up the two disks. If you haven't already done so, rename SNAPSHOT as SNAPSHOT.ACC in the GEMBOOT folder on Disk Two. This will make the utility available under the PAINT menu.

To start *GEM Paint* from scratch, first load the GEM start-up disk (Disk Two) in drive A and let it boot up. Replace it with the new *Desktop/Paint* disk when prompted and while that is loading insert the IMAGES disk in Drive B. The 'A:' window of DESKTOP shows PAINT.APP ready for loading, and behind the scenes OUTPUT is ready to send Image files to the printer. To load an IMG file (once PAINT has been selected) select 'OPEN' from the FILE menu and in the 'Item Selector' window alter the Directory to read B:\IMAGES*.IMG. Hereafter Images will automatically be loaded and saved in the right directory.

87. Printing GEM screens

The standard Amstrad issue of GEM comes with a print driver routine designed for Epson printers, which is incompatible with many other models. GEM, like many programs, uses the driver to send initialisation codes for graphics mode and line spacing which override any previous settings achieved by DIP switches and software, so there's no way you can run a 'setup' program then go into GEM.

Digital Research produce *Fonts and Drivers* packs to allow GEM to be used with other printers, including the Epson JX80, IBM Colorjet and Diablo C150 and C630, and some printer manufacturers claim 'Epson' or 'IBM' compatibility. The only way to prove this is to connect it to your PC and try it out.

A rather different approach is to use the MS-DOS GRAPHICS program, which is designed to allow the printing of screen graphics when [Shift] and [PrtSc] are pressed. Try this:

1. Load MS-DOS.
2. Enter `graphics /r`.
3. Insert your GEM Startup disk and load GEM by entering `gem`.
4. You can now dump any GEM screen to printer using [Shift] with [PrtSc].

You may find it necessary to use the MODE command after stage two to set the printer to eight lines per inch:

`MODE LPT1:,8`

Why the `/r` switch on the GRAPHICS command? GRAPHICS is a horny-handed old MS-DOS retainer, introduced in the days when backgrounds were dark and foregrounds were lighter – long before GEM, *BASIC 2* and all that other fancy paper-white stuff. Consequently it 'helps out' by reversing what it thinks would be all that sombre background into lighter

86. Setting Desktop Preferences

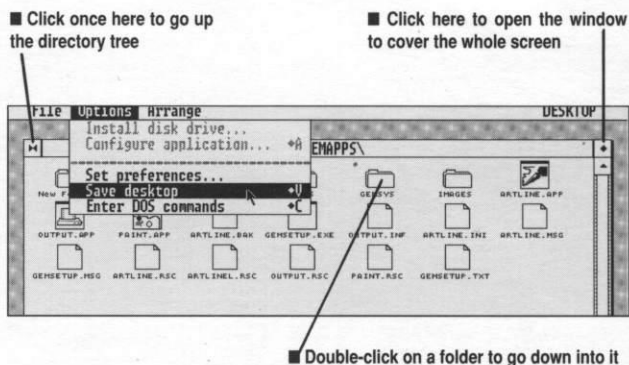
You should now set up the default display that appears when you boot up – which directory the window/s open onto:

Remember that a window that opens onto the contents of a floppy disk drive requires a disk in that drive if you are to avoid an error message – have the bottom window showing 'Disk Drives' and the top window showing the root directory of drive A on a floppy machine, or any directory of drive C if you have a hard disk model.

Select whether the files are shown with icons or with text, and the order in which they are shown, using the Arrange pull-down menu.

Select Set Preferences from the Options menu and make any alterations required.

Finally, save your settings with Save Desktop from the Options menu, or by pressing [Alt][V]. The settings are saved in the text file DESKTOP.INF in the \GEMDESK sub-directory.



shades. You can however reverse the reversal using the /r switch.

88. GEM Paint on two disks

The Amstrad PC1512 has four system disks, three of which are needed to boot up *GEM Paint* from floppy disks – a laborious process at the best of times. The following allows you to boot up *GEM Paint* using just two disks, one in drive A and one in drive B. Below is a list of which files should go in which subdirectory of which disk, and the AUTOEXEC.BAT file for the disk that is to go in drive A:

AUTOEXEC.BAT file:

```
ECHO OFF
\AUTO\NVRPATCH
IF NOT EXIST C:\COMMAND.COM COPY
\COMMAND.COM C:\ >NUL
SET COMPSPEC=C:\COMMAND.COM
PAINT.BAT file:
ECHO OFF
MOUSE
PATH B:\
CD \GEMSYS
IF "%1" == "" GOTO NOPROG
IF EXIST B:\IMAGES\%1.IMG GOTO PROGEXIST
ECHO Image not found
GOTO EXIT
:PROGEXIST
GEMVDI PAINT.APP B:\IMAGES\%1.IMG %2 %3
GOTO EXIT
```

FILES ON DISK FOR DRIVE A:

```
Root directory:
COMMAND  COM
RPED     EXE
MOUSE    COM
AUTO     <DIR>
GEMBOOT  <DIR>
GEMSYS   <DIR>
AUTOEXEC BAT
PAINT    BAT
AUTO sub-directory:
NVRPATCH EXE
GEMBOOT sub-directory:
GEM      EXE
GEM      RSC
SNAPSHOT ACC
SNAPSHOT RSC
GEMSYS sub-directory:
AMSLSS10 FNT
AMSLSS14 FNT
```

```
AMSLSS18 FNT
AMSLTR10 FNT
AMSLTR14 FNT
AMSLTR18 FNT
OUTPUT   APP
GEMVDI   EXE
DEFAULT  OPT
OUTPUT   RSC
AMSTRAD  SYS
ASSIGN   SYS
METAFIL6 SYS
```

FILES ON DISK FOR DRIVE B:

```
IMAGES    <DIR>
SNAPSHOT  <DIR>
PAINT     APP
PAINT     RSC
```

```
:NOPROG
GEMVDI PAINT.APP
:EXIT
PATH;
CD\
```

89. Printing from BASIC 2

To print text from within *BASIC 2* you use the PRINT command with stream #0, or use the LPRINT command. By default, both of these print to the DOS device LPT1. To use other DOS devices you need to first assign one of *BASIC 2*'s streams to that device. This is done with the command.

OPEN #stream PRINT device

where *stream* is the number of the stream to be used in future PRINT commands and *device* is the number corresponding to the DOS device as follows:

```
0 PRN 1      LPT1
2 LPT2 3     LPT3
4 COM1 5     COM2
```

As with all of *BASIC 2*'s streams, you should quote the stream number in all output statements which do not use the default device. You can change the default stream at any time with the STREAM command.

Graphics can be printed on a matrix or laser printer either by copying the screen to the printer using the MS-DOS screen dump feature or by using a printer as a graphics device from within *BASIC 2*. While using the screen dump is simpler it only gives output with the same resolution as the screen. Using a printer as a graphics device can give far superior quality.

But as each point in the image has to be stored in memory, using this technique takes up lots of memory. It is therefore essential that you provide *BASIC 2* with as much memory as possible in order to achieve the best results. Even then, MS-DOS and GEM themselves take up so much memory that you may not have sufficient memory to produce the results you want. To give *BASIC 2* the maximum amount of memory take the following steps:

1. Upgrade your PC to 640K of memory:

Instructions for doing this to an Amstrad PC1512 are detailed in the 'Hardware' section of this book).

2. Remove all desk accessories:

Load GEM Desktop in the normal way. Insert your GEM start-up disk (or select your hard disk) and open the GEMBOOT folder. You will see a number of files displayed with a filetype ACC. Select each of these in turn by clicking on it once, pull down the File menu and click on Info/Rename. Change the name to a file type

other than ACC (for example just delete the final C so it has the type AC).

Check that all the accessories have been removed by restarting GEM, and pulling down the DESKTOP menu. No accessories should be listed. If there are still some listed then check through all the folders on your start-up disk for other ACC files and treat them in a similar way. When you want to reinstate the accessories, simply restore their names to have filetype ACC.

3. Remove all unnecessary printer fonts:

First make a copy of ASSIGN.SYS, which you will find in the GEMSYS folder, calling it something like ASSIGN.OLD. This gives you a back-up in case of disaster.

Use a text editor (or a word processor that can read and write ASCII files) to edit ASSIGN.SYS. You can even use the BASIC 2 editor for this – it doesn't matter that it is not a BASIC program! Move down the file to the section which starts with a number in the printers range (21 – 30) and a description of the printer you are going to use. For example:

```
21 epsmonh6;ibm/epson GRAPHICS printers high res...
;printer parallel port #1 ( lpt1: )
```

If you can't find a section corresponding with your printer, stop now. You will need to add details of your printer at this point and you will need to install a compatible 'printer driver' for GEM to use. If there is a section corresponding with your printer, all is well! Remember the number used to start the section (most likely 21, but it could be anything in the range 21-30). You will need to quote this as the device-number in BASIC 2's OPEN command.

Following this, there is a list of 'fonts', taking the form:

```
epshss10.FNT;epson hi res (120x144 dots/in) swiss 10 PT
```

Remove as many of these as you can. Ideally, choose to use a single style, such as SWISS and a single small point size such as 7 or 10. You will then have that size and style of text available on your printer, plus the same font at twice the size.

Now save the file back as your new (smaller) ASSIGN.SYS and you are ready to run BASIC 2 using the printer for text and graphics.

4. Run BASIC 2 from MS-DOS, not from DOS Plus, and remove any RAM disk that might be present by deleting any line reading `device=ramdrive.sys nnn` from your CONFIG.SYS.

To test BASIC 2 can now print graphics, try running this program:

```
OPEN #5 DEVICE 21
CIRCLE #5, 2500;2500, 2000
PRINT #5, "Hello there"
GRAPHICS #5 UPDATE NEW
```

This assumes that the device-number of your graphics printer is 21. If it isn't, substitute the number you remembered above.

If this program runs it should produce the text 'Hello there' and a circle on your graphics printer. If it produces a GEM error in the Dialogue window you need to free off more memory. It is likely that you have some other software which is in memory or have not been brutal enough in following all the above steps. Check that the printer font you kept in ASSIGN.SYS has a small point size (say 10 or 7) – the larger the point size, the more memory a high resolution printer font requires.

Once you have succeeded in making this short test work, you can go on to write BASIC 2 programs which use the printer effectively. Anything you can display on a graphics virtual screen you can print on the graphics printer. However, you do need to start with the OPEN DEVICE command and must use the stream number (or change the default stream) in all your graphics output commands.

Also, nothing will be printed until you issue the GRAPHICS UPDATE command, at which point the whole page will be printed. Unlike a screen (or a plotter), you can't show each command as it happens – printers just don't work that way.

The final thing to check is that you are working with the coordinates you expect. A screen is normally wider than it is high and so by default is about 5000 units high and 8000 wide. The printer on the other hand is longer than it is wide and so is only 5400 units wide. Therefore you may need to use the USER SPACE command to set printer coordinates to match those on the screen. For the default screen dimensions and an IBM PC Graphics Printer, you can do this with:

```
USER #print-stream SPACE XVIRTUAL (#screen-stream)
```

If you have a different printer or have changed the aspect ratio of the screen, you will need to experiment with other values to get the desired result.

90. The Mouse in BASIC2

When running GEM's BASIC2 the Amstrad PC1640, or indeed any EGA machine, uses the 640 by 350 pixel screen rather than the 1512's 640 by 200. This means that programs written for the PC1512 can run into problems on a 1640. This is particularly true of the mouse, as its coordinate system is dependent on the screen coordinates.

The problem can be solved using the function YDEVICE which returns the screen height, 350 on an EGA machine; 200 on CGA. However YDEVICE only works in Stream £1, not in Stream £2. It can be used in Stream £2 with:

```
STREAM #1
LET M_size = YDEVICE
```

and then using M_size in Stream £2.

The function YMOUSE returns the height coordinate of the mouse pointer position. The following can help make this machine independent:

```
LET Y_MUS = YMOUSE * 350/YDEVICE
  (to convert to EGA coordinates);
LET Y_MUS = YMOUSE * 200/YDEVICE
  (to convert to CGA coordinates).
```

These problems do not arise with XMOUSE as both screens are 640 pixels wide. Horizontal menus may therefore be an easier choice when choosing colours, patterns, etc.

91. Keyboard Shortcuts

Although GEM Desktop is usually used with a mouse, the following keys, used in combination with [Alt], perform functions available from the menus:

- [I] to rename selected file, or set it **Read-Only**,
- [D] to delete selected files/folders,
- [A] to configure selected application.
- [V] to Save the Desktop,
- [C] to enter DOS commands directly (enter **exit** at the MS-DOS prompt to return to the Desktop.
- [S] to switch file display between icons and text,
- [N] to display files in alphabetic order,
- [P] to order files by type,
- [S] to order by file size,
- [D] to put files in date order (useful for WORK folders).

You can also select more than one file/folder at once by holding down [Shift], or the right mouse button, while clicking with the left.

92. Machine Independent BASIC2

Despite the fact that BASIC2 is supposed to enable the writing of 'machine independent' programs, many programs written on an Amstrad PC1512 do not work properly on a PC1640, giving error messages such as 'coordinate outside virtual screen'. This is usually because the programmer has not made proper use of the virtual screen

BASIC2 uses two separate coordinate systems for addressing graphic devices, whether they be printers, screens or whatever: Device Coordinates, which are specific to the device; and User Coordinates, which are not because they refer to an ideal or 'virtual' screen and rely on GEM to map this screen as best it can to the device in question. User coordinates are the more sensible ones to use, as graphics drawn under this system transfer to a CGA, EGA or Hercules screen, or indeed a printer, with impunity.

User coordinates treat the screen as a rectangle whose shorter edge is 5000 units long. The longer edge is in proportion to its displayed shape, but a full size screen is around 8000 units wide. Because this coordinate system is rather fine when it comes to PC graphics, it can be redefined with the command USER SPACE.

Device coordinates treat the screen as a grid of pixels, so the coordinates of a given point depend on the resolution of the graphics adaptor being used. The size of the screen in device coordinates (or pixels) is returned by three BASIC2 functions. Values returned for various adaptors are as follows:

Adaptor	XDEVICE	YDEVICE	YUSABLE
PC1512	640	200	189
EGA	640	350	333
Hercules	720	340	331
VGA	640	480	461

The difference between YUSABLE and YDEVICE allows for the BASIC2 menu bar at the top of the screen, which cannot be addressed in user coordinates.

So graphics drawn using user coordinates should translate fine from one adaptor to another. The problem arises when mixing device coordinates with user coordinates, as when using commands such as WINDOW SIZE and WINDOW PLACE to open a window, or SCREEN GRAPHICS to define a graphic screen; all these commands require device coordinates to specify size and position.

The mouse position is also returned in device coordinates by XMOUSE and YMOUSE. There is a sound reason for this, as the mouse pointer may be anywhere within the physical screen, and not just in the area covered by the virtual screen. However this too can cause problems across machines.

What programmers should do is translate between device and user coordinates as required to create a machine-independent program, and BASIC2 does provide the means to do this. The size of a device pixel in user coordinates is returned by the functions XPIXEL and YPIXEL, for example. The full algorithm for converting a mouse position from device coordinates to user coordinates is:

```
(xmouse-xplace)*xpixel+xscroll  
(ymouse-yplace)*ypixel+yscroll
```

Similar methods can be used for windows and screens.

The problem actually goes deeper than this, as text fonts translate to marginally different sizes on different screens. Nevertheless, programs which address the screen entirely with user coordinates, and translate mouse positions accordingly, should work reasonably well on any machine. For further details see the *BASIC2 User Guide* available from Locomotive Software on (0306) 740606.

93. How GEM Works

The GEM Graphics Environment is layered like an onion. It is made up of several discrete components that inter-relate in a complex manner to give both power and flexibility. It is also designed to be machine independent, and future-proof (whatever that means). If you have an Amstrad PC1512 or 1640 a quick DIR of Disk 2 and 3, which hold the bulk of the GEM system, reveals that it involves many different files, while TYPEing some of the BATch files is an education in the art of batch file design. But remember that much of the complexity is because of Amstrad's desire to cater for two operating systems (MS-DOS and DOS PLUS): it is not as complicated as it looks.

The first layer of the onion is the file GEMVDI.EXE, which loads the 'Virtual Device Interface'. To avoid the vagaries of the actual screen, keyboard, printer or whatever that happen to be in use at this particular time on this particular machine, this creates an idealised, imaginary device for applications to deal with.

GEMVDI.EXE is to be found in the \GEMSYS directory (with GEM version 2 at least). Also in this directory are the various device drivers that deal with the Amstrad PC's actual interfaces. On an Amstrad PC1512 the special 16-colour screen display is handled by AMSTRAD.SYS, while on other machines it is handled by one of a number of drivers, depending on the display fitted. Which one is selected is determined by the clever use of the environment parameter DISPLAY – take a look at the batch files for further details.

The printer driver supplied with Amstrad machines is EPSMONH6.SYS, backed up by a number of screen and printer font files for both the Swiss and the Dutch typeface. This driver caters for most Epson compatible printers but is limited in that it is black or white only – it won't print grey scales. Details of the drivers that are loaded are held in the file ASSIGN.SYS, which is a straight text file. Many other

device drivers are available, and can be loaded either with the GEM application *Setup* or by editing this file directly. Further details from Digital Research.

Once the interface has been sorted out the file GEM.EXE, found in the \GEMBOOT directory, is loaded. This provides AES or 'Application Environment Services', giving WIMP support to applications with Windows, Icons, Mice and Pull-down menus.

That is all there is to GEM itself, since GEM is simply an environment in which applications can run. The GEM *Desktop*, which everyone associates with GEM, is simply a GEM application like GEM *Paint* or GEM *Draw*. It is however the next program that is run if nothing else has been specified, and is to be found in its own \GEMDESK directory. Like most applications it consists of two files, DESKTOP.APP and DESKTOP.RSC. This latter 'Resource' file consists mainly of the messages, dialogue boxes and other 'conversation' elements of the application, and is separated from the main program file so that alterations can be made to these independently. It also means that these messages can easily be translated into other languages.

Two other files are to be found in this directory. DESKLO.ICN – DESKHI.ICN on EGA machines – supplies the icons featured on the Desktop, DESKHI.ICN for high resolution displays and DESKLO.ICN for lower resolution. DESKTOP.INF holds the defaults for the desktop display, and like ASSIGN.SYS is a text file so it can be examined with the MS-DOS command TYPE.

You are provided with three other applications on a PC1512, each of which lives in a separate directory. In \GEMSYS is OUTPUT.APP and OUTPUT.RSC, which comprise GEM's powerful output program; plus DEFAULT.OPT which holds *Output's* default settings. In its own private directory, is BASIC2.APP and .RSC, the BASIC2 programming language. PAINT.APP and .RSC are in the \GEMAPPS directory. Unless otherwise specified all GEM application programs should be put in this 'GEM APPLICATIONS' directory.

Finally come the Desktop Accessories, summoned from the menu to the extreme right of the menu bar. These are 'resident' utilities that can be summoned from within another program, and are loaded along with GEM itself. The relevant files are in the \GEMBOOT directory, and are SNAPSHOT.ACC and .RSC, and CALCLOCK.ACC. *Snapshot* takes up a fair amount of memory, so frequently the extension .ACC is left off to prevent it loading. RENAME it if you want to use it.

The above applies to version 2 of GEM. Version 3 works on similar principles but is structured differently.

94. GEM on a Hard Disk

On an Amstrad PC1512 the GEM system comes on several floppy disks and is designed to run under the DOS PLUS operating system. To make matters worse, if you have a hard disk model the system is set up to offer you a choice of operating systems before you get to GEM at all. Far better to run GEM under MS-DOS, as this gives you more space, and to re-format your hard disk under MS-DOS. Once this is done you can install GEM in a more sensible fashion.

The first step, before we get involved with GEM, is to get the contents of Disk 1 – the MS-DOS operating system and utilities – onto the hard disk. The external utilities should all go into a directory called \MSDOS, leaving the root directory virtually empty, which is the way it should be. However the root directory does need to contain a suitable CONFIG.SYS:

```
files=20
buffers=20
country=044
```

and a suitable AUTOEXEC.BAT:

```
echo off
path c:\msdos;c:\
prompt $p$g
verify on
KEYBUK
```

These can be created using `copy con <filename>`.

Your hard disk is now effectively ready for GEM, although you should copy the files BACKUP.EXE and RESTORE.EXE into the \MSDOS directory, and put PARK.COM or a similar head-parking routine into the root directory. These are on Disk 5 for the 1512, but again suitable software should have been supplied with the hard disk unit. 1640 owners will find all the necessary files in a directory called \SUPPLEME on their Disk 4.

To get GEM onto your new hard disk you simply copy the files listed below from the relevant floppies, taking care to preserve the directory structure so GEM knows where to find its component parts. You do not need to copy over all the files on the four Amstrad system disks, though, as many are only used for installation, or are only relevant to a floppy system. For example, the contents of \AUTO on Disk 2 (called \GEMSTART on Disk 1 of the 1640) are not needed

60.

This is a directory listing of the contents required from each of the Amstrad Master disks, and the sub-directories they should end up in. The files shown in plain text should be deleted.

Volume in drive A is 46001
Directory of A:\

COMMAND	COM	23612	FDISK	EXE	16830
ANSI	SYS	1699	FIND	EXE	6403
CONFIG	SYS	34	FORMAT	EXE	10989
DRIVER	SYS	1110	GRAFTABL	EXE	8194
RAMDRIVE	SYS	6566	GRAPHICS	EXE	13314
AUTOEXEC	BAT	88	JOIN	EXE	9012
GEM	BAT	46	KEYBUK	EXE	2985
GEM3	BAT	786	LABEL	EXE	2750
ANSI	COM	139	MODE	EXE	13930
APPEND	COM	1725	PRINT	EXE	8824
ASSIGN	COM	1523	RECOVER	EXE	4145
MORE	COM	282	REPLACE	EXE	4852
MOUSE	COM	7048	SHARE	EXE	8544
RTC	COM	516	SORT	EXE	1898
SYS	COM	4639	SUBST	EXE	9898
ATTRIB	EXE	8234	TREE	EXE	8588
CHKDSK	EXE	9680	XCOPY	EXE	5396
COMP	EXE	14560			
DEBUG	EXE	15647			
DISKCOMP	EXE	3808			
DISKCOPY	EXE	4096			
EDLIN	EXE	7356			
EXE2BIN	EXE	3050			

Volume in drive A is 46003
Directory of A:\

BASIC2	<DIR>	
GEMAPPS	<DIR>	
GEMDESK	<DIR>	
GEMSYS	<DIR>	
DOODLE	APP	28672
DISKCOPY	COM	8832
NVR	EXE	8596
RPED	EXE	4612
DOODLE	RSC	2388

Volume in drive A is 46001
Directory of A:\IMAGES

.	<DIR>	
..	<DIR>	
PC1512	GEM	96
PC1512	IMG	25948
TIGER	GEM	94
TIGER	IMG	6671

since these files are only used for installing GEM for floppy disk machines.

There are many ways of copying over these files, but possibly the best is to use the MS-DOS external command XCOPY (now on your hard disk in the \MSDOS directory). Before starting, reset the computer so that it boots up from the hard disk. Put Disk 2 in the disk drive, make sure you are in the hard disk root directory (the prompt should look like C:\>), and enter `xcopy a:\gemboot \gemboot /S/E`. When asked, specify \GEMBOOT as a Directory rather than a Filename. Continue with each of the required directories from each of the system disks, and you will have transposed the directory structure intact – see below for details.

Finally, you need a batch file so that you can evoke GEM automatically. GEM.BAT should be in the root directory and contain the following:

```
echo off
MOUSE
path c:\msdos;c:\;c:\basic2
cd \gemsys
gemvdi %1 %2 %3
```

To run GEM simply type `gem` at the MS-DOS prompt and you will be presented with the familiar GEM Desktop. If you don't wish to use the Desktop itself, the parameters after `gemvdi` (on the last line of the batch file) mean you could, for example, enter `gem paint` to go directly into GEM *Paint*.

When you first run GEM.BAT you may well get an alarming GEM error message, offering you the choice of 'Cancel' or 'Retry'. This is because GEM is trying to read drive A and there is no disk in drive A. Don't despair – simply click on 'Cancel' and the Desktop is loaded.

But as soon as you get to the Desktop you should re-configure it for your hard disk drive. You do this from the **Options** menu; the required option, however, is only available once you highlight one of the disk icons.

So first make sure one of the Desktop windows is showing the disk drives themselves. Now select the icon marked RAM DISK, which is in fact the icon used for a hard disk. Now you can select **Install Disk Drive...** from the **Options** menu, and a dialog box appears on screen. Make sure the Drive Identifier is **C**, the Label **HARD DISK**, and the Icon Type **Hard**. Click on **Install**, select **Save the Desktop** from the Options menu and it's done for all time!

Volume in drive A is 46002
Directory of A:\GEMBOOT

```
.      <DIR>
..     <DIR>
SNAPSHOT      13312
SNAPSHOT RSC   1408
CALCLOCK ACC   28672
GEM           EXE   64000
GEM           RSC    5390
```

Volume in drive A is 46002
Directory of A:\GEMSYS

```
.      <DIR>
..     <DIR>
OUTPUT APP     54352
DEFAULT OPT     204
OUTPUT RSC    10502
METAFILE6 SYS   5120
GEMVDI  EXE     7680
AMSTRAD SYS    39936
ASSIGN  SYS     739
```

Volume in drive A is 46003
Directory of A:\GEMSYS

```
.      <DIR>
..     <DIR>
FORMAT  COM     9216
AMSLSS10 FNT   1522
AMSLSS14 FNT   2282
AMSLSS18 FNT   3238
EPSHTR07 FNT   2816
EPSHSS07 FNT   2582
EPSHSS10 FNT   4698
EPSHSS14 FNT   8546
EPSHTR10 FNT   4736
EPSHTR14 FNT   8448
AMSLTR10 FNT   1920
EPSMONH6 SYS   38912
AMSLTR14 FNT   2432
AMSLTR18 FNT   3456
METAFILE6 SYS   5120
```

Volume in drive A is 46004
Directory of A:\GEMAPPS

```
.      <DIR>
..     <DIR>
PAINT    APP    72448
PAINT    RSC    13156
```

Volume in drive A is 46003
Directory of A:\GEMDESK

```
.      <DIR>
..     <DIR>
DESKTOP  APP    46848
DESKLO   ICN    9054
DESKTOP  INF     498
DESKTOP  RSC   10332
```

Volume in drive A is 46003
Directory of A:\BASIC2

```
.      <DIR>
..     <DIR>
EXAMPLES <DIR>
PROGRAMS <DIR>
BASIC2   APP    81920
BASIC2   RSC   11546
DEMO     BAS     6277
```

GEM functions far better in 640K than it does in 512K. While you're at it you might as well upgrade with one of the many kits available – see our 'Hardware' section for further details.

95. Booting up BASIC2

You can boot straight into *BASIC2* from the MS-DOS prompt, avoiding *GEM Desktop* altogether, simply by changing the last line of the GEM or GEMSTART batch file from

```
gemvdi %1 %2 %3 to
gemvdi basic2
```

This boots up *BASIC2*, but unfortunately the 'Item Selector' from which you select the BASIC program you want to run defaults to the \GEMBOOT subdirectory. But try changing the line to

```
gemvdi basic2 \basic2\setup
```

This loads *BASIC2* as before, but then tries to load a program called *SETUP.BAS* from the sub-directory \BASIC2. Whether or not this program is actually found, it initialises the Item Selector to the \BASIC2 directory.

If you are trying to run *BASIC2* from a single floppy system you will find that, once loaded, you can happily remove the *BASIC2* program disk from the disk drive and replace it with your own work disk. *BASIC2* does not require any overlay or message files but loads everything into memory first time.

96. GEM on a Single Disk

It is possible, just, to create a 'turnkey' disk for *GEM Desktop* – that is, a single disk that will boot up from cold and automatically take you straight to the *Desktop* – although how useful this is must be debatable, as there is precious little space left on the disk for anything actually useful. However it is also possible to create turnkey disks that will boot straight into a GEM application, bypassing the *Desktop* altogether, which is rather more useful. This, for example, is how you would put together a disk that boots straight into *BASIC2* without any disk swaps:

1. Turn on the machine and put the MS-DOS Utility Disk in the left-hand floppy drive when asked. If you have an Amstrad PC1512, don't use DOS PLUS as MS-DOS takes up rather less disk space and GEM is happy under either system.

2. Label a blank disk as 'BASIC2 Boot Disk' and, if you have two drives, put it in the right-hand drive. In any case, enter `format b:/s` at the A-prompt and follow the instructions.

3. When the A-prompt reappears, enter both the line

`copy mouse.com b:` and `copy keybuk.exe b:` to copy the files that let you use a mouse and give you a UK-style keyboard.

4. Remove the MS-DOS Utility Disk and replace it with your new Boot Disk. Enter the line `md gemboot` followed by `md gemsys` and `md basic2` to make directories for the GEM files.

5. Put the blue GEM Startup Disk in the right-hand drive and enter `cd \gemboot` to move into your new \GEMBOOT subdirectory. Then enter the command `copy b:\gemboot\gem.*` to copy the two files GEM.EXE and GEM.RSC from the equivalent subdirectory of the GEM Startup disk.

6. Enter `cd \gemsys` to move into the \GEMSYS subdirectory, and enter `copy b:\gemsys\*.sys` followed by `copy b:\gemsys\gemvdi.exe` to copy four further files across.

7. Remove the GEM Startup disk and put the green GEM Desktop and BASIC2 Disk into the right-hand drive. Enter `copy b:\gemsys\epsmonh6.sys` to copy one last file into the \GEMSYS subdirectory.

8. Enter `cd \basic2` and then `copy b:\basic2.*` for the two files that make up BASIC2.

- 9 You now need to create a suitable CONFIG.SYS file. Enter `cd\` to get back to the root directory and then enter the following at the A-prompt, pressing [Return] at the end of each line:

```
copy con config.sys
files=20
buffers=20
country=044
^Z
```

You enter the last character by holding down [Ctrl], hitting [z], releasing [Ctrl] and hitting [Return].

10. Finally, enter the following to create the file AUTOEXEC.BAT:

```
copy con autoexec.bat
echo off
verify on
path a:\basic2
KEYBUK
MOUSE
cd \gemsys
gemvdi basic2
```

Terminate with Ctrl-Z as above.

The secret to this disk is in the last line of the AUTOEXEC.BAT file created in stage 10 above. This runs the program GEMVDI with the parameter *basic2*.

GEMVDI is the environment within which GEM applications run, and GEM *Desktop* is simply another GEM application like *BASIC2* or indeed GEM *First Word Plus*. It is the 'default' application which means that, in the absence of any other parameter, *Desktop* will load automatically.

However our batch file specifies *BASIC2* instead, so booting up from your new Boot Disk takes you straight into *BASIC2* without bothering with *Desktop* at all. The same principle could be used to create a disk that boots straight into *Desktop* simply by creating a \GEMDESK subdirectory instead of a \BASIC2 subdirectory at stage 4, and copying the files from the \GEMDESK subdirectory into it at stage 8. Make the last line of the AUTOEXEC.BAT file simply `gemvdi` and there you are. The same could also be done with GEM *First Word Plus*, but putting the appropriate files into a subdirectory called \GEMAPPS which is where GEM usually expects to find application files – although whether you could get it all on a single disk is another matter.

You may have room on your Boot Disk for some of the printer and screen font files to be found in the \GEMSYS subdirectory of the GEM Desktop and BASIC2 Disk. If so these can be copied over at stage 7 above. Owners of single floppy machines might find it easier to create a 360K RAM drive and copy all the relevant files into there first, before copying them onto the new Boot Disk.

Finally, these instructions only work with GEM version 2 – version 3 is organised in a rather different way.

97. Increasing Available Memory

GEM, like most such systems, needs a lot of memory. Luckily the system is fairly intelligent and will warn of 'insufficient memory', but there are ways of increasing the amount of memory available to your application.

1. Disable the pop-up accessories, such as the clock/calendar or Snapshot. GEM loads any accessory that has the file extension ACC, so renaming CALCLOCK.ACC, for example, to just CALCLOCK will prevent it taking up memory.
2. Make sure you have not loaded any other memory-resident programs, and haven't set any RAM disk. Delete the line starting `device=ramdrive.sys` if it is present in your CONFIG.SYS.
3. Some GEM applications load fonts into memory before they are actually needed. Try renaming files with the extension FNT to prevent them being identified as fonts – you can remove them all if need be, as GEM will simply fall back on its own system font.

98. Speeding Up GEM

To make GEM work faster on a 640K PC with a hard disk, try the following. It configures an 80K RAM disk into which are copied all the necessary GEM *Desktop* application, resource and icon files.

You will find that most of your applications will happily run under GEM with the slightly reduced system memory. The result is a much faster return to the Desktop when you switch between GEM applications. Here's how to do it on an Amstrad PC.

1. If you don't already have a RAM disk configured to at least 80K, use your text editor (eg RPED) to add this instruction to your CONFIG.SYS file:

```
device=ramdrive.sys 80
```

Alternatively, if your ramdisk size is stored in non-volatile memory, run the NVR utility and change RAN disk size to 80.

2. Edit GEM.BAT to make the RAM disk (drive D) the first parameter in the PATH instruction, as in:

```
path d:;%path%;c:\basic2
```

3. Include the following instructions in the GEM.BAT file, after the PATH instruction above:

```
copy \gemdesk\*.app d: >nul
```

```
copy \gemdesk\*.rsc d: >nul
```

```
copy \gemdesk\*.icn d: >nul
```

Don't include `copy \gemdesk\*.inf d: >nul` in the above, or else you will find that any changes you make to the Desktop are not be saved to the hard disk but written instead to the volatile ramdisk!

4. (Skip this step if you already have a ramdisk installed under GEM) Execute GEM as normal, and install the RAM disk as follows:

Click on and highlight one of the disk icons. Pull down the OPTIONS menu and click on the CONFIGURE APPLICATION option.

The INSTALL DISK DRIVE dialog box will now appear. Select the drive identifier D, the icon label as RAM DISK (or whatever you like) and the icon type to HARD. Select the INSTALL option and finally SAVE DESKTOP to make the changes permanent.

5. Finally, re-boot the system and run a much faster GEM Desktop.

Incidentally, it is important that you don't have GEMVDI.EXE in the same subdirectory as the Desktop files on your hard disk. When you quit a GEM application, GEMVDI goes hunting for the Desktop files in the usual DOS way: that is, it will look in the current default directory and then along the specified search path. If the search of the current directory fails, putting `D:;` at the start of the PATH statement will ensure that

the RAM disk version of Desktop is found first. If GEMVDI finds Desktop in the current directory however, it won't bother looking anywhere else; it'll just load the hard disk copy and you won't get any benefit from the RAM disk copy at all.

If you do have your Desktop files jumbled up with other GEM bits and pieces, you'll probably find it easier to rearrange them to fit the instructions above rather than vice versa.

100. BASIC2 Indexed Files

The *BASIC2* language that comes with the Amstrad PC1512 and 1640 range is quite a powerful version of BASIC, particularly in the field of graphics and, though less often noted, in its file handling capabilities.

The simplest kind of disk file that *BASIC2* can handle is a 'sequential' file. This consists of a sequence of data written to the file one item after another, which can only be read back in the same order; you cannot go back and re-read a previous item, but instead must start at the beginning again and read through. The data does not have any particular structure either, with each item treated as totally

independent to its neighbours. A typical input statement would be:

```
INPUT #file, surname$, salary
```

which would read in the two next items from the list, and assigns them to the variables *surname\$* and *salary*. It is up to you to ensure that these items are a string and a numeric value.

More powerful are 'random' files. Here the information is organised into 'records', each containing a fixed structure of 'fields'. For example, the records in a company's personnel file could consist of fields containing name, address, present salary, date due for next salary review, and so on. Records are identified by their 'record number', and can be accessed directly by simply stating the position in the file. So a record would be read with a statement like:

```
GET #file, record$ AT rec_number
```

A particular field would printed, say, with:

```
PRINT record$.salary
```

99. Bugs on BASIC 2



There are a number of bugs in *BASIC 2* version 1.12, as supplied with some earlier PC1512s, which can be fixed by running the following BASIC program. Make a copy of your disks before running this, as it patches directly to the main files. You should see the message **CORRECT** if the program has run successfully. **BREAK** indicates a mistake somewhere:

```
REM Program to patch BASIC2 Version 1.12 to 1.12B
p$="a:\basic2\basic2.":REM Path & filename

REPEAT:READ c$:FOR i=1 TO LEN(c$)
  s=(s+ASC(c${i}))*2:s=(s+s\1000)MOD 1000
NEXT:UNTIL c$="e":IF s<>182 THEN STOP
RESTORE:RECORD b;a [0 TO 127]UBYTE
REPEAT:READ c$
  IF c$="o" THEN READ f$:OPEN #9 OLD RANDOM p$+f$
  IF c$="c" THEN CLOSE
  IF c$="g" THEN READ r$:GET #9,r$ AT r
  IF c$="p" THEN PUT #9,r$
  IF c$="x" THEN READ f,l:FOR o=f TO l:READ r$.b.a[o]:NEXT
UNTIL c$="e"
DATA c,o,app,g,1,x,18,19,13,175,p
DATA g,310,x,85,86,64,4,p
DATA g,319,x,23,28,232,198,250,233,73,251,p
DATA g,324,x,73,73,65,x,75,78,84,2,137,4
DATA x,87,90,138,236,50,232,x,92,92,196
DATA x,94,97,3,233,19,7,x,99,103,30,7,233,92,110
DATA x,105,116,247,229,93,137,84
```

```
DATA 2,247,218,112,14,127,20
DATA x,120,120,212,x,122,127,216,131,218,0,235,202,p
DATA g,325,x,1,13,11,192,117,4,11,201
DATA 120,196,247,218,232,188,255
DATA x,61,61,171,x,87,87,147,p,c
DATA o,rsc,g,13,x,127,127,66,p
DATA g,14,x,2,3,50,57,x,5,7,77,97,121
DATA x,10,10,55,p,c,e
```

```
REM Program to checksum BASIC 2 Version 1.12B
p$="a:\basic2\basic2.":REM Path and filename
```

```
f$="app":GOSUB check:IF s<>28e0d839 THEN STOP
f$="rsc":GOSUB check:IF s<>610abee0 THEN STOP
?"CORRECT":END
```

```
LABEL check
```

```
CLOSE:OPEN #9 OLD RANDOM p$+f$
RECORD w;a[0 TO 63]UWORD:s=0
WHILE NOT EOF(#9):GET #9,r$:FOR i=0 TO 63
  s=(s+r$.w.a[i])*2:s=(s+s&3fff0000)MOD &3fff0000
NEXT i:POSITION #9,NEXT:WEND:CLOSE
```

```
RETURN
```

This should only be run with version 1.12 of this program. You can check the version number by clicking on 'About BASIC 2...' under the 'BASIC2' option on the menu bar.

These two types are fairly standard to high-level languages on microcomputers, but *BASIC2* also supports the even more powerful 'keyed' file format, similar to the ISAM file format used by mini and mainframe computers.

To understand keyed files it is first necessary to understand the main limitation of random files, which is that records can only be found if you already know the record number. The records in keyed files are linked to 'keys' that are held in separate indexes, automatically sorted in numeric or alphabetic order as appropriate. Upto 20 independent indexes can be linked to one keyed file, effectively allowing the records in that file to be accessed on 20 different criteria. This means that the actual order of the records in the file itself becomes completely irrelevant. The personnel file described above could be indexed on name, salary, review date and so on as appropriate to the program's requirements.

The actual record required would be accessed with a GET statement in the same way as with a Random file, however a statement along the lines of:

```
rec_number = POSITION #file "Fortescue" INDEX 3
```

Would return the record number of the first record containing 'Fortescue' on INDEX 3. The same effect could of course be achieved with Random files using a loop, however the Keyed file would do it a lot quicker.

Keyed files are a powerful concept that need to be used if you are to grasp their full potential, and are best suited to writing large business packages. However *BASIC2* running under GEM is not an ideal environment for this kind of package. The best book on the subject if you do want to get into them, and vital reading for anyone learning *BASIC2*, is Locomotive Software's own 'BASIC2 User Guide'. Phone (0306) 740606 for details. ●

T H E
**HELP
SCREEN**
C O L L E C T I O N

67.



W I N D O W S

The main attraction of *Windows* is its apparent simplicity. Like the Apple Macintosh's *Finder* interface, it is based on work carried out in the early seventies at Rank Xerox's PARC facility to develop a 'user-friendly' face for the computer, and allows the user to control the computer with just a mouse, a pointer and pull-down menus. However underneath this face is a complex system of linked programs, much of which can be tailored by the user to his or her specific requirements.

101. Controlling MS-DOS Applications

Any application running under *Windows* can be displayed in a window or as an icon. The window can be toggled between one filling the whole screen, or to the size and shape last set by the user. Non-*Windows* applications can also be run full-screen, just as they would appear when loaded from the MS-DOS prompt. The following keystroke combinations, together with the Control Menu, can be used to control applications running under *Windows*:

- [Alt] [Esc]** Holding down [Alt] and repeatedly tapping [Esc] rotates through the icons and windows. Non-window applications running full-screen are reduced to icons.
- [Alt] [Shift] [Esc]** Rotate in reverse order.
- [Alt] [Tab]** Holding down [Alt] and repeatedly tapping [Tab] rotates through the title bars of each loaded application. Releasing [Alt] restores the selected application as last set by the user.
- [Alt] [Shift] [Tab]** Rotate in reverse order.
- [Alt] [Enter]** Switches between full-screen and window display modes (non-*Windows* applications only).
- [Alt] [Space]** Selects application's Control menu.
- [Alt] [PrtSc]** Copies screen into clipboard.
- Beep** Means an inactive application needs to display a status or error message. Message will be displayed as soon as you select the appropriate application.

CONTROL MENU

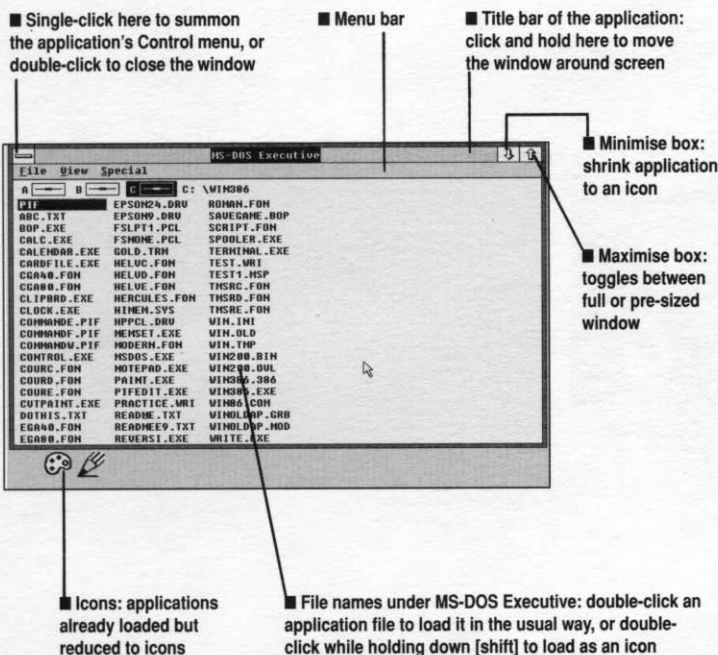
Commands can be executed either by pressing the direct-access key listed here (corresponding to the underlined letter on the menu) or by using [up] or [down] to highlight the required item and then pressing [return]. Alternatively they can be executed using the mouse in the usual way.

- Restore [R]** Restore window to size last set by the user,
- Move [M]** To move a window or icon,
- Size [S]** To change size of window,
- Minimise [N]** Shrink application to an icon,
- Maximise [X]** Toggle window between full screen and size last set by user.
- Close [C]** Closes a window and removes application from memory if it is the application's main window.

*The following commands only appear on the Control Menu of non-*Windows* applications, as clipboard functions usually appear under the Edit menu of applications written for *Windows*:*

68.

102. Controlling a Window's Window



Mark	[K]	Mark an area for copying into the clipboard,
Copy	[Y]	Copy marked area into the clipboard,
Paste	[P]	Paste contents of clipboard at cursor.
Scroll	[L]	Scroll application within window.
Settings	[T]	Brings up dialog box for controlling the multi-tasking of the application (<i>Windows/386</i> only).

DIALOG BOXES

Used to select files or from a set of options:

[Tab]	Rotate through items or groups of options.
[Shift] [Tab]	Rotate in the other direction.
[up] or [down]	Move within a group of options.
[Alt] [letter]	Execute item marked on menu with underlined letter.
[space]	Select item (if not already selected).

103. Controlling Windows Applications

To use a pull-down menu from the keyboard, first press [Alt] to select the menu bar then either press the direct-access key (that corresponding to the menu's underlined letter) or use [left] and [right] to highlight the required menu. Commands can then be executed either by tapping the appropriate direct-access key or by using [up] or [down] to highlight the required item and then pressing [return]. Menu operations can be abandoned by pressing [Esc].

Many applications written to run under *Windows* adhere to certain conventions when it comes to pull-down menus and direct-access keys, particularly the 'File' and 'Edit' menus:

FILE MENU

Can usually be summoned with **[Alt] [F]**, followed by the appropriate direct access key:

[N]	Open a new untitled document,
[O]	Open a file selected with a dialog box,
[C]	Close a window,
[S]	Save a file under its existing name,
[A]	Save file under a new name,
[P]	Print a file (using the print spooler),
[X]	Exit and remove application from memory.

EDIT MENU

Can usually be summoned with **[Alt] [E]**, followed by the appropriate direct access key. Alternatively some of these commands can be executed directly using the appropriate keystrokes:

[U]	[Alt] [Bksp]	Undoes last action (if possible),
[T]	[Shift] [Del]	Cut information into the clipboard,
[C]	[Ctrl] [Ins]	Copy information into the clipboard,
[P]	[Shift] [Ins]	Paste contents of clipboard at insertion point, or in place of selected text,
[L]/[R]	[Del]	Clear the clipboard altogether,
[S]/[A]		Select the whole document.

104. Version Numbers

There are a number of different versions of *Windows* available today, each offering different features:

Version 1 is the original version (as you might have guessed) and has now been superseded; however it can still be found bundled with some AT compatibles. It comes with RAMDRIVE.SYS, which can turn expanded or extended memory into a super-fast RAM disk into which programs can be swapped from conventional memory. Programs so swapped can be swapped back into memory, for you to carry on where you left off, far quicker than from a hard disk drive.

Version 2 is the current version. It changes the way the windows work on screen and provides 'direct-access' keys to make it easier to use from the keyboard. It offers considerably faster graphic routines too – up to 10 times faster for some operations.

It also allows certain applications to be run in expanded memory, and provides SMARTDrive which can use extended memory for an intelligent disk cache. Programs that have been written specifically to run under *Windows 2.0* or higher can handle more than one document open at a time, too.

Windows/286 is very similar to version 2.1 but adds considerably more printer drivers and the ability to run under MS-DOS version 4. It is tailored to make particular use of certain features offered by Intel's 80286 processor, as found in the IBM AT and compatibles. Specifically it provides HIMEM.SYS which allows a portion of *Windows*' own program code to be loaded into high memory (above the 640K barrier imposed by MS-DOS), so freeing an extra 45K of conventional memory for application programs.

Windows/386 is tailored to take full advantage of the 'virtual 86' mode of Intel's 80386 processor, allowing far more versatile multitasking of standard applications (programs designed to run directly from MS-DOS rather than under the *Windows* environment). *Windows/386* loads a special version of *Windows* version 2 into a first virtual machine, and then creates

separate virtual machines in extended memory for each standard application as it is loaded.

Applications written to run under *Windows* are loaded into the first virtual machine under *Windows* version 2 in the normal way, and are swapped out to expanded memory when more space is required. *Windows/386* can turn extended memory into expanded memory using the virtual addressing abilities of the 80386 processor, either for running *Windows*-based applications or for standard applications that can use expanded memory.

Windows/386 interacts dynamically with the SMARTDrive disk cache, increasing or reducing the amount of memory it uses according to overall demand.

If you have a PC or XT compatible, using an 8088 or 8086 processor, then either version 1 or version 2 of *Windows* can be used although the later version is to be preferred. *Windows* is rather slow on these machines though.

If you have an AT compatible you can still run both these versions of *Windows* and they become far more usable. You would, however, be better advised to use a version specifically tailored to your machine's processor. *Windows/286* is quite happy on either the 80286 or the faster 80386 processor, but *Windows/386* can only run on a machine using the 80386.

As far as graphics adapters are concerned, all versions of *Windows* require either a Hercules, EGA, MCGA or preferably a VGA display. In CGA the 640 by 200 monochrome mode is used, but the display is not really satisfactory. A hard disk is a must for serious use.

105. Using Memory

Any version of *Windows* can 'break the 640K' barrier on any machine by simply swapping code out to hard disk when the space is needed. This allows swapping between programs, although it can take up to 30 seconds to re-load a large program.

If your machine has extended memory the swapping process can be speeded up by swapping out to a RAM disk. For example a 384K RAM disk could be created, using the driver supplied with version 2 upwards, by including the line

device=ramdrive.sys 384 /a

in your CONFIG.SYS (and making sure the file RAMDRIVE.SYS is in the root directory of your hard disk). See later section on WIN.INI for settings required to use this disk for swapping.

Alternatively you could leave *Windows* to decide what files should be stored in memory by using SMARTDrive, the disk cache driver. This can be

implemented in much the same way as the RAM disk, by including the line

device=smartdrv.sys 384

in your CONFIG.SYS (but note that the /e parameter is not required). It is worth experimenting with both drivers to see which best suits your way of working, as they cannot be used together.

Windows version 2 and *Windows/286* can use expanded memory conforming to LIM EMS version 4.0 for not only storing but also running programs. Expanded memory can be used for the disk cache too, and *Windows* will then dynamically juggle available expanded memory as the situation demands. The appropriate line for the CONFIG.SYS is

device=smartdrv.sys 384 /a

As a rule of thumb you should start by allocating around a quarter of your available memory to the disk cache.

Windows/286 will also benefit from inserting

device=himem.sys

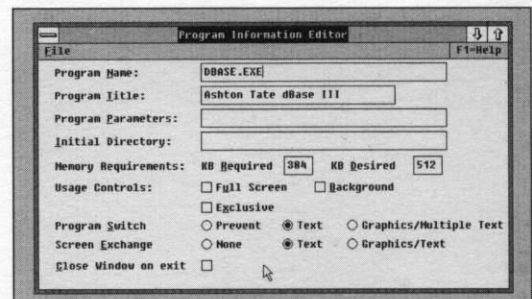
on the immediately preceding line of your CONFIG.SYS.

On an 80386-based AT compatible extended memory can be converted to expanded memory using software alone, usually using device drivers listed in CONFIG.SYS. *Windows/386* installs its own such drivers which will conflict with any already loaded, so these should be removed first. This means that either a RAM disk or SMARTDrive can be used, but it has to be loaded into extended memory as it is inevitably loaded before *Windows/386* has had a chance to create any expanded memory.

106. Running MS-DOS Applications

Although standard DOS applications can be run simply by double-clicking on the appropriate filename from the *MS-DOS Executive*, their operation can be customised by creating a Program Information File (which has the extension PIF). These can be kept in a single \PIF sub-directory, and applications loaded by clicking on the appropriate PIF. It is quite in order for a number of different PIFs to be created for a single application.

PIF files are edited using the utility PIFEDIT.EXE,



which pops up a dialogue box:

Program Name: The full file specification of the program file including path and extension, as in `c:\data\dbase.com` for example.

Program Title: The name that will appear in the window's title bar, as in **dBase III Plus**. Up to 29 a maximum of characters.

Program Parameters: Any switches or parameters that would otherwise be included in the DOS command line, such as `/n` or `/m=64`, for example. Entering ? causes *Windows* to prompt for parameters each time the program is loaded.

Initial Directory: The default directory required by the program, which would either be where the program files are kept or where data files are stored.

KB Required: The minimum amount of memory that the application requires (see documentation). If *Windows* cannot find this amount it displays the message 'Not enough memory to run' when you attempt to load it. If left blank it will try and allocate 640K.

KB Desired: The maximum amount of memory that *Windows* should allocate. If you leave blank or enter 0 *Windows* will allocate all available memory up to a maximum of 640K. If you enter -1 and the application runs full screen, *Windows* (but not *Windows/386*) will try and swap itself to disk to make room.

Directly modifies: (not *Windows/386*) Select the devices that the application accesses directly or cannot share with other applications. Check 'Screen' to run full-screen when the program has problems running in a window. Check 'Keyboard' only if the program is having problems with the keyboard. Check COM1 or COM2 if the program uses the RS232 ports or an internal modem at all. Check 'Memory' if the application loads any memory-resident utilities.

Usage Controls: (*Windows/386* only) Determines how you want the application to multitask. Check 'Full Screen' if you wish it to appear as it would from DOS, otherwise it will be loaded into a window. Check 'Exclusive' if you want other applications to be frozen when this is in the foreground, a good idea for comms packages or graphics packages that otherwise appear sluggish. Check 'Background' if you want the application to carry on running when switched to the background, although the program would still be frozen if the foreground task was in 'Exclusive' mode.

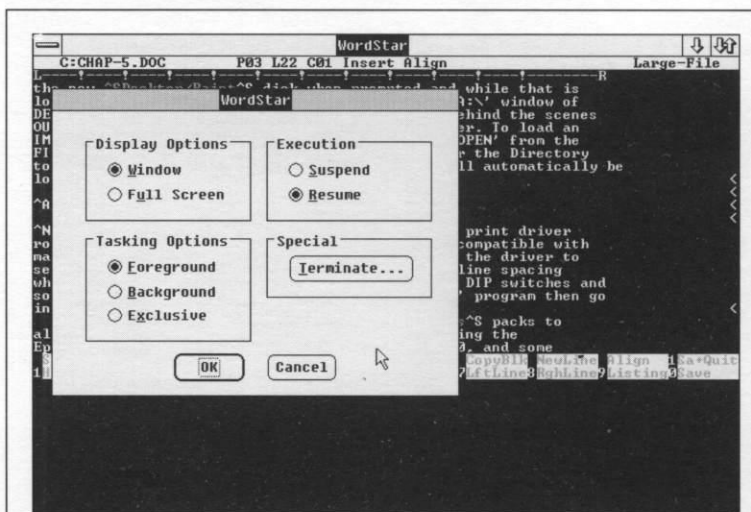
Program Switch: (ignored by *Windows/386* where you can always switch between DOS applications) These options determine whether you can switch back to *Windows* with [Alt-Esc], and determines how much memory is reserved for storing the screen image. Check 'Text' to reserve 4K, or 'Graphics' to reserve up to 36K for graphics screens or programs that store multiple text screens. Check 'Prevent' to reserve no memory at all, which means you will have to quit the application altogether to swap back to *MS-DOS Executive*.

Screen Exchange: These options determine how much memory is reserved for transferring screens to the Clipboard, with the buttons having much the same meaning as the above. *Windows/386* automatically determines how much memory is required, so only set this option to override default values.

Close Window on exit: Check this to have the window close when you quit the application, otherwise it is left open but marked 'inactive'.

107. Customising Windows

Windows' operation is also determined by the contents of the file WIN.INI, in much the same way as CONFIG.SYS tailors the operation of DOS. WIN.INI is

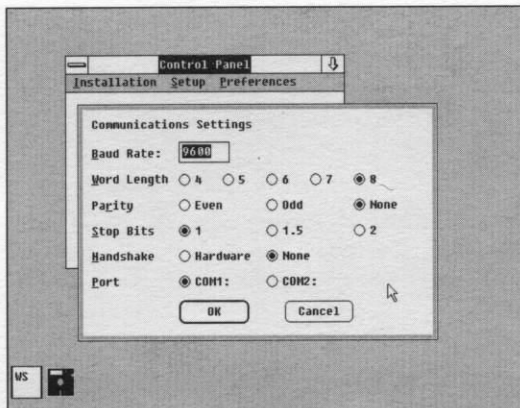


■ Under *Windows/386* an extra option appears at the bottom of the Control menu marked 'Settings'. This summons the dialogue panel shown here, which allows you to control the multitasking of the application and how it is displayed on screen as it is running. 'Display Options' and 'Tasking Options' perform the same functions as they do in the PIF Editor, while 'Suspend' and 'Resume' allow you to temporarily freeze an application so *Windows* can switch resources elsewhere. 'Terminate' should only be used as a last resort

a text file and so can be edited using any ASCII text editor, including *Windows*' own *Notepad* program; indeed double-clicking on the filename from the *MS-DOS Executive* automatically loads WIN.INI into *Notepad*.

The file is divided into sections, each headed by a single word enclosed in square brackets. Within these sections are listed various parameters settings, liberally interspersed with explanatory notes. These last are identified by a semi-colon in the first column, rather like REM statements in batch files.

Although you can edit this file directly, double-clicking on the file CONTROL.EXE loads the Control Panel which provides a friendlier face to some of the settings. However there are many others that are best altered by directly editing WIN.INI:



● Although WIN.INI can be altered directly, many of its settings are more easily changed from the Control Panel which appears when you run CONTROL.EXE

[windows]

programs=com exe bat it is a good idea to add 'pif' to this list.

load= programs listed here are automatically loaded as icons when *Windows* is started. List program or PIF filenames one after the other, assuming the extension to be one of those listed under 'programs' above, separated by a space and omitting the extension. Data files that have been linked to programs in the [extension] section below can also be listed here, but including their extension, and are automatically loaded into the appropriate application.

run= works much the same as 'load=' except that programs are loaded into windows rather than as icons.

MinimizeMSDos=yes including this line means that *MS-DOS Executive* will be reduced to an icon when another application is run, freeing a little more memory.

The line can be added automatically by selecting 'run' from the File menu of *Executive* and checking the appropriate box.

[extensions]

wri=write.exe ^.wri This means that double clicking on any file with the extension WRI causes the program WRITE.EXE to run with that file already loaded – the ^ character is a wildcard equivalent to DOS's * symbol. Any number of entries are permitted under this heading, including multiple entries for the same application. Two particularly useful entries would be:

bat=notepad.exe ^.bat

sys=notepad.exe config.sys

Note the specific filename on the last example.

[pif]

swappedisk= (not *Windows/386*) This determines which disk drive is used for swapping standard applications out of memory. For example, if you have set up a RAM drive as drive D this could be set to equal d:. It can also be set to equal ? in which case the drive and directory specified by the DOS environment variable TEMP will be used, or the lowest lettered hard disk if the variable does not exist.

Alternatively the line

swappedisk=? /e means that files will be swapped to expanded memory first, and only swapped to the specified drive when that runs out.

swapspace=0 (not *Windows/386*) Specifies in kilobytes the amount of memory reserved for swapping. If zero *Windows* will reserve the amount specified in the PIF of the first standard application loaded, which is why you should always run the largest program first.

attrib.exe=96 Next comes a series of entries devoted to MS-DOS utilities. These can be thought of as mini PIF files, each entry corresponding to the 'Memory Required' panel of the PIF file.

[win386]

Entries in this section are only relevant to *Windows/386*:

windowmemsize=640 This specifies the amount of memory allocated to the virtual machine in which *Windows* is run (see 'Version Numbers' above). If you are running applications written for the *Windows* interface this is best left at 640K, however if you are mainly running DOS applications this can be reduced down to around 384K. Too low a setting will generate a 'Not enough memory to run' message – increase in 20K increments.

emmsize= This specifies the maximum amount of memory in kilobytes you want *Windows* to set aside as expanded memory. It is probably best left blank to allow *Windows* to decide for itself, however it should

be set to equal zero of 'windowmemsize' has been set to anything other than 640K. If you are running *Windows* applications together with DOS applications that use expanded memory then experiment with figures, bearing in mind they are rounded to the nearest multiple of 16K.

altescape=on *Windows/386* uses a number of [Alt] keystroke combinations for controlling programs (see earlier in this chapter). These can be disabled by setting them to **off** in this section, should your applications use them for their own purposes. ●

T H E
**HELP
SCREEN**
C O L L E C T I O N

75.



U S I N G H A R D D I S K S

108. Disabling DOS PLUS

The original hard disk versions of the Amstrad PC1512 were configured to offer the user a choice of running under MS-DOS or DOS PLUS when first booted up. Instructions on circumventing the 'Select operating system' prompt were included in the manual, but were not that clear. The process essentially involves renaming a hidden file called DOSPLUS.SYS so that the machine boots straight into MS-DOS. However DOSPLUS.SYS is a hidden file (i.e. it doesn't appear in DIRectory listings), and for that reason it can't be RENamed. Luckily, you can overcome this using the DOS PLUS commands SDIR and FSET, as follows:

1. Boot up and select DOS PLUS. If this takes you into the GEM desktop, pull down the FILE menu and click on EXIT TO DOS.

- 2 At the C-prompt, type:

```
sdir [sys]
```

This will show you a list of the hidden files in your root directory, among which should be DOS.SYS and DOSPLUS.SYS.

3. To make DOSPLUS.SYS 'visible', type

```
fset dosplus.sys [dir]
```

4. Now change its name with something like:

```
ren dosplus.sys dosplus.old
```

and reboot the system. Your next system load should go straight into MS-DOS.

5. To reverse the process, rename the file back to DOSPLUS.SYS (while running under MS-DOS), then reboot with [Ctrl][Alt][Del] and you should get a choice of operating systems again. To make DOSPLUS.SYS a hidden file, type

```
fset dosplus.sys [sys]
```

FSET and SDIR will not run under MS-DOS, so once you've 'disabled' DOS PLUS you won't be able to make the renamed DOSPLUS.SYS hidden without resorting to more drastic measures. This means it can be DELETED, so be careful.

If by some chance you want to disable MS-DOS, the file to rename is DOS.SYS, not MSDOS.SYS which also appears in the root directory.

109. Using Tree

The following gives a listing of all directories, sub-directories and files on your disk – very useful if you have a full hard disk. It is also possible to get a hard copy of the TREE structure, but note that these will only work if the program files TREE.EXE and

MORE.COM are available, either in the current directory or through an established PATH.

At the system prompt enter **tree/f** [more] to display one screen of TREE output at a time. At the end of each screen it is possible to do a screen dump to the printer by pressing [Shift] [PrtSc], moving to the next screen and repeating the process until finished. Enter **tree/f >prn** or **tree/f >lpt1** to print a full list, but do ensure that your printer is on line. the second routine may also be used with the DIR command if you require a printout of the full directory.

Alternatively, if you press [Ctrl] and tap [P] anything that appears on screen is automatically 'echoed' to the printer. This command is a 'toggle', so if you enter [Ctrl][P] again it will switch the printer off. It can be used with anything generated from the operating system prompt, so you could simply enter **tree /f** or **dir** to get a complete printout.

110. Lost Clusters

Lost clusters are one of MS-DOS's more mysterious little beasts. You only see them mentioned by CHKDSK with something like '2 lost clusters found in 2 chains', and neither the program nor the manual are very forthcoming about what they actually are.

The smallest subdivision of a disk (hard or floppy) is a sector, which is usually 512 bytes long. However, to keep its 'map' of what files are where on a disk, MS-DOS groups these sectors into clusters. The number of sectors in a cluster depends on the type of disk – four for hard disks and two for floppies on the Amstrad PC's – and space is allocated to files in units of one cluster at a time.

A cluster can only contain one file at a time, so the minimum physical space taken by a file is one cluster. This means that small files occupy more space on a hard disk, which has 2048-byte clusters, than on floppy with 1024. The size shown on DIRectory displays, which can be as small as 1 byte, is just the recorded 'end-of-data' marker – the real space occupied is the number of bytes in the nearest whole number of clusters, rounded up.

MS-DOS keeps a map of clusters in the File Allocation Table (or FAT), which occupies a reserved space at the beginning of each disk. Each cluster on the disk has a corresponding entry in the FAT, and the entry states whether the cluster is currently allocated to a file, marked as bad, or free for use.

When a file is extended (for example, when you save a new copy of a word processing document which is larger than the old one), MS-DOS looks for a free cluster in the FAT, and 'chains' it on to the end of the file. This is done by setting a 'pointer' value in what was previously the file's last FAT entry to point to the new cluster. Thus a file which occupies a number of

clusters will have a corresponding chain of FAT entries, not necessarily adjacent to each other.

When a file is deleted MS-DOS works through this chain of FAT entries setting them all back to 'free' status, freeing the space for allocation to new files. However, this doesn't always work out as planned, and sometimes clusters get 'lost' – marked as allocated, but to files which no longer exist, or can't be traced. Lost clusters are almost always caused by a program being interrupted prematurely, usually by the user pressing [Ctrl] [Break] or [Ctrl] [Alt] [Del]. One of the worst culprits are Microsoft's compilers and linker which, if interrupted, leave behind very large chains of lost clusters.

MS-DOS won't allocate these cluster to new files, but they aren't actually doing anything, so CHKDSK offers to convert them into files for you (with names like FILE0000.CHK, FILE0001.CHK, and so on). It does this, rather than simply marking them as free, so that you can have a look at their contents to see if there's anything valuable in there. If there isn't, just DELETE the .CHK files and the space is free for future use.

So, lost clusters don't necessarily mean a bad disk or drive, just that some minor error has occurred at some point. It is, however, a good idea to run CHKDSK periodically as lost clusters do waste space on your hard disk. Bad sectors, however, are a different matter. If the number of these increases over a period of time, suspect that the disk is on its way out.

111. Backing Up Work Files

A simple way of keeping an up-to-date copy of the contents of your working directory on a floppy disk is the command:

```
xcopy *.* a:/m/v
```

This copies all files in the current directory which have their archive bit set, and then turns the archive bit of the source files off. Add /s to include sub-directories. The nice thing about this is that if your floppy becomes full during the backup you can simply insert another one and retype the command. It won't copy files that have already been backed up because these will have had their archive bit turned off.

112. Speeding Up a Hard Disk

A hard disk drive (or hard card) operates in essentially the same way as a floppy disk drive, only much faster and with much closer tolerances. The drive itself consists of at least two double-sided disk 'platters' mounted on a single spindle, with a separate read/write head over each surface linked so that the four heads are always accessing the same track

number. A 'cylinder' is the set of tracks simultaneously accessed by all the heads.

Each track of each disk is further divided into 'sectors', each holding 512 bytes of data plus special sector addressing information. A hard disk usually has 17 sectors on each track, or 26 if RLL coding has been used. RLL coding, standing for 'Record Length Limited', is the method usually used for packing more data onto the surfaces of a drive, allowing suitable 20 Mbyte drives to offer 30 Mbyte or more.

So a floppy disk has two sides of 40 tracks each containing nine sectors of 512 bytes, making a total capacity of 368,640 bytes, or 360K. A typical 20 Mbyte hard disk has 615 cylinders read by four heads, each containing 17 sectors of 512 bytes making a total of 21,411,840 bytes, or 20,910K. A 30 Mbyte disk with 26 sectors rather than 17 provides 32,747,520 bytes, or 31,980K.

A disk, whether floppy or hard, requires formatting at both a low level and a high level. The low level format actually defines the sector regions on the disk service, and writes the sector addressing information so the disk controller can find out what sector it is reading or writing to. The high level format builds the directory and FAT structure.

On a floppy disk, both levels are performed by the MS-DOS command FORMAT. But on a hard disk drive FORMAT only performs the high level format – other routines are required for the low level format.

When a low level format is performed, the sectors are rarely laid down in succession, i.e. with Sector Two straight after Sector One. This is because it takes a certain finite time for the controller to prepare itself for reading each sector, so by the time it is ready for Sector Two the rotation of the disk (at over 3300 rpm) would have taken the head past the sector and the controller would have to wait for a whole rotation – 17 sectors – before it could be read again.

Referring to the diagram, imagine that the controller, having read Sector One, is ready to read Sector Two by the time disk rotation placed the head at position A. An 'interleave' of 1:1 would mean that Sector Two would have long passed; but an interleave factor of three (in other words 3:1) would mean that Sector Two would be along any millisecond now. If however the controller reacted slower, and was not ready to read Sector Two until point B, then an interleave factor of four would be more suitable.

The 30 Mbyte Western Digital FileCard, for example, is usually supplied with an interleave factor of four (three for the 20 Mbyte model), however this turns in a poor performance when used with an Amstrad PC1512: the Norton SI Disk Index gives a reading of 0.5; while the Core Disk Performance Test Program returns a data transfer rate of just 25K/sec. The following procedure can greatly improve performance,

by setting the interleave of this system to 10. It can be used to good effect on many XT compatibles, as most use Western Digital disk controllers. You will need to experiment to find the optimum interleave factor though.

First of all though, remember that this procedure renders the existing contents completely unusable until they have been successfully restored, so your backup is your only copy of your valuable data! Remember too that *PC PLUS* takes no responsibility for the results, and cannot offer any help by telephone or by letter. You have been warned!

1. Make sure that any valuable information is properly backed up, as you are about to low level format the hard disk – perhaps you should make a second backup, just in case!
2. Remove the hard card from the machine and examine the case. There should be a label on the case listing bad tracks found during testing. Jot down their cylinder and head number.
3. With a PATH command set so that MS-DOS can find

DEBUG, enter **debug** at the prompt.

4. Enter **g=c800:5** at the hyphen that forms the DEBUG prompt. This executes Western Digital's own low level format program, stored in the disk controller's ROM. A message along the lines of 'Super Bios Formatter revision 2.4' will appear, followed by a series of questions.

5. Set interleave to 10.

6. Answer N so as not to dynamically configure the drive (used to specify individual areas for formatting).

7. Answer N to not virtually configure the drive either (for setting split points between separate partitions).

8. Proceed with the low level format, which should take five or ten minutes.

9. When it is finished you will be asked if you want to enter any tracks as bad. You do not have to do this as this particular controller tests sectors fairly thoroughly itself, but no harm is done if you do.

10. Now put a copy of your MS-DOS system and utilities disk in the floppy disk drive, reboot the computer, and enter **fdisk** at the A-prompt. If your disk does not contain the FDISK program, you should find it on the setup disk provided with the hardcard.

11. Press [Return] twice to select the default option 'Create DOS partition' from the initial menu and to use the entire disk surface for MS-DOS.

12. The system will reboot from your MS-DOS disk. Now enter **format c:/s** to perform the high level format, and turn your hard disk into a boot disk.

13. Reboot the system, this time from your hard disk.

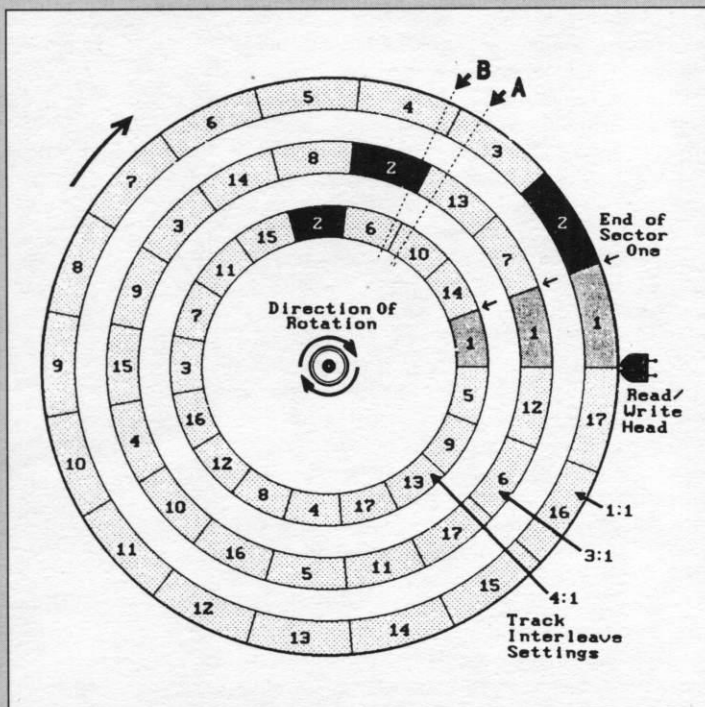
14. Restore your programs and data from the backup you made earlier.

After performing this operation, our system returned a more respectable Norton SI Disk Index of 1.1, while the Core test reported a considerably more nippy 67K/sec data transfer rate.

It is possible that performance could be improved a little more with an interleave of 9:1, while some drives may be best with 8:1 – you need to experiment. The same principles should work with other makes of drives too, but again this is an area for experimentation.

Exactly why the Amstrad PC1512 should require such high interleave factors is not known, but it could

78.



■ How the sectors are laid out for three different interleave settings – see text for details (re-printed from *SpinRite* promotional material)

well have something to do with the expansion bus being driven with a 4MHz clock, rather than the faster 4.77MHz used on the IBM PC. AT compatibles may similarly benefit from reducing interleave down to 3 or even 2. It takes considerable experimentation to determine the best interleave, and you will have to develop a suitable test of hard disk speed – perhaps a batch file that repeatedly copies and appends files.

Alternatively you could use *Spinrite*, written by Gibson Research and available in this country from Riva Limited on (0420) 22666. This tests the performance of your hard disk, finds the optimum interleave factor, and low-level formats the disk to this setting without damaging your data! It can also test the disk surface extremely thoroughly for potential faults, marking bad sectors where necessary and making available sectors that have been unnecessarily marked as bad. If you are running version 4 of MS-DOS, make sure the current version of *Spinrite* works under this version of the operating system before buying it.



113. Finding Lost Files

A useful command for finding files that have been lost in a complicated directory structure is:

```
chkdsk/v | find "filespec" | more
```

Where *filespec* is all or part of the lost file specification. The *chkdsk/v* outputs a list of all files with sub-directories, but make sure you've run *chkdsk/f* first to correct any lost clusters. The *FIND* command then searches this list for a match on the file specification, and the *MORE* filter pauses each screenful if there is more than 24 matches.

One disadvantage of this command is that you must remember to put the *filespec* in capital letters as *FIND* is case specific. For a little routine in *BASIC* which overcomes this problem, see the 'Programming' section of this book.

114. Checking the directory path

One problem which can occur when using the MS-DOS commands *BACKUP* and *RESTORE* is that *DIR*ectory listings of the backup floppies don't tell you which subdirectories the files were dumped from. This is particularly confusing when you happen to have files with the same name in several different directories – for example the two files *\LIVE\PAYROLL.BAS* and *\TESTING\PAYROLL.BAS* would actually appear as *PAYROLL.BAS* and *PAYROLL.@1* on the backup floppy disk's directory.

You can find out the full pathname of a file on a *BACKUP* floppy disk by using the MS-DOS utility *DEBUG*. Say you have a file *PAYROLL.BAS* on a

backup floppy in drive A, type:

```
debug a:payroll.bas
```

you should see a – prompt appear. Now enter *d* and a block of numbers and letters will appear. Somewhere in the top right hand part of this display the full directory and file pathname will be clearly visible. To leave *DEBUG* just enter *q*.

115. Problematic 32Mbyte Drives

Some hard disk units that use an RLL (Run Length Limited) controller to pack 32Mbytes of data onto the same size disk as an ordinary MFM (Modified Frequency Modulation) 20Mbyte drive, seem very prone to faults. Although an RLL controller provides a cheap way of increasing a disk's capacity, the higher density of the data makes the quality of the disk unit crucial. Disk manufacturers warn that RLL controllers must only be used with disks approved for RLL use, but unfortunately many dealers seem either ignorant or indifferent and are using an RLL controller with unsuitable drives.

Just as you can format a double density floppy disk to a high density format and get away with it for a while, so an RLL controller used with a non-RLL disk may appear to work satisfactorily for a few months. But after perhaps six months use errors start to occur, particularly on booting up, by which time your guarantee may have run out! It is worth remembering, in the event of problems, that you would still have a case against the dealer under the Sale of Goods Act, as the disk clearly was not suitable for its 'normal purpose' when sold – storing the advertised amount of data, in other words.

```
A>debug addle.exe
```

```
-d
1479:0100 FF 01 00 00 00 2F 53 55-50 45 52 44 53 4B 2F 50 ...../SUPERDSK/P
1479:0110 52 4F 47 52 41 4D 53 2F-41 44 44 4C 45 2E 45 58 ROGRAMS/ADDLE.EX
1479:0120 45 00 00 00 00 00 00-00 00 00 00 00 00 00 00 E.....
1479:0130 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
1479:0140 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
1479:0150 00 00 00 1D 00 00 00-00 00 00 00 00 00 00 00 .....
1479:0160 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
1479:0170 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
-d
1479:0180 4D 5A 68 01 4C 00 00 00-02 00 00 00 FF FF 00 00 MZh.L.....
1479:0190 80 00 00 00 AA 01 00 00-1C 00 00 00 00 00 00 00 .....
1479:01A0 54 55 52 42 4F 20 42 41-53 49 43 2F 43 6F 70 79 TURBO BASIC/Copy
1479:01B0 72 69 67 68 74 20 28 63-29 20 31 39 38 37 20 62 right (c) 1987 b
1479:01C0 79 20 42 6F 72 6C 61 6E-64 20 49 6E 74 65 72 6E y Borland Intern
1479:01D0 61 74 69 6F 6E 61 6C 2F-41 6C 6C 20 52 69 67 68 ational/All Righ
1479:01E0 74 73 20 52 65 73 65 72-76 65 64 00 00 01 43 00 ts Reserved...C.
1479:01F0 52 53 5A 61 6C 65 00 00-00 00 00 00 00 00 00 00 RSZale.....
-q
```

■ Do a directory of the backup disk and the file entry simply reads *ADDLE.EXE*. Use *DEBUG* and the full path of the original is discovered at the beginning of the file

To find out if you've got the wrong drive/controller combination, look for the model number, which should be prominently marked on the drive itself. Several manufacturers, including Seagate and Miniscribe, denote the drive's unformatted capacity by the last two digits of the serial number. Unformatted capacity is about 20 percent higher than the MS-DOS capacity of the drive, a 20Mb non-RLL drive may have a serial number ending with 25 whilst the 32Mb RLL version has a number ending in 38. Seagate put an 'R' on the end of model numbers suitable for RLL use.

This system however does not apply to some older Rodime drives nor to Western Digital drives, but we understand all Western Digital models are designed for RLL use. If you don't know whether you have an RLL controller, the chances are that a disk with a capacity of 30 or 32 Mbytes is, whereas one of about 20 Mbytes usually isn't. The most common RLL controllers are labelled the WD1002-27X and the Seagate ST11R. Check your number against the following:

Make	Not for RLL use	Suitable for RLL
Miniscribe	8425	8438
	3650	3675
Seagate	ST225	ST238R
	ST125	ST138R
	ST251	ST277R

116. Updating backup disks

Although one should of course keep one's hard disk fully backed up at all times, for many of us our day-to-day work only involves changes to a few files in a single sub-directory, so a full backup seems like overkill. It is easier to simply keep up-to-date copies of those files that have changed on a floppy disk.

Unfortunately it is not always that easy to remember which files you have changed through the day. The batch file below, UPDATE.BAT, uses the MS-DOS command REPLACE to automate the process.

Note that this technique relies on the internal clock being set correctly. Check with the DATE command.

117. Partitioning Hard Disks

Despite the fact that hard disks of 40Mbyte or more have been around for many years, only MS-DOS version 4.0 has allowed us to use the whole of such a disk in one go. Earlier versions have required various tricks that split the disk into 'partitions', each of which can be treated as a single disk with a capacity of 32Mbyte or less.

Formatting a hard disk involves three processes: the low-level format where the basic sectors and interleave information is placed on the disk; running FDISK to create the MS-DOS partitions; and a high-level format executed with the MS-DOS utility FORMAT. Under version 3.2 of MS-DOS, larger hard

Listing of UPDATE.BAT

```
ECHO OFF
for %%p in ("%1" "%2") do if %%p=="" goto wrong
for %%f in (a d) do replace %1*.* %2 /%%f/p
goto end
:wrong
ECHO.
ECHO Format for the command is
ECHO UPDATE source-disk: dest-disk:
ECHO as in UPDATE a: c:
ECHO Copies files to the back up disk that are either not
ECHO already present or are later versions of those already
ECHO there.Asks for confirmation before copying each file.
ECHO Current directories are used unless otherwise specified.
:end
```


disks have to be partitioned at the low-level format stage, effectively tricking MS-DOS into thinking the single unit is in fact two separate drives. However the FDISK program of version 3.3 allows you to define more than one MS-DOS partition on a disk. Each partition still can't be more than 32 Mbytes long, but the remainder of the disk can then be defined as if it were a separate disk.

This is not a process to be approached lightly as it can easily render your hard disk unusable. It is certainly not something to attempt without at least one and preferably two backups of your programs and data – preferably each carried out using different backup utilities. It is, however, made easier by the provision of routines built in to the Western Digital disk controller, used on most XT compatibles and can be used to considerably enhance disk performance by changing the interleave factor. These instructions can be adapted for an AT, but remember to set the Drive Type first (see elsewhere for details).

Before starting, it is necessary to find out how many cylinders your hard disk has, and how many heads. These details should be in the disk's manual – the following instructions take a 615 cylinder disk with four heads as an example, and apply if you are running version 3.2 of MS-DOS:

1. Put a copy of your MS-DOS utilities disk in drive A and enter **debug** at the A-prompt.

2. Enter **g=c800:5** at the debug hyphen prompt. This runs the special low-level format routine built in to the Western Digital controller. If your machine does not respond to this command it probably doesn't have a Western Digital Controller, and should come with its own low-level format routine which you should use instead.

3. Hit [return] to select C as the current drive.

4. Hit [return] to select the default interleave factor, or enter a new factor (see elsewhere in this chapter for details).

5. Enter [Y] to dynamically configure the drive.

6. The response will be a load of instructions, after which you should enter, in this case, **615 4**. This is the number of cylinders on your hard disk, and the number of heads – you can ignore the other entries which should default to their previous settings.

7. Enter **x** to virtually configure the drive.

8. You now need to specify the size of the first partition by entering the cylinder number where the division is to

occur in response to 'drive split'. Specify **307** if you want to divide it roughly in half.

9. Enter **x** to begin the low level format, which may take up to five minutes. If there is an error message try the process described below to format without two partitions. If there is still an error message there may be something wrong with your drive.

10. Enter **n** when asked whether you wish to format the bad tracks.

11. The computer will reboot from drive A. When it has finished, enter **fdisk** at the A-prompt.

12. Hit [return] when the main menu appears to 'create DOS partition'.

13. Press [Y] to use the entire disk for DOS.

14. Press any key and the system will re-boot.

15. Return to the FDISK main menu and select option 5 from the menu, to select next fixed drive.

16. Press [Y] to use entire disk for DOS.

17. Press any key and the system will re-boot.

18. Run FDISK again and select option 4 to check you have two partitions with the first set as Active.

19. Format drive C using **format c:/s**.

20. Format drive D with **format d:**.

If you simply wish to format the whole of the drive as a single unit, answer [N] rather than [Y] at step 5 and jump straight to step 9.

If you do not have a Western Digital drive this procedure cannot be carried out as detailed here. However you should have been supplied with a floppy disk containing, amongst other things, a low level format utility designed for use with that drive.

If you are using MS-DOS version 3.3, then you should again answer [N] rather than [Y] at step 5, and jump straight to step 9. At step 13, after loading FDISK and choosing to 'Create DOS Partition', you should select [N] so as not to use the whole disk for the 'Primary DOS Partition'. You will then be asked for the partition size, and should enter the number of cylinders you want for the first drive – 307 in our example here. You should then create a second 'Extended DOS Partition', running from cylinder 307 to 614 (the last cylinder on the disk). This second partition is called a 'logical drive' and should be assigned drive letter D.

Make sure that the Primary DOS Partition is the Active partition (rather than Non-Active) so that it can be used to boot up the machine. You can then proceed with the high-level format as from step 19.

118. AT Drive Types

It is important for AT owners to know the Drive Type number of their hard disk, particularly if they are going to attempt a reformat! To understand this number's importance it is necessary to understand a little of how a hard disk system works.

The whole system, whether in a PC or an AT, actually consists of two components: the hard disk itself and an expansion card containing the drive controller (on a hard card the two are combined on the expansion card). The hard disk bit can be thought of as anything from two to eight floppy disks stacked one above the other, with each surface served by a separate read/write head. Like floppy disks, each surface is divided into circular tracks which are in turn divided into a number of sectors, although it is more usual to call the tracks 'cylinders', and to refer to a particular track by its cylinder and head number.

Each 'platter', as the individual disks are called, stores considerably more data than a floppy disk, and rotates at a far higher speed. While the typical 1.2M floppy disk has 80 tracks on a side and 15 sectors per track; a hard disk platter pretty well always has 17 sectors per track but can have anything from just over 300 up to 1024 tracks per side. Hence the higher capacity of a hard disk drive, and the faster access time.

The hard disk unit itself is fairly dumb; it knows how to move its heads about and spin its platters, but not how to read or write data to the disk. This is taken care of by the disk controller which tells the drive where to move its heads and when to read or write data. The disk controller contains the electronics that translates a command to write data to disk into the actual actions required of the drive to carry this out. Most can look after a total of four drives as well: two hard drives and two floppy drives.

On the PC and XT the disk controller is specifically matched to the hard disk itself, so the controller for a 10M hard disk drive that uses four heads and 306 cylinders will only work with another hard disk of the same specification. Upgrade to a 20M unit and you need a new controller as well. However on the AT the controller is rather more flexible, which is where Drive Type numbers come in.

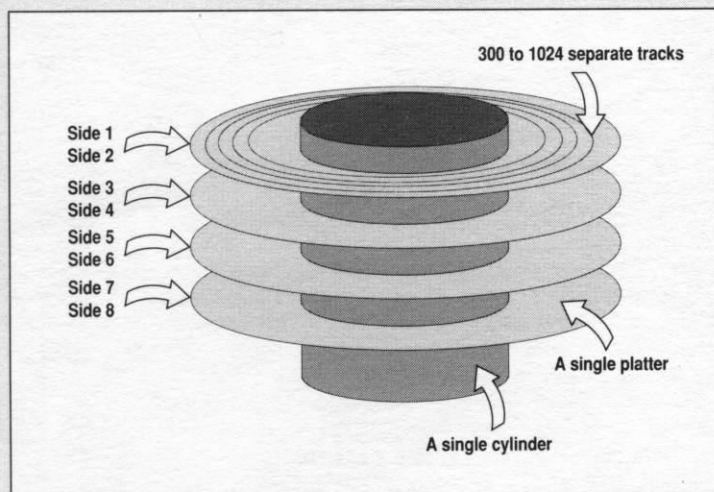
Unlike most PCs, the AT has a small area of memory where the configuration of the machine is held, powered by a set of batteries so that the information is not lost every time the machine is switched off. Here the type of display adapter is set, the capacity of the floppy drives, the amount of memory and the Drive Type of any hard disk drives installed. In this way a 20M hard disk can be changed for a 100M unit without having to buy a new disk controller; all you have to do is set the Drive Type appropriately.

But how to find out the type number? This is determined by the number of cylinders, the number of read/write heads, the 'land zone' and the 'pre-comp' number. Most of this information can be found either written on the drive casing itself, or in the drive's manual. The 'landing zone' is the cylinder where the drive heads can be safely 'parked', and is usually the last cylinder on the disk. 'Write precompensation' lists the cylinder at which the level of electrical current fed to the write head is increased to compensate for the smaller radius of the inner tracks.

In theory a simple table can be used to look up the Drive Type number once this data is known. Unfortunately disk and AT compatible manufacturers have been unable to agree on the assignment of these numbers above around 15, so it is a question of finding out what particular assignments your manufacturer has chosen and written into your machine's BIOS.

On quite a few machines the SETUP program, used to edit the information in the battery-backed RAM, is actually part of the ROM BIOS and can be summoned by pressing [Ctrl] [Alt] and [Esc] during the boot-up of the machine. This is often a relatively friendly program that gives you access to the computer's own Drive Type table so you can work out what number to use.

The number you choose determines how the



■ Both sides of each platter of the hard disk are formatted into tracks, with a cylinder referring to a set of tracks all at the same radius on each side of each platter

controller will format your disk, and should be set before even the low-level format is carried out. You do not have to use exactly the right number, as long as the number of cylinders and heads specified does not exceed those actually present. In general it is better to set for too few cylinders than too few heads: set one too few cylinder and you loose certainly less than 20 tracks of space; set one too few heads and you loose access to a whole side of a disk containing up to 1024 tracks. If you are uncertain which of two or three numbers to choose, it is safer to go for the lower number.

For example, the Miniscribe 6053 has 1024 cylinders and five heads, giving a total formatted capacity of 42.5 Mbyte. If you have an IBM AT you are out of luck, as this configuration does not figure in this machine's Drive Type table. Drive Type 11 offers 855 cylinders and five heads, giving a capacity of 35.4 Mbyte, but perhaps better is Drive Type 17 with 977 cylinders and five heads. This means that 47 cylinders will go un-formatted, loosing around 400K of space, but that is the best the IBM AT itself could offer.

If you're lucky your machine's BIOS will cater for this configuration. If not you can simply put up with this minor loss or use a hard disk setup utility such as *Disk Manager* from Ontrack or *SpeedStor* from Storage Dimensions. These override the BIOS table to cater for

a wider range of drive configurations and provide a number of other utilities. They may, however, interfere with the operation of programs that operate at a particularly low level and expect standard IBM procedures.

More curious readers might be wondering at this point what happened to the sixth side of the Miniscribe drive? Indeed quite a few of the Drive Types have an odd number of drive heads, and there must of course always be an even number of drive sides. These models use a linear motor to control the movement of the head assembly, rather than a stepper motor that moves in discrete steps across the disk surface. There is indeed a sixth head, but it is used to read special index marks that ensure the head assembly is correctly positioned to read each track, which means it is not available for reading or writing data. Now you know.

119. Useful Books

Hard Disk Management – A Quick Reference Guide by the ubiquitous Ray Duncan, is an excellent pocket-size guide to using a hard disk drive sensibly and is available for £4.95 from Microsoft Press who can be contacted on 01-938 2200.

T H E
**HELP
SCREEN**
C O L L E C T I O N



H A R D W A R E

120. 640K For The PC1512

Inside the PC1512 are eighteen empty IC sockets which can be filled with 128K more memory, taking the machine to the operating system limit of 640K. Your Amstrad dealer may be able to do the job for about £40, but you can do it for yourself for almost half that. The upgrade is extremely simple and requires no soldering, no special tools and no deep understanding of electronics. It should take less than one hour, but remember that, as with any other 'unauthorised' rummaging about in the machine's interior, the warranty is invalidated if you do this yourself.

The instructions that follow are in great detail so that anyone can do the upgrade whatever their knowledge of electronics. But before you start, some words of advice. Make sure you are in a patient, careful mood, and remember that none of the operations requires much force! Follow our instructions to the letter if you want this to work first time.

Note also that these instructions apply to older PC1512s. We understand that recently Amstrad changed the motherboard design which means that the sockets are in a different place and, more

important, different types of chips are required. If you bought you 1512 in the last year or so, proceed with caution!

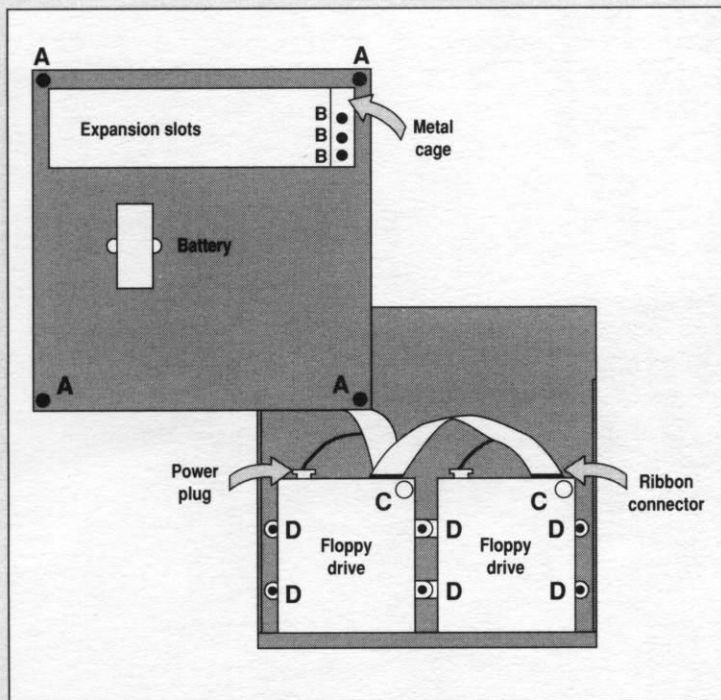
You need eighteen 4164-15 64Kbit Dynamic RAM ICs. The figure 15 in the code number refers to the access time of the chip, and implies 150 nanoseconds. You may also see the suffix 12, meaning 120ns. Either type is quite suitable, but chips slower than 150ns should not be used. The actual make is not important, but the code number is.

Chips are supplied with the legs slightly splayed, but in this form they will not fit in the sockets. The legs must be bent so that the two rows are parallel and at right-angles to the body of the chip (see diagram). This may be done by resting each row of legs in turn on a table and pushing the body of the chip into a vertical position. These integrated circuits are a very common item and are available from almost all electronics suppliers but at widely varying prices.

Before you start find a suitable working area. A table of at least three feet square will do nicely, covered with a white cotton sheet (not nylon). Make sure the working area is well illuminated, and have to hand a No. 2 Philips screwdriver. A small flat-blade screwdriver and a pair of good-quality long-nose pliers will make life easier, but are not essential.

THE INSTRUCTIONS

1. Switch off the Amstrad and remove the plug from the wall socket.
2. Disconnect the keyboard and mouse from the system unit, and set them safely to one side.
3. Remove the monitor power and video plugs from the back of the Amstrad, lift off the monitor and set it to one side.
4. Remove any connections from the serial and parallel ports, and place the system unit on your work area.
5. Remove the top and side expansion board covers by pushing in the direction of the arrows moulded into the plastic.
6. Lift off the plastic caps covering the front two screws marked **A**, and then remove all the screws marked **A** and **B**. The four screws **A** are all the same size, as are the three screws **B**. Slide the metal cage released by the screws **B** in the direction of arrow so that it clears the edge of the plastic casing.
7. Gently lift up the top cover – but do not try to lift it off completely because the memory back-up batteries are attached by wires to the main circuit board. It may be necessary to pull the front panel towards you slightly to release the front edge of the top cover.



■ Inside the PC1512: top left shows the system box after the monitor has been removed; bottom right shows the disk drives after the lid has been removed

8. Unclip the cable tie securing the battery back-up wires to one of the disk drives and then detach the battery wire plug from the main circuit board with a firm pull upwards. The top cover may now be lifted off and set aside.

If you have only one disk drive the sockets for the RAM chips will now be visible. It is possible to plug in the extra memory chips without further ado, but this is not recommended as the circuit board underneath the sockets is unsupported and the chips need quite a firm push. This is, however, the way the most dealers do it.

9. Noting which way up they are, disconnect the ribbon cable connectors from the rear of each disk drive: a small flat-blade screwdriver can be inserted in hole **C** to lever gently against the plug to assist its removal. Do not attempt to detach the ribbon cable from the plug – the cable and plug should come off complete.

Pull out the white plastic four-wire power plug from the back of each drive. These will only go into the socket one way round – but it is quite possible to put the ribbon cable back upside down so beware! You could put a sticky label on the top of the ribbon connectors to serve as a reminder.

10. Remove the six screws **D**, noting the position of any earthing connectors for reassembly. Lift out the drives, having marked them in some way so that they can be replaced in the same positions.

11. Remove the plug for the front panel power light and the loudspeaker connector from the main board. Both plugs are on the left of the circuit board and require a firm pull upwards. Pull off the volume control knob situated just forward of the keyboard connector on the outside left of the case.

12. Identify the nine Philips screws which secure the main board to the case and remove them. The main circuit board may now be lifted out front right-hand corner first. The lower section of the metal interference shielding stays in the case, but the upper section has to come out with the board – you can free it from the board by threading the connectors through the appropriate holes (earlier machines didn't have this shielding).

13. With the board flat on the work surface fit the chips into the sockets, which are labelled IC153 through to IC170. All the chips go the same way round, with the small semi-circular cut-out at one end of the chip facing towards the mouse and keyboard connectors. Take care inserting the chips, and inspect them carefully to make sure the legs have not folded underneath rather than entering the sockets. When you have finished, make a final check to ensure that all

chips are pointing the same way.

14. The last job before reassembly is to set the jumper LK4, which is situated just to the right and below the top right-hand RAM chip IC162. The jumper consists of three vertical pins, the left hand and middle pins being connected by a copper strip inside a black plastic link. The position as the machine is supplied is marked on the board as the 512K position. Pull off the link with fingers or a pair of pliers and replace it so that it links the middle and right hand pins, marked on the board as the 640K position.

15. Reassembly is simply a reversal of the instructions 12 back to 1, subject to the following notes:

- Refitting coarse threaded screws into plastic can be tricky. Turn the screws anticlockwise first, until a slight click is felt, then do them up. If the screw is stiff, try again.

- When refitting the circuit board screws, locate them all before tightening any up, and do not overtighten.

- The plugs for the front panel power light, loudspeaker and the battery back-up can only be fitted one way round. Inspection of the plug will show which way.

- Check that the disk drives are properly centred in the front panel apertures before tightening their screws.

16. Boot up the machine with your MS-DOS disk and observe the magic words 'Amstrad PC 640k' on the top line of the screen. The internal speaker might give you a surprise if you left the volume set too high as well. Type **Time** and then **Date** to reset the internal clock.

17. If the machine doesn't work properly, switch off immediately. Recheck everything and try again. If it still doesn't work then remove the RAM chips, preferably using a chip puller; replace the jumper in its 512K position and try once more. If it works now one of the RAM chips must be at fault, so get your supplier to replace them.

121. File Transfer

Transferring files from one type of computer to another is an unpleasant business. There is, in theory, a standard way of connecting computers, known as RS232, but unfortunately this is so flexible that it is almost no standard at all. Below are shown some of the wiring systems that *PC PLUS* readers have found satisfactory for wiring a PC's RS232 or serial port to

the equivalent on another machine.

Once you're connected up the next stage is to tell the machines to transfer a file across the link. You can do this either by purchasing suitable communications software for both ends, or you can use the operating systems to do it for you.

At first sight the PC seems ideal for this sort of thing, as the MODE command in MS-DOS allows you to set baud rate, parity, stop bits and data bits. MS-DOS does not provide any of the handshaking or error checking routines you would get with a proper communications package, but if you are determined not to spend more than you really have to you can

transfer files to and from a PC without them.

Handshaking is the process by which one computer tells the other when it is ready to send or receive data, and can either be controlled by the software or the hardware. Software handshaking, such as is provided by setting XON/XOFF, means that one computer sends a message in amongst the data that tells the other computer to 'hang on a microsecond'. Hardware handshaking, on the other hand, means that handshaking signals are sent down separate pins – pin 5 for 'Clear to Send Input' (CTR) and pin 20 for 'Data Terminal Ready Output' (DTR).

In MS-DOS you can use the MODE program as follows:

```
MODE COM1:baud,parity,data,stop
```

Where **baud** can be 110, 150, 300, 600, 1200, 2400, 4800 or 9600,

parity is N (none), O (odd) or E (even),

databits is either 7 or 8, and

stopbits is either 1 or 2.

So to set 1200 baud, no parity, eight data bits and one stop bit you would enter:

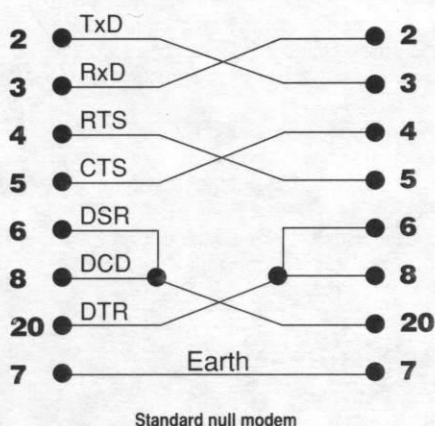
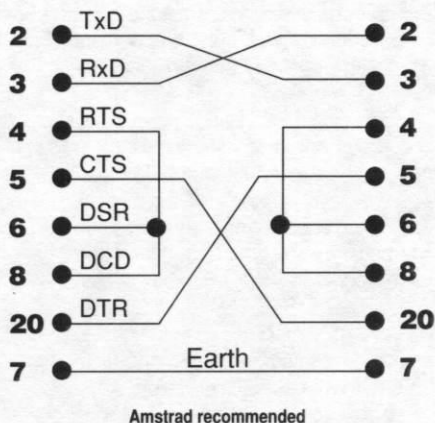
```
MODE COM1:1200,N,8,1
```

You can use the MS-DOS command COPY to receive a file on the PC, with `COPY AUX: filename/A`, where **filename** is the name of the new file you wish to create. Use the /A if you are receiving an ASCII file. It adds a [Ctrl][Z] to the end of the file, which is necessary to close the file and save it to disk. If you are transmitting a file from the PC then the instructions are `COPY filename AUX:`.

There are a number of very reasonable communications packages available for the PC as shareware or even in the public domain, which support proper handshaking and error checking protocols, in particular either the XMODEM or Kermit systems which are designed specifically for transferring files. Popular packages include *Procomm*, *Kermit* and *Qmodem*, but remember that you will need suitable software for the other machine as well.

If you are using an Amstrad PC1512, you can also use the PIP command that comes with the DOS PLUS operating system. This is similar to the MS-DOS COPY command, but rather more powerful. To import a file to a disk in drive A, enter `PIP A:IMPORT.FIL=AUX:`. If in drive B, enter `PIP B:IMPORT.FIL=AUX:`.

Now send the file from your other computer. At the end of the process the drive should run and the prompt reappear. If you require some reassurance that things are transferring properly you can terminate the PIP command with [E] so everything will be echoed to the screen, as in `PIP A:IMPORT.FIL=AUX:[E]` To send a



■ Two possible ways of connecting two RS232 ports together. If the Standard Null Modem wiring doesn't work, try Amstrad's recommendation for the PC1512.

file from the PC1512 to another machine, enter **PIP AUX:=A:EXPORT.FIL**

If your source data is not in ASCII form and there is no convenient way of making it so, you may still be able to send data by redefining the printer channel and telling your application to print the data. Re-define the printer channel with **DEVICE**, entering:

```
AUXOUT:=NULL
PRN:=COM1
```

Now set up the destination computer to receive data and run your application. Print the relevant data, and it will be passed to the other machine as if it were a printer. If the destination machine hangs up at the end of a transfer from the PC1512, enter **PIP AUX:=EOF**:

AMSTRAD PCW

File transfer from the Amstrad PCW8256 or 8512 word processor is fairly straightforward thanks to the PCW's implementation of the CP/M operating system. Using the SETSIO program (on side 3 of the master disks) you can set not only baud rate, parity, stop bits and the like; but also handshaking. This program supports both XON/XOFF software handshaking and RTS/CTS hardware handshaking.

Files can then be transferred using the PIP program supplied. To receive a file on the PCW use **PIP filename=AUX:[E]**, where **filename** is the new file you wish to create. Use the **[E]** option with text files as it echoes the file to screen as it comes through. So to send a file from the PCW the command to use is **PIP AUX:=filename[E],EOF**. The **EOF** ensures there is an end-of-file marker at the end of the file.

You do however need to purchase a serial interface, as this does not come with the PCW. Our sister mag *8000 PLUS* has tested several, many of which come with file transfer utilities. As for the cable, success has been reported with both the standard null modem wiring and the 'self-shaking' cable shown here.

There are a few communications packages available for the PCW, and there are several Public Domain packages available for CP/M machines, most of which work on the PCW.

AMSTRAD CPC MODELS

Like the PCW, the 6128 comes with the CP/M PLUS operating system so much the same applies. Pace makes a particularly good RS232 interface that is suitable for the whole CPC range and comes with excellent communication software in ROM, supporting a range of protocols including XMODEM.

An alternative is Amstrad's own Serial Interface at £50 which comes with an instructive if eccentric manual. It includes a ROM that adds the necessary commands to Amstrad BASIC, or you can use CP/M.

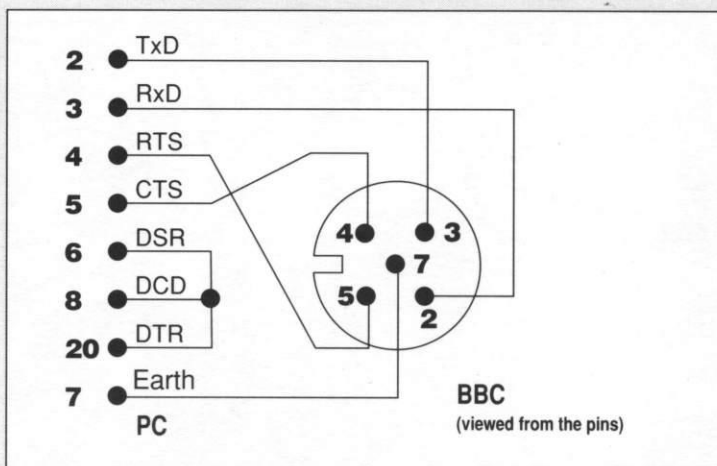
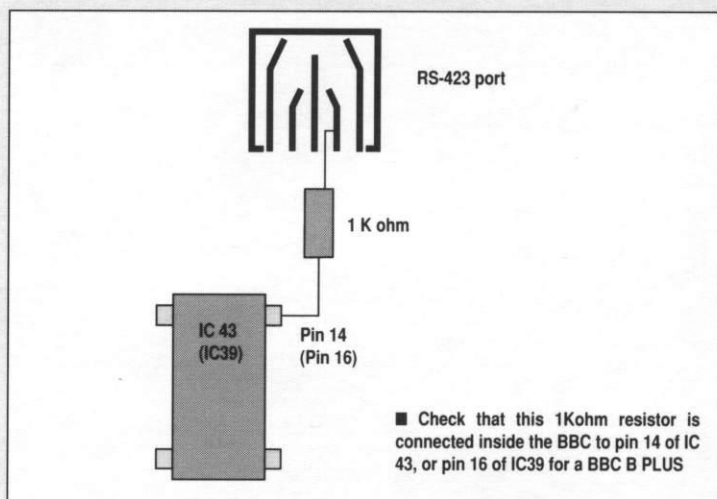
ACORN BBC B

The BBC does have a serial interface, though it is a little unusual. It is officially an RS423 interface but is compatible with the RS232 standard, and consists of a female five-pin DIN socket. One problem with the socket is that the plug can be inserted in either of two ways – so make sure you put it in the right way round!

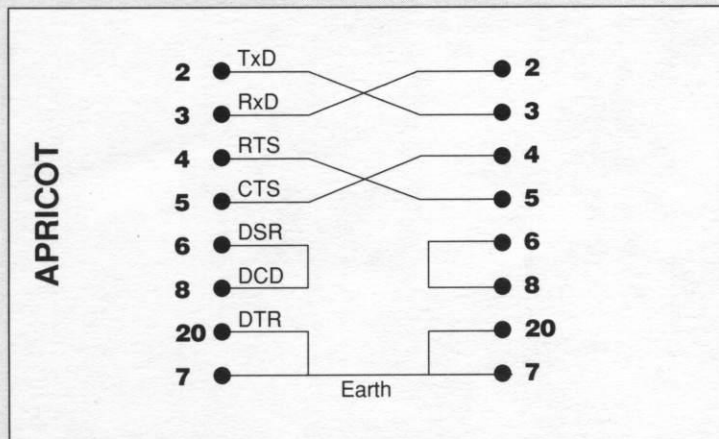
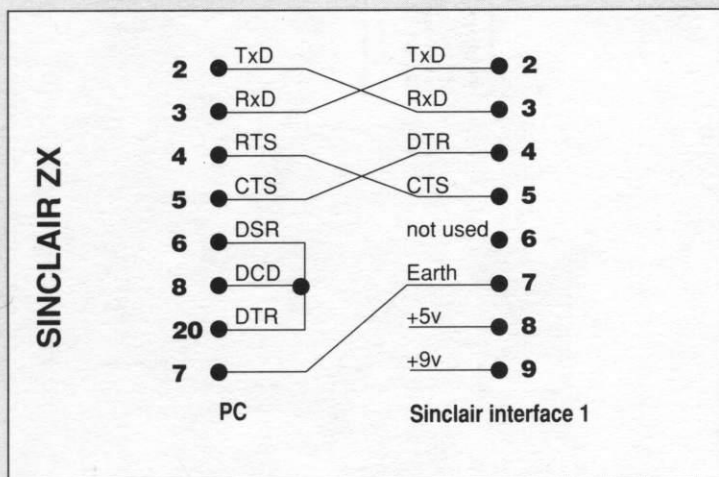
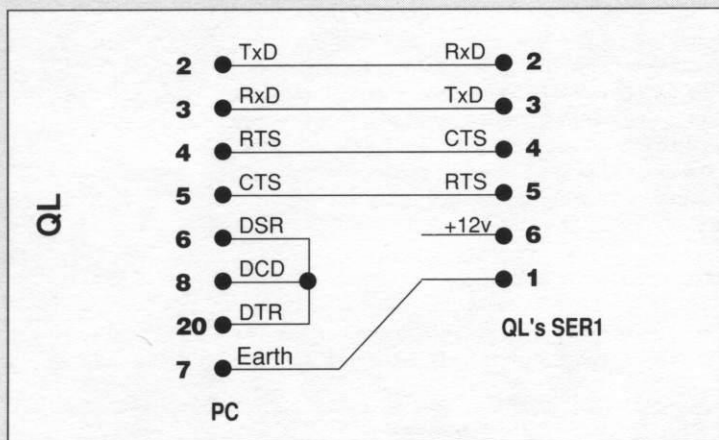
The cable should be wired as shown. The optimum is 2400 baud, which can be set using the MS-DOS **MODE** command as in:

```
MODE COM1:2400,N,8,1
```

To transfer from BBC to PC, first enter the line **COPY AUX filename** on the PC, which will copy the



■ If you have problems with this wiring system, try disconnecting pins 4 and 5 on the BBC end and connecting them together at the PC end



■ Suggested wiring for connecting the PC's RS232 port to three very different machines. Note that some of the QL and Spectrum's pins carry power, which could damage the PC port

data into the file named. On the BBC, enter the following program:

```
10 REM Data to PC
20 *FX6,0
30 *FX7,5
40 *FX8,5
50 *FX5,2
60 VDU2
```

RUN

Anything which appears on the BBC screen will automatically be transferred to the PC, although you may not be able to see anything at this stage (don't press [Break] whatever you do!). To end transmission, enter this on the BBC:

```
VDU1,26
VDU3
```

To go the other way, enter the following program:

```
10 REM Data to BBC
20 *FX6,0
30 *FX7,5
40 *FX8,5
50 *FX5,2
60 *FX2,1
70 A$=""
80 *SPOOL NEWFILE
90 REPEAT
100 PRINT A$;
110 A$=INKEY$(100)
120 UNTIL ASC(A$)=26
130 *SPOOL
140 *FX2,0
150 END
```

Run this program on the BBC, and you can now send files from the PC with **TYPE filename >AUX** or alternatively **COPY filename AUX**. Try **DIR >AUX** to display the PC's directory on the BBC! The ***FX 2,1** tells the computer to treat the RS423 port as if it were the keyboard, so you could send command sequences from a file on the PC1512 if you wished.

If you don't have any luck, check that a one Kohm resistor is connected as per the diagram. It is essential that this modification is carried out for the system to work properly.

If you would rather use professional software, the ROM-based *CommStar* software is almost universally recommended for the BBC. It supports XON/XOFF and XMODEM.

SINCLAIR QL

This eccentric machine does sport RS232 interfaces, but uses British Telecom phone jack connectors wired in a most unusual fashion. The QL's operating system is outside the scope of this book, but suffice it to say that SER1 is wired as a modem while SER2 is connected as a terminal. The diagram shows a modem cable suitable for serial interface SER1. Do not connect anything to pin 6 on the QL though as it carries 12 volts DC.

Success has been reported using a speed of 1200 baud, but we will play safe with 600 baud, so boot up the PC1512 under DOS PLUS and again enter **DEVICE**, followed by **AUX:=COM1** and:

```
COM1 (SPEED=(600,600),PARITY=NONE,DATABITS=8,STOPBITS=2)
```

Again, use the MS-DOS command **MODE** if you do not have DOS PLUS. The QL transmits eight data bits with no parity and two stop bits by default, so all you have to enter is **baud 600**.

To test the setup type **PIP CON:=AUX:** on the PC1512.

On the QL enter:

```
copy_n con to serlhc
```

Type what you like on the QL and you should see your text appear on the PC1512. When you wish to stop, press [ENTER] and then [Ctrl][space]. Both computers should return to normal.

To test the reverse connection enter on the QL:

```
copy_n serlhc to con
```

And on the PC1512 enter **PIP AUX:=CON:.** Anything you type on the PC will now appear on the QL screen. To finish press [Return] followed by [Ctrl]Z then **PIP AUX:=EOF:.** Everything should return to normal.

To send a file from the QL, put the microdrive cartridge containing your source file into drive £1 and set up the destination machine to receive. Then, assuming your file is called **outfile_txt**, enter:

```
copy_n mdv1_outfile_txt to serlhc
```

It is particularly easy to generate ASCII files on the QL by simply printing, or use the export function. If you do have to pretend that the destination is a printer, you might have to send a ^Z to close the file. Should this be necessary, quit the application and enter:

```
open #8,serlhc
close #8
```

and the character will be sent automatically.

To send a file **in_txt** to the QL enter

```
copy_n serlhc to mdv1_in_txt
```

SINCLAIR SPECTRUM

Although principally a games machine, with added microdrives the Spectrum is quite widely used for business. Add Interface 1 and you get a serial port too. The diagram shows a possible wiring for the 48K version, although we have not tried it ourselves. We understand the Spectrum Plus 2 is wired in the same way as the QL's SER1 interface covered above.

APRICOT F, PC and Xi

The old Apricot PC, XI and F ranges have rather unusual serial interfaces, which means that you have to earth the DTR pin for it to successfully send data through its RS232 port to a PC. Pin 7 is signal ground, so the simplest way is to connect pin 20, the DTR pin, to pin 7 (see diagram opposite).

Most systems assume that the other computer is ready to receive whenever DTR (Data Terminal Ready) is 'on' or 'high'. These Apricot machines require it to be set 'off' or 'low', so will not send data through a cable wired up in the normal way.

These machines also came bundled with the quite adequate *Async* communications package. Note that they have 25-way D-type sockets, but female instead of the male socket found on most PCs. (Computer plugs and sockets, like most members of their clan, either have pins coming from them or holes to accept the pins of others. The ones with pins have become known as 'male', while the ones with holes are 'female'. Ain't that cute?)

POSSIBLE PROBLEMS

If there is data in the buffers then funny, nil length files might be created. This usually occurs only if a previous transfer has failed in some way. To clear any residues, direct the input to the screen and then close down the transfer correctly.

Sometimes you may find that data is passed with errors or characters added, missing or corrupted. Try re-transmitting at a lower baud rate. Ensure that your leads pass well clear of sources of interference such as transformers and monitors. Ensure that all plugs are pushed well home and all solder joints are good and firm. Should you get nothing but total rubbish, then check that all your **DEVICE** parameters are correct and that the baud rates are the same.

Set up the destination machine first and then send the file, otherwise the first part of the file might be lost before you can get the receiving computer ready. Use five-core screened cable; If this is unavailable then five-way ribbon cable, maximum length two metres, will do.

Once a file has been transferred it may need some work before it can be used by your PC applications.

Most serious packages accept input from an ASCII file so there should be no difficulty importing to *dBase* or spreadsheet. You may have a little difficulty in some cases but almost all problems can be sorted out with a text editor or a word processor.

The things to look out for are control codes and end of line characters. Usually these only occur if you have sent the data to a pseudo printer, in which case printer control sequences might be included. By far the commonest problems are caused by too many or too few carriage returns and newline characters ([Ctrl]M and [Ctrl]J respectively). Inspect any problematic file with a word processor and use your common sense. Normally a couple of global search and replace commands will do the trick.

ALTERNATIVE METHODS

An alternative to transferring data from one machine to another is to try and persuade the PC to read other disk formats and sizes. By itself the PC will read disks from any other PC compatible, but will not read disks – even 5.25-inch disks – from CP/M machines or micros such as the BBC, Apple II, Commodore 64 or the Sirius. This is because these machines store data on the disk in a totally different format from the PC.

It is however possible to add an extra disk drive to your PC that can read some these formats, or indeed to add a 3 or 3.5-inch drive to read disks from Amstrad CPC or PCW machines, Apricots and the Atari ST. This is of course an expensive option, but could be worthwhile if you regularly need to read disks from other machines. Check advertisements in magazines such as *PC PLUS* for details.

If you don't want to go to all the trouble of buying the hardware and software needed to transfer files yourself, you can, of course, pay other people to do it for you. There are several firms who will transfer data between a wide variety of disc sizes and formats, making a reasonable charge for doing so. As experts in the field, they often have at their fingertips knowledge that it would take you many days and pounds to discover.

122. Mouse Compatibility

Amstrad's marketing literature states that the PC1512 and PC1640 includes a 'Microsoft-compatible MOUSE.COM'. Unfortunately the Amstrad mouse will not drive the Microsoft *Windows* pointer. However, if you install third party software (neither Amstrad nor Microsoft) and tell the installation program that you have a Microsoft mouse, the program will usually use the Amstrad mouse without problem.

Microsoft publish specifications for device drivers for software to run with the Microsoft mouse, but it looks as though Microsoft breaks its own rules by not

using its own device drivers in *Windows*. Instead, this package checks at a hardware level, and looks for particular interrupts. Although there are a number of mice other than Microsoft's that are OK with *Windows*, the Amstrad mouse isn't one of them.

123. Joystick Problems

The Amstrad PC1512 and 1640 joystick – the one that plugs into the back of the keyboard – is not compatible with the standard IBM joystick. There is a way round this though, which works with many PC games and relies on the fact that moving the Amstrad joystick control is equivalent to pressing the cursor keys on the keyboard.

Instead of selecting for 'joystick' at the opening screen of the game, select 'keyboard'. If you are lucky you will be given the option of defining your own keys, in which case you simply move the joystick in the appropriate direction rather than select keys from the keyboard.

124. PC Joysticks

The Amstrad joystick port, which is unique to the Amstrad range of PC's, works in a completely different way from the standard IBM games adaptor. The Amstrad port expects a switched joystick – where each direction is either 'on' or 'off' – while the games adaptor is designed for an analogue device. But this is just where the differences start:

AMSTRAD JOYSTICKS

The Amstrad joystick port is an integral part of the keyboard firmware. Whenever a key is pressed the keyboard outputs a unique identifying 'keycode'. The Resident Operating System (or ROM BIOS) reads this keycode and, using an internal translation table, translates it into a 16-bit 'key token' which is placed in the Key Token Buffer.

The keycodes generated by the Amstrad joystick are the same as the cursor keys only after the ROM BIOS firmware has got hold of them and translated them. In fact the low level keycodes generated by the stick are in the range hex 77 to 7C, while the 'real' cursor keys are in the range 47 to 52. Any program which gets right down to the hardware, ignoring the operating system, will not map the joystick on to the cursor keys, and games are notorious for getting right down to it.

IBM GAMES ADAPTOR

The standard PC-compatible joystick, which most PC games accept, plugs into a games adaptor card which in turn plugs into one of the PC's expansion slots. A single bit of input/output port 201 (hex) is all that is

needed to store the analogue position of the stick in either the horizontal or vertical direction, so that the four low-order bits contain all the information needed to describe the state of the two joysticks catered for by this design.

How can this be? When a 'read' byte is output to this port these four bits are set to zero, each returning to one after a time delay that is proportional to the position of the stick at the time. Software can use the internal clock of the PC to measure the time delay, so generating a value that can be used by the program.

A neat idea but, as you can see, totally incompatible with the Amstrad joystick port. The four high bits of the port are used to read the state of the two buttons on each joystick, each going to zero if the corresponding button is pressed.

125. Sticking disks

Although the 5.25-inch floppy disk is surprisingly reliable considering its construction, the mylar disk itself (the coated plastic disk that actually holds the data) can get stuck inside its cardboard sleeve if the sleeve is bent or folded. The PC PLUS *SuperDisk* is particularly prone to this sort of problem if the magazine distributors bond the magazines too tightly, or it gets folded in the post.

The problem usually manifests with a 'General failure error reading drive A' when a simple MS-DOS command such as DIR is executed, and a general stiffness if you (gently) try and turn the disk within the sleeve. It can however be fixed by very carefully cutting open the side of the sleeve with a razor, making sure the razor does not contact the disk itself, extracting the mylar and slipping it into an undamaged sleeve that has been opened in the same way.

An alternative home is to be found with some disk drive cleaning kits. Some makes use an empty disk sleeve or 'glove' to hold the cleaning disk which makes a perfect temporary home for your newly-recovered mylar.

If this does not work immediately, try selecting 'retry' each time the error message is generated. As soon as the disk is successfully read, copy its contents to a fresh disk and there you are!

126. Amstrad PC1512 bug

The NVR program has a bug in it – if you set the 'extended flow control' to ON, the ramdrive can change from drive C to drive E. This causes problems with some software, like GEM, which on the PC1512 relies on the RAM disk being drive C. Fortunately setting the flow control back to OFF resets the RAM disk to C.

127. Alternative keystrokes

Did you know that, on an Amstrad PC1512 at least, instead of pressing [Alt][Ctrl][Del] for a system reset, you can in fact press [Alt][Ctrl][L-Shift][CapsLock]? or [Alt][Ctrl][L-shift][CapsLock]? or [Tab] [Alt] [Ctrl] [L-hift]?

To turn CAPS LOCK on and off, try [Tab][Alt][Ctrl], or [Ctrl][L-Shift][Del].

From MS-DOS, [Ctrl][Return][Del] scrolls the cursor down one line.

Also, [Ctrl][Ins][CapsLock] and [Ctrl][L-Shift][Ins][Del] produce ^A.

(These invaluable nuggets were provided by one Daniel Francis – aged 15 if not 16 by now – who went on to send in several Elite hacks and a picture of himself printed out on his new printer, before sending us a serious tip and winning himself a £30 voucher)

128. Non-booting Hard disks

Occasionally people who have fitted hard cards to an Amstrad PC have found that their machine refuses to boot of the hard disk and has to be booted of a floppy disk before the hard disk can be used. Apparently this is all to do with the colour monitor. The power supply for the Amstrad PC is in the back-left corner of the monitor, and on some machines it apparently produces quite a magnetic field – actually strong enough to affect the heads of a hard disk card sitting just below.

The solution is simply to swivel the monitor forward so there is a gap of about one inch between it and the case – you could even glue a rubber doorstep to the case to keep them apart. This may be all you need to do, but in some cases the boot tracks of the hard disk may have been corrupted as well, so you have to reformat it. Don't forget to backup up the contents before you start!

If all is well you should be able to turn your machine on, sit back and wait for the C-prompt to appear. In the interim your PC will access whatever floppy disk drives you have, but this is simply to determine what drives are installed, and whether there is a system disk in drive A. If there is then the PC will boot off that in preference to drive C; if there isn't then it should boot off drive C.

129. Changing Screen Colours

Owners of CGA machines frequently tire of games appearing in only cyan, purple, black and white; particularly as there are other palettes available. It should be possible for the adventurous to change this simply by swapping the wires that carry the colour

signals in the lead between adaptor and monitor.

On an Amstrad PC1512 the video connector is an eight-pin DIN socket, with pin 4 carrying BLACK, pin 5 carrying GREEN, pin 6 carrying BLUE and pin 8 carrying RED. Pins 3 and 8 are earth, pin 2 carries the Intensity signal while pin 1 carries the Composite H & V Sync.

We on *PC PLUS* haven't tried it and are not going to take responsibility for anyone who does – we suspect the whole new set of colours this is supposed to make possible are more likely to be seen flashing through the ventilation slots than on the screen!

130. Schneider Mouse

Anyone wanting to use a mouse with a Schneider Euro PC will be pleased to know that its nine-pin mouse socket will take the mouse from an Atari ST without problems. Simply plug it in, use the MOUSE.SYS driver provided with the Euro PC, and you've got a Microsoft compatible bus mouse. The Atari mouse can be picked up separately for less than a similar PC mouse.

131. Problems with Coloured Disks

Apparently some machines have problems reading floppy disks with coloured sleeves, as the reflectivity or transparency of the sleeve upsets one of the photosensors in the drive. The solution is to cover the offending disk with matt black paper which renders it instantly readable.

With a little experimentation you can work out the position of the sensor, and then cover the offending area with a black write-protect tab. On a Commodore PC40 covering the front lower right hand corner of the disk solved the problem.

132. Adding a Disk Drive

There are three disk sizes now in common usage, 5.25-inch, 3.5-inch and 3-inch. Most IBM PC-compatibles use 5.25-inch drives with a formatted capacity of 360K, however many are using the more robust (and more expensive) 3.5-inch disk with a formatted capacity of 720K. Most AT type computers are fitted with 1.2 MByte 5.25-inch drives, which are capable of reading and writing to a 360K disk; or 3.5-inch disks with a capacity of 1.44 MByte, capable of reading and writing to a 720K disk. The 3-inch disk drive is found on the Amstrad CPC and PCW range, where it has a capacity of 180K (or 720K for the high-capacity second drive on a PCW).

Fitting extra disk drives to a PC should be quite a simple job, as most PC's come with a floppy disk drive controller capable of handling more than one drive and

a variety of drive types. Most PC clones have a data cable that enables two floppy disk drives to be physically connected, whether or not they are actually supplied with the machine. The main problem is the actual location of the second floppy within in the casing of the processor box, although some are designed to sit outside the case on their own 'flying lead'.

With the Amstrad PC1512 or 1640 a single floppy disk machine should not be a problem, provided it is not already fitted with an internal hard disk drive. A hard card poses no problem as it contains its own disk controller and does not occupy the front panel. Most PC floppy disk controller can cater for both 360K and 720K capacity 5.25-inch drives, and 720K 3.5-inch drives. High capacity 1.2 Mbyte floppies would require another controller card, which are available but are not cheap.

The following instructions are specifically for the Amstrad PC, although a little common sense could adapt them for most other PC compatibles. It is fairly easy to fit another drive to your PC, either internal or external, and by no means calls for specialist skills. However, do bear in mind that performing this operation yourself would invalidate your guarantee, and that we cannot take responsibility for any problems that might arise. If you're not confident, get a qualified engineer to do it for you.

Most of these instructions are equally applicable to external drives, supplied with their own box and designed to sit next to your PC. The problem is linking the ribbon cable to that from the PC's disk controller, but the manufacturer should supply a suitable link that allows you to feed the cable through gap left when the side expansion slot cover is removed. External drives usually come with their own power supply too, so no problem there either.

Before you start, it is a good idea to check the setting of the 'drive select' switch on your new drive. When you open up your machine you will find that each floppy drive unit is connected to the same ribbon cable. In fact you could plug either drive into either of the two connectors coming off the cable, length permitting.

Each drive has a small switch and it's the setting of this switch that determines if it's going to be drive A or drive B (although you may find the switch marked drive '0' and drive '1'). Part of the information coming down the ribbon cable from the drive controller is a 'drive select' signal. Both drives receive the same signal, but only the one whose switch setting matches the drive select signal is supposed to respond. This, incidentally, is why a PC can only access one floppy disk at a time. If you were to have both floppy disks set as drive A, typing `dir a:` would cause both drives to start up and attempt to transfer data, with bizarre results!

1. SWITCH THE COMPUTER OFF AND REMOVE THE MAINS PLUG FROM IT'S SOCKET.

2. Unplug the monitor from the processor unit and take the monitor off the processor box.

3. Slide off the expansion port top and side covers.

4. Remove any cards you might have slotted into the expansion slots.

5. Undo the three expansion port screws and remove the four screws on the top of the processor box. Two of these are hidden under little round covers at the front of the box.

6. Ease the little metal cage by the expansion slot screws out, and gently remove the top cover of the system box. Lift from the back of the box and 'wiggle' it free.

7. Carefully unplug the battery lead from the mother board.

8. Remove the front panel and remove the mains indicator lead from the motherboard.

9. There are six or seven plastic lugs holding the left hand blanking cover on to the front panel. Use a pair of wire snips or a stanley knife to cut these lugs, and the blanking plate will drop out. Trim any surplus plastic from the lugs.

10. Fit the new disk drive to the mounting plate. At this point it is a good idea to check the 'drive select' switch has been correctly set (see panel elsewhere).

11. Undo and remove the two screws in the middle of the processor box holding Drive A.

12. Fit the new disk drive in the space on the left of the box. The plate sits on the four plastic pillars, the height of which is slightly different. Use two spacers, such as rubber grommets, under the mounting plate on the two centre pillars.

13. Replace the two middle disk drive screws but do not tighten them.

14. Fit the two new screws on the left hand side of the new drive. These screws are self tapping and can be very difficult to screw in – a smear of detergent or oil will help.

15. Refit the front panel, and tighten the middle screws sufficiently to line the new disk drive up with the front panel cut-out.

16. Replace the mains indicator lead on the motherboard.

17. Fit the ribbon cable to the new disk drive – looking from the front, the red stripe goes to the RIGHT. Fit the power cable, which will only go in one way.

18. Replace the battery lead on the motherboard, and

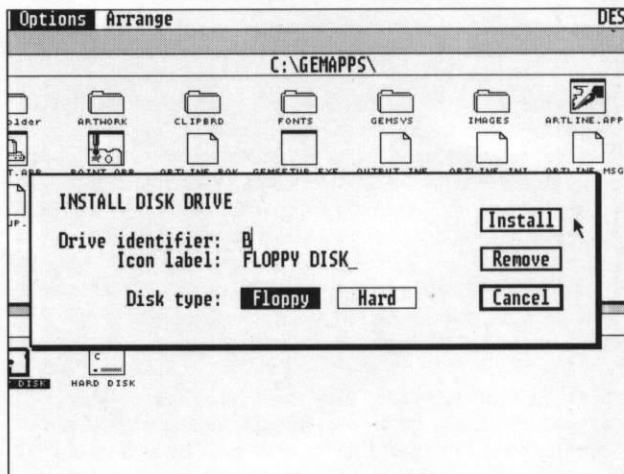
133. Configuring GEM for a New Disk Drive

The MS-DOS operating system will automatically recognise the new drive, and does not need to be told that this modification has been done. GEM, however, is not so intelligent, so you will have to 'install' the drive.

To do this, first get to the usual GEM Desktop display by entering gem at the with the appropriate disk in the old drive. Now make sure that one of the two windows is titled 'Disk Drives' and showing the disk icons – if not, click on the butterfly symbol at the top left-hand corner of the window until it does.

Click once on the floppy disk icon labelled 'A', so that it is highlighted. Now go up to the 'Options' menu and click on 'Install disk drive...', so a new window opens up entitled INSTALL DISK DRIVE. Hit [←Del] to delete the Drive Identifier 'A', and type B in its place. Now click on 'Install', and a new drive icon will appear in the window.

You now have to save this configuration if you want both drive icons to come up every time. To do this, go up to the 'Options' menu again, and select 'Save desktop'.



refit the top cover.

19. Replace the four corner screws.

20. Replace any expansion cards that were fitted and replace the expansion port screws.

21. Put the monitor back on the processor box and replace the monitor cables.

22. Power up the computer and boot up in your usual way.

If you've fitted a 40 track second drive (one with 360K capacity) then you do not need to change anything. If you've fitted an 80 track drive, whether 3.5-inch or 5.25-inch, you need to make a special formatting disk:

1. Format a blank disk in Drive A as a system disk, with the line **FORMAT A:/s**

2. Copy the file **FORMAT.EXE** on to the disk.

3. Add the line **drivparm=/d:1/f:2** to your **CONFIG.SYS** file. This command overrides the

controller's default characteristics. The **/d:1** designates drive number one, or the B drive. Drive A is the zero drive, and so on. The parameter **/f:2** designates the device type as a 720K floppy disk – type zero is the standard 360K drive.

When you want to format disks to 720K (80 track double sided), boot the computer with the disk you have just made, format your disk(s) and carry on! The computer will automatically recognise the new format whenever you use drive B in 80 track mode.

134. For the Disabled

A company that specialise in computer hardware and software tailored for particular disabilities – keyboards with locking control keys, for example – is Complete Computer Systems on (0895) 71490. For the visually impaired there is the VIPCUG user group on (0533) 403147.

135. No Screen Display

If your PC appears to be operating normally (disk drives whirr, you get bleeps in appropriate places etc.) but the screen remains blank, first check that the

96.

Key to Table of PC and PS/2 Vdeo Display Modes

Mode number: These are actually used to set the mode from the IBM BIOS, but provides a useful reference for those modes designed by IBM.

Resolution: The width and depth of the screen, in pixels for graphic modes or characters for text modes. Graphics modes allow the addressing of individual pixels on the screen, while text modes map predefined patterns into character 'cells'.

Colours: The maximum number that can be on screen at any one time.

Palette: The range from which the screen colours can be chosen.

Adapter: The basic PC design does not have graphics hardware built in – instead expansion cards (adapters) can be added to provide the modes required. Most of the adapters available conform to IBM designs, with the exception of the Hercules and that in the Amstrad PC1512. Many clones these days, including the Amstrad range, have the display adapter built into the main circuit board and not on a separate card.

MDA (Monochrome Display Adapter) Introduced by IBM in 1981, this adapter produces good quality text on a monochrome monitor, but without any graphics.

CGA (Colour Graphics Adapter) Also introduced by IBM in 1981, this produces text of a lower quality than the MDA, but also provides four-colour graphics in two palettes. It supports either RGB or composite video displays.

HGC (Hercules Graphics Card) Introduced in 1982 by Hercules, this emulates MDA but also has a graphic mode, so being the first adapter to allow graphics and text on the same monitor.

Can be configured either in **FULL** mode with 64K video RAM, or **HALF** mode with only 32K which allows it to co-exist with a CGA.

1512 Uses 64K of video RAM to add an extra non-standard mode to standard CGA. Only found on the Amstrad PC1512 models launched in 1986.

EGA (Enhanced Graphics Adapter) Introduced by IBM in 1985 to provide decent colour graphics and readable text from one adapter – has rapidly become the standard. In addition to the modes outlined in the table, most allow special 43-line or 132 column text modes.

MCGA (Multi-Color Graphics Array) Introduced by IBM on the PS/2 Models 25 and 30 launched in 1987, this adapter generates an analog RGB signal.

VGA (Video Graphics Array) Introduced by IBM in 1987 with its PS/2 models 50, 60 and 80, and also generates an analog RGB signal. It uses patented IBM chips (as does the MCGA), so some VGA clones are not totally compatible.

Some models provide additional modes over and above those of VGA, and specialised graphics adapters are available offering extremely high resolution for CAD and the like.

136. The PC and PS/2 Graphics Display Modes

Mode number		Adaptors supported										Notes
Hex	Dec	Resolution	Colours	Palette	MDA	CGA	HGC	1512	EGA	MCGA	VGA	
0	0	40 x 25 chars	16	—		●		●	●	●	●	Identical to mode 1, but gives better results for CGA with a mono display.
1	1	40 x 25 chars	16	—		●		●	●	●	●	Character cell 8x8 in CGA, option of 8x14 in EGA or 9x16 in MCGA and VGA.
2	2	80 x 25 chars	16	—		●		●	●	●	●	Identical to mode 3, but gives better results for CGA with a mono display.
3	3	80 x 25 chars	16	—		●		●	●	●	●	Character cell 8x8 in CGA, option of 8x14 in EGA or 9x16 in MCGA and VGA.
4	4	320 x 200	4	2x4		●		●	●	●	●	Only two palette selections possible: cyan, magenta, white and black; or green, red, yellow and black.
5	5	320 x 200	4	—		●		●	●	●	●	As mode 4, except with CGA when colours are cyan, red, white and black.
6	6	640 x 200	2	—		●		●	●	●	●	Highest CGA resolution – black and white only.
		640 x 200	16	—				●				Non-standard mode used on Amstrad's PC1512 range.
7	7	80 x 25 chars	2	—	●		●		●		●	Character cell of 9x14 on MDA and HGC, with option on 9x16 on VGA.
		720 x 348	2	—			●					Graphics mode of Hercules Graphics Card (HGC), as fitted to Amstrad PC1640 MD.

(Modes 8 to 0A used by IBM PC Junior only, 0B and 0C for internal EGA use only)

0D	13	320 x 200	16	64					●		●	
OE	14	640 x 200	16	64					●		●	
OF	15	640 x 350	2	—					●		●	
10	16	640 x 350	16	64					●		●	Only 4 colours on EGA with 64K video RAM.
11	17	640 x 480	2	—						●	●	
12	18	640 x 480	16	64							●	
13	19	320 x 200	256	262,144						●	●	Requires analogue input on monitor.

brightness or contrast control haven't been changed. If you own an Amstrad PC1512 or 1640, it is a good idea to check that the NVR settings haven't been changed before panicking, too. The NVR allows screen colours to be changed and it is quite possible for these to be set to a rather illegible black characters on a black background! Switch the PC off and remove the batteries underneath the monitor for a minute or so. This will reset all the NVR settings back to their defaults and can cure many strange problems.

137. Connecting Computers

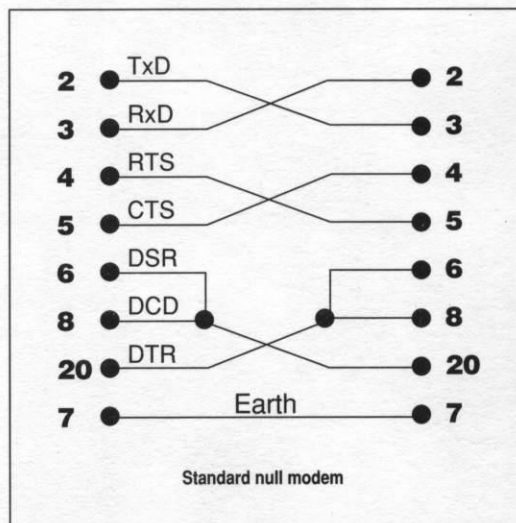
The RS232 interface, otherwise known as the 'serial port', is usually a 25-pin male D connector at the rear of the PC. Although the RS232C standard – or the CCIT standard V24 – covers all 25 pins, only eight are usually required (marked in bold below) so a nine-pin D socket is frequently used instead:

Pin	Direction	Name	Details
1		FG	Frame ground (earth)
2	out	TxD	Transmitted Data
3	in	RxD	Received Data
4	out	RTS	Request To Send
5	in	CTS	Clear To Send
6	in	DSR	Data Set Ready
7		SG	Signal Ground (earth)
8	in	DCD	Data Carrier Detect
9	in		Testing positive voltage
10	in		Testing negative voltage
11	in		Used in equalising
12	in	SDCD	Secondary DCD (used to detect high speed)
13	in	SCTS	Secondary CTS
14	out	STD	Secondary Transmitted Data
15	in	TC	Transmitter Clock (synchronous protocols)
16	in	SRD	Secondary Received Data
17	in	RC	Receiver Clock (synchronous protocols)
18			Not used
19	out	SRTS	Secondary RTS
20	out	DTR	Data Terminal Ready
21	in	SQ	Signal Quality detector
22	in	RI	Ring Indicator (for auto answer)
23	out		Data rate selector
24	out	ETC	External Transmitter Clock (synchronous protocols)
25			Not used

When connecting two bits of equipment with an RS232 cable, it is important to note the distinction between a

'terminal' and a 'modem'. Computers are regarded as terminals, so the TxD pin of one should be connected to the RxD pin of the other, the RTS pin of one connected to the CTS pin of the other, and so on. The cable wires 'cross over' between the machines.

When modems are involved, however, the cross over is assumed to take place somewhere in the telephone link, so the cable connecting the computer to its modem should not 'cross over'. This means that pin 2 at the computer end should be connected to pin 2 at the modem, pin 3 to pin 3, and so forth:



■ The cable used to connect two computers together, with the cross over, is known as a 'null modem' cable

PROTOCOLS

Once connected, it is vital that each end of the link uses the same 'protocol' to code and decode the signal. If reception is garbled one of the following parameters is probably set wrongly.

Speed:

Speeds are described either in terms of the number of bits transferred each second, or in terms of the number of times the system changes state each second. This is called the Baud Rate, and at lower speeds is the same as the rate of data transfer. At higher speeds phase modulation systems are used to increase the data transferred for the same baud rate.

These are the speeds in common use today, together with their CCIT standard V-number. Note that Viewdata systems and some bulletin boards use the asymmetrical 1200/75 standard where data is received at 1200 bits/sec, but transmitted at the very slow 75 bits/sec:

Bits/sec	Baud	Standard	Uses
300	300	V21	Amateur BBS (Bulletin Boards)
1200/75	600	V23	Amateur BBS and Viewdata
1200	600	V22	Professional systems
2400	600	V22 bis	High speed connection

Word length:

Usually either seven or eight bits are used to describe each character of data that is sent. If you are just transmitting text in ASCII format then seven bits will do, however for *WordStar* files or binary program files a full eight bits is necessary. Most bulletin boards use **eight** bits, but Telecom Gold and Prestel use **seven** bits.

Stop bits:

An 'asynchronous' system, of the sort usually used over standard telephone systems, does not use a clock to keep both transmitter and receiver in step. Instead 'stop bits' are put at the end of each character for synchronisation. Usually **one** stop bit is used, although **two** is quite common.

Parity:

Sometimes another bit is added to create a primitive error-checking system. The parity bit is set so as to make the number of 1's in the word either even or odd, as specified. Most systems set parity to **None**, however Prestel and Telecom Gold use **Even** parity.

HANDSHAKING

The receiving end of a communications link frequently has to perform other operations while receiving the data, such as storing it on disk or sending it to the printer. A 'handshaking' system provides a means for it to effectively say to the transmitter 'hang on a minute' or 'OK to continue'. This can either be done in hardware, using certain pins on the RS232 link; or in software by sending special characters. There are various standards in either case:

RTS/CTS: Pin 4, Request To Send, is connected to pin 5, Clear To Send, of the other terminal. Dropping the voltage on pin 4 pauses transfer.

DTR/DSR or **DCD:** Pin 20, Data Terminal Ready, is connected to either or both pins 6 and 8. Dropping the voltage on pin 20 again pauses transfer.

140. Recommended Reading

Despite its title, one of the best introductions to the field of communications is *Hacker's Handbook* by Hugo Cornwall, published by Century Hutchinson on 01-240 3411.

XON/XOFF: Sending ASCII code 17 (Ctrl-Q) enables transmission while ASCII code 19 (Ctrl-S) pauses the system.

ETX/ACK: Transmitter sends an ASCII code 3 or 23 (Ctrl-C or W) after a specified number of characters have been sent. Receiver responds with ASCII code 6 (Ctrl-F) when it is ready for more.

TERMINAL EMULATIONS

In addition to sending alphanumeric characters to the receiver's screen, the transmitter can also send special characters that control other aspects of the display. How these are interpreted depends on the terminal emulation being used, and many on-line databases expect particular types of terminal:

Teletype or **Dumb:** The simplest form of emulation, and usually the default. Correctly interprets carriage returns, line feeds, backspace and beep, and displays text.

ANSI: Correctly interprets escape codes in the same way as the ANSI.SYS display driver supplied with most PCs.

VT52: Interprets codes in the same way as this popular DEC terminal system.

VT100: Another very common terminal made by DEC and allowing a variety of visual effects.

Viewdata: Deviates considerably from the Teletype specification, and is vital for the correct display of Prestel screens.

138. 9-Pin to 25-Pin RS232

The following shows the connections required for a cable to connect a 9-pin RS232 socket, as found on an AT compatible, to a 25-pin RS232 plug:

9-pin		25-pin
1	Data Carrier Detect	8
2	Received Data	3
3	Transmitted Data	2
4	Data Terminal Ready	20
5	Signal Ground	7
6	Data Set Ready	6
7	Request To Send	4
8	Clear To Send	5
9	Ring Indicator	22

139. Jammed Hard Disk

If your hard disk stops working and isn't making any sound, but everything else works fine, it is quite possible that the heads have got jammed. This can easily happen on a Monday morning after the disk has been used for a few days, or after a holiday, particularly if the heads have been parked, or it is a

self-parking drive. They can sometimes be freed simply by giving the drive a hard bang, although this should be done with great care!

141. 2nd Drive for Amstrad PC2000

If you've got an Amstrad PC2000 machine the chances are you'll want to get at software and data on 5.25-inch drives as well as the 3.5-inch ones built in. Amstrad has kindly provided an easy way to add a standard 5.25-inch disk drive to the PC2000 series computers. Predictably enough, the 'official' add-on drive doesn't come cheap, but it is quite easy to do it for considerable less by assembling your own from readily available parts.

Remember though that, as with all bits of hardware jiggery-pokery, *PC PLUS* cannot accept liability for any form of damage, data loss or (heaven forbid) injury sustained while following these instructions.

First you need the drive itself, and the type you need is by far the most common made. Do make sure you are getting a double-sided drive, though. Some sold as suitable for BBC or other home computers look identical but have only one read/write head. If the supplier won't describe it as a double sided 360K IBM compatible type, then go elsewhere.

Next you need cables and connectors. For the power cable a standard four-pin DIN plug is needed at

the computer end, and a standard four-pin disk power plug (male body, female pins) at the other. The cable needs only three conductors, as there are duplicated ground pins at both ends, so ordinary lighting flex will suit. See the diagram for the connections.

You need to solder the connections at the DIN end, but the disk end uses tubular pins which crimp onto the conductor and are then inserted into the plastic shell. This makes it difficult to re-use an old plug, and new ones are not stocked by Maplin or Tandy so can be hard to find.

The data cable looks daunting, but is actually quite easy to make up using ribbon cable and IDC (insulation displacement connector) plugs. You need a 36-way IDC Centronics printer type plug and a 34-way IDC edge connector, plus a few feet of 34-way IDC ribbon cable.

Using IDC connectors means everything just presses together (press hard!) without the need to solder 68 tiny joints. The only tricky part is that the Centronics connector has room for 36-way cable, so you must make sure that the 34-way cable is inserted flush to the end with pins number 1 and 19, leaving pins 18 and 36 without a connection.

One edge of the ribbon cable has a stripe. If you use that edge for pin number one at the computer end, and also for pin one at the disk end, then all the signals will line up properly. The Centronics plug is polarised and has pin numbers marked inside it, but the edge connector can be attached either way round unless you fit it with a key to match the slot in the drive's card edge.

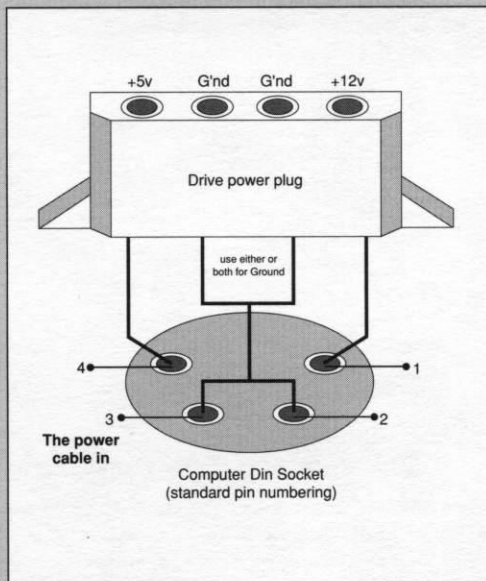
Oddly enough, this 34 way data lead is identical to the printer lead for an Amstrad CPC464 home computer. You can buy one of these already made up for not much more than the price of the bits. You will have to move the key in the edge connector, if fitted, and the cable might be longer than you want, but it should work.

Once the connections are sorted out, you have to configure the drive and computer so that the drive will be recognised for what it is. Needless to say, make sure the power is switched off before plugging anything together, and before messing about with jumpers and switches.

Check that the drive has its drive number select link set to number 2. These links are found near the edge connector, and links anywhere else on the board should be left alone. Also near the edge connector is what looks like a chip socket. This should be empty. If it has what looks like a chip in it, then remove it. This is actually a block of resistors which should only be present on drive A, and will cause your real drive A to malfunction if left in place.

Now for the computer. The little switch next to the external disk sockets must first be switched to 'EXT',

Power for the Second Drive



■ The power connection for the new drive requires a 4-pin DIN plug at one end and a standard 4-pin disk plug at the other

then plug everything together and power up. Assuming there is no smoke, your computer should now know it has an external drive attached but will probably think it is a 3.5-inch type. To tell it otherwise, enter the command:

```
device setfd /d:1 /t:0
```

Then reboot. The reboot is important as the DEVICE program writes the drive parameters into the computer's non-volatile memory and they don't take effect until the computer is rebooted. Many a do-it-yourselfer has spent hours trying to get his new drive recognised because he didn't realise he had to reboot.

Once fitted, your new drive should act just like a normal PC drive, and you will at last be able to load software from standard disks. You can even configure the computer so that the external drive is drive A, and boot from it. However, you will find that some software using 'Key disk' copy protection will not accept the key disk in the external drive.

These instructions should also work for a PC2286 and 2386, but a different technique is needed if the

machine is to recognise its new drive for what it is. Instead of using the DEVICE SETFD command you must run SETUP, as detailed in the manual.

142. Computer Hang-ups

If the computer hangs up during operation, check that it isn't trying to print something. Some applications hang if the printer is not connected or on-line, or you might have hit [Shift] [PrtSc] by mistake. The only solution is to turn the printer on quick, or re-boot.

143. Keyboard problems

If one of the keys stops working it is almost always one of the most often used keys: [Return], [space], [Esc], [E] or [T]. You can, however, get the missing character by pressing [Alt] and then typing its ASCII code on the numeric key pad. Alternatively, if you have an Amstrad PC1512 or 1640 you could use the NVR program to configure the two user-definable keys [Del->] and [Enter] to the missing keys. You will need to look up its key code in Appendix 4 of the Amstrad User Guide. ●

T H E
**HELP
SCREEN**
C O L L E C T I O N

103



A P P L I C A T I O N S

144. Lost Files In WordStar

Although MS-DOS can handle as many files in a subdirectory as you like, *WordStar* cannot. *WordStar* 4, for example, can only display up to 370 files in its directory listings, indicating the presence of additional files with ... (three dots) as the last entry. So if you find files disappearing when you enter *WordStar*, it's simply that you've got too many files in the one sub-directory. Try creating a new sub-directory and moving surplus files there.

145. Patching QuickBASIC

Microsoft's *QuickBASIC 2.01* is not fully compatible with the Amstrad PC1512 keyboard when it comes to [£] and [®], but can be made so by making the following changes:

Location	old QB.EXE	new QB.EXE
0002 A176	40h	22h
0002 A17A	23h	9Ch
0002 A241	60h	23h
0002 A252	22h	40h

Addresses are those found using *PC Tools*. Remember that DEBUG loads files by default from offset address 0100h. If you make a backup before

you start and use the backup to mess around on, you should be able to do the job in three minutes or less.

146. Running CP/M Software

There are a number of software CP/M emulators, such as *Z80EM*, available from public domain software libraries. They are, however, relatively slow: on average it takes ten instructions to emulate each Z80 instruction, which can become very tedious.

An alternative solution is to replace the 8086 processor in your PC compatible with an NEC V30. This is 8086-compatible, but runs some operations slightly faster and has the ability to also execute all instructions for the 8080 processor as well. Since CP/M was written for the 8080 this means that it can run most CP/M software at twice the speed of the original. A CP/M emulator that uses this chip is also in the public domain, but remember that all these emulators emulate CP/M 2.2 only; none will handle the extra features of CP/M Plus but most CP/M software will run on them.

147. Running MS-DOS from WordStar

As owners of *WordStar* know, the command ^KF takes you out to an MS-DOS prompt from which you can run one MS-DOS command. It is not a proper MS-DOS shell as once the command has executed you must return to *WordStar*. However, if you enter **command** at the prompt the command processor will be loaded and you can carry on for as long as you wish, entering **exit** to return to *WordStar*. Do remember though that the command processor requires space for itself, so you may not have enough memory.

Customising WordStar

If you are using the *WordStar 4* or *NewWord 3* word processor and haven't yet explored WSCHANGE (called NWCHANGE in *NewWord*) then you are using it at a fraction of its full potential. This useful utility enables you to modify the main program considerably.

The documentation may look daunting but WSCHANGE is actually quite easy to use. It is run by entering **WSCHANGE** at the MS-DOS prompt, and asks you for the 'name of the file to install' and the name of the 'file to hold installed WordStar'. The answer to the first question is **ws.exe** (or **b:ws.exe** if you have two drives); while the answer to the second question can be the same, or another filename if you wish to preserve the original. On a single drive machine you have to make sure that both WSCHANGE.EXE and WS.EXE are on the single disk.

From the Main Installation Menu select 'Computer' by pressing [C]. This leads to the Computer menu,

Memory Usage

A	Default printer driver in RAM.....OFF	NWFLG
B	NewWord RAM resident (no help msg)...OFF	NWFLG
C	Immediately load main spell dict....OFF	NWFLG
D	Main spelling dict buffer (kbytes)...145	SPMEM
E	Most recent words checked (records)..24	SPSIZE
F	Messages & menus buffer (records)...4	VMSIZE
G	Printer/update buffer (records).....8	OVSIZE
H	Text spillover, 0=auto (records).....0	TYSIZE
I	Shorthand buffer size (records).....4	HANMAX
J	Dot command buffer, RR etc (bytes)...500	DSTKSZ
K	Max header/footer size (bytes).....100	HFMAX
L	Max unerase characters (bytes).....500	UNSIZE

X Done with this menu
? Help

Enter your menu selection...

■ The Memory Usage screen allows you, as you might have guessed, to set how *WordStar* or *NewWord* uses memory

from which go to 'Memory Usage' by again pressing [C]. This takes you to the screen shown below left.

Forget about the mnemonics on the right – these are simply the names of variables used by *WordStar*. What you are presented with here is the opportunity to run *WordStar* totally from RAM; which means that it will run considerably faster and that you won't have to keep a copy of the files WS.EXE, WSPRINT.OVR or WSMGS.OVR on the disk at all times. You can even arrange to have the whole of the dictionary sitting in RAM too, which makes the spelling checker run like you've never seen before.

Selecting [A] and [B] and setting these both to ON means that the basic word processor and printing overlay will load into RAM next time the program is used – provided there is room. Setting [C] to ON means that the whole of the main dictionary file will load into RAM the next time you use *WordStar*, but beware: this takes several minutes and is only really worth it if you have a single drive machine and use the spelling checker constantly. Remember that *WordStar* loads the spelling checker into memory as it is used, so after a few minutes it is virtually operating from RAM anyway. We have this set to OFF.

While you are in WSCCHANGE you may also find it worth modifying the file WSPRINT.OVR, which stores the various printer drivers available to *WordStar*. As you are unlikely to need all the drivers, and probably only need the one, much of this file is a waste of space. By selecting 'Printer' from the main installation menu and going through the options you can generate a smaller WSPRINT.OVR that only holds the drivers you need.

If you have a twin-drive machine, and like to hold the program disk in drive A and your documents in drive B, it may have annoyed you that *WordStar* insists on logging in to drive A so that you have to log in to drive B every time before you start work. Fortunately WSCCHANGE allows you to automatically log in to drive B, or indeed any subdirectory you wish.

To do this you select 'Directory Display' from the 'Computer' menu by typing [E], which brings up the screen shown right. From here you again select [E] to change 'Initial directory log on', and enter b: as the new value. From now on *WordStar* will automatically log on to drive B from start up:

If you have carried out all these changes then the next time you boot up *WordStar* it will load straight into RAM, together with the main dictionary if you selected that option, and automatically log on to drive B.

149. Sending messages on Prestel

Electronic mail has been a weakness of Prestel when compared to a scrolling system such as Telecom Gold. On Telecom Gold you can 'upload' a text file of virtually

any length to your mailbox and let the Gold computers take care of word wrap and other formatting features. On Prestel the mailbox will only take one screen-full of text at a time.

However, a little time spent with a decent word processor should enable you to create a file that can be uploaded successfully. You will have to set the line length (or right margin) to automatically word wrap on a 39-character line, and you will have to make sure that the file does not contain more lines than one Prestel screen. It is possible to send more than one screen's worth of text, providing you send a # symbol at the end of each screen – some experimentation will doubtless be necessary!

150. Pictures in Fantasy

Unfortunately, picture files have not been blessed with a standard file format in the same way as text files. The ASCII code has long been established as the standard format for storing text, although *WordStar* chooses to do it differently, but seemingly every graphics package stores images in a different way.

For a start, there are two fundamentally different ways of storing images. 'Bit maps' take a picture and divide it into an array of points, the file consisting of a string of data, each element describing each point in the array. This is not necessarily the same as a pixel map as the bit map is normally to an 'ideal' resolution

Directory Usage

```
A Display file directory.....ON      INIDIR
B Directory in alphabetical order.....ON  DIRSRT
C File types excluded from directory...0787 NOTYPE
D File names that are shown.....?????????.???
```

```
X Done with this menu
? Help
```

Enter your menu selection...E

Currently: None

```
Do you want to change this? (Y/N) Y
Enter new value...b:
```

■ The Directory Usage screen of WSCCHANGE allows you to change which directory *WordStar* (or *NewWord*) initially logs on to

of some sort, and not to the resolution of the device that happens to be in use at the time. The IMG files produced by GEM Paint or Electric Studio's *Video Digitiser* are of this type.

On the other hand, 'object descriptor files' or 'vector image files' work by breaking the picture into a series of discrete, and simple, objects. These are then described in the file with instructions the equivalent of, for example, 'Draw a circle radius 400 at 300,800 and fill with colour 7', or even 'Put the bit-map described by file *bloggs.img* at 200,200'. GEM Draw produces .GEM files which are of this type, as do most CAD packages.

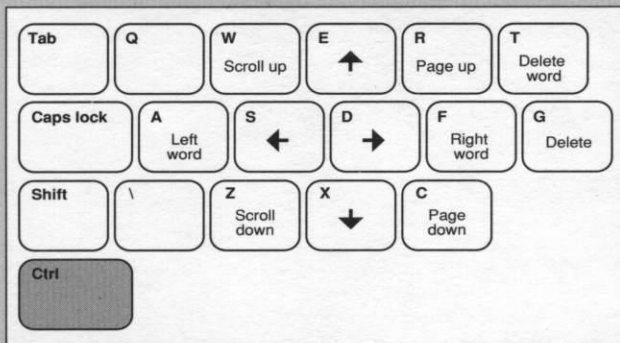
Aldus Software's Postscript language also generates this type of file. Incidentally, you can convert GEM files into IMG files by using the GEM Desktop Accessory *Snapshot*.

Fontasy can manipulate graphics created by *BASICA* in the 640 x 200 high-res CGA graphics mode. *BASICA* is the version of BASIC given away with real, genuine IBM PC's, and is a subset of Microsoft's *GW-BASIC*, which in turn is a subset of the language used by the *QuickBASIC* compiler.

In *QuickBASIC* `GET (x1, y1) - (x2, y2), arrayname` will store the image in the screen area defined by

151. WordStar Control sequence Commands

The commands used while editing a document within *WordStar*, and compatibles such as *NewWord*, are invoked using [Ctrl] in conjunction with either other single keys or multiple keystrokes. For example, the command sequence ^A is invoked by holding down [Ctrl], tapping [A] and then releasing [Ctrl]. The sequence ^PB is invoked by holding down [Ctrl], tapping [P], releasing [Ctrl] and tapping [B]. Alternatively [Ctrl] could be held down for both keystrokes and only released after the [B].



CURSOR MOVEMENT

^S or [left] Left one character,
^A Left one word,
^QS Beginning of line;

^D or [right] Right one character,
^F Right one word,
^QD End of line;

^E Up one line,
^QE or [Home] Top left of screen,
^R or [PgUp] Up one screen,
^QR or ^[Home] Beginning of file;

^X Down one line,
^QX or [End] Bottom right of screen,
^C or [PgDn] Down one screen,

^QC or ^[End] End of file.

DELETING TEXT

^G or [Del] Deletes character at cursor position;
^H or [BckSpce] Deletes character to left of cursor;
^T Delete next word;
^Y Delete line;
^QY Delete to end of line;
^Q [Del] Delete from beginning of line to cursor;
^U Undelete.

MISCELLANEOUS

^W or ^[PgUp] Scroll up one line;
^Z or ^[PgDn] Scroll down one line.

^B Aligns paragraph text as specified;
^QU or ^QB align paragraphs in rest of document;
^I or [Tab] Moves cursor to next tab stop, inserting spaces if Insert is On;
^M Insert carriage return;
^N Insert line feed, or remove hard carriage return;
^V or [Ins] Toggle Insert on or off (for overwrite).

^J Summon help Menu;
^K Summon Block & Save Menu;
^O Summon On-screen Format Menu;
^P Summon Print Controls Menu;
^Q Summon Quick Menu;
[Esc] Summon Shorthand Menu.

BLOCK & SAVE MENU

^KS Save to disk;
^KD Save to disk and exit to Opening Menu;
^KX Save to disk and leave program;
^KQ Abandon and exit to Opening Menu.

^KB Mark beginning of a block with ;
^KK Mark end of block with <K>;

x1,y1,x2 and y2 in an array. This array can then be stored on disk in the usual way. Make sure that you are in the right screen mode with a **SCREEN 2** command though, and for further details check manual.

152. Auto-loading Macros

If you're writing macros in Lotus 1-2-3 (or compatibles such as *VP Planner* and *The Twin*), it's always nice to make a note of a macro's name beside the cell in which it appears. This can then act as an identifier, a

useful jog to the memory if you've forgotten what your macro's cryptic squiggles represent.

However, macros are conventionally described by the \ (backslash) character followed by a letter or number, and that can cause problems. The difficulties stem from the fact that 1-2-3 uses the backslash for a special function – filling a cell with the same character. If you type \M, for example, what actually appears in the cell is **MMMMMM**.

The solution to this is surprisingly simple. If you define a cell as containing text (rather than numeric data or a formula) by entering ' (an inverted comma)

^KW Write marked block to a disk file;
^KC Copy marked block to cursor position;
^KV Move marked block to cursor position;
^KY Delete marked block;
^K' Converts letters in marked block to lower case;
^K" Converts letters in marked block to upper case.

^KR Insert file from disk into cursor position;
^KF Display MS-DOS prompt.

ONSCREEN FORMAT MENU

^OL Set left ruler margin;
^OR Set right ruler margin;
^OI Set tab stops;
^ON Clear tab stops;
^OX Temporarily release ruler margins (toggle);
^OG Set a temporary left margin indented one tab stop;
^OF Set ruler line to current line length;
^OO (At start of line) Insert new ruler line.

^OW Toggle word wrap on or off;
^OJ Toggle justification on or off;
^OC Centre text of current line;
^OE Insert a soft hyphen;
^OH Toggle help on soft hyphens;
^OSn Set line spacing to n.

^OD Toggle display of print controls;
^OP Toggle Preview Protect mode (print controls and dot commands not shown and text cannot be changed);
^OB Toggle display of soft spaces;
^OT Toggle display of ruler.

PRINT CONTROLS MENU

WordStar uses the ASCII codes below 30 to toggle printer effects on and off. This means that ^PA through ^PZ can be used to generate ASCII codes 1 through to 26 while ^P@ generates 0 or NUL and ^P[generates 27 or ESC. A useful feature.

Used as printer control codes the effects generated depend on the

model and the print driver loaded. These are the most usual:

^PA Alternate pitch (12 cpi or 'elite');
^PN Normal pitch (10 cpi or 'pica');
^PB Toggle bold print;
^PY Toggle italic print or alternate ribbon colour;
^PS Toggle underlining;
^PV Toggle subscript text;
^PT Toggle superscript text;
^PD Toggle double strike text.

^PX Toggle strikeout (overprint with hyphens);
^PH Next character prints over previous character;
^PO Insert a 'binding space' preventing word-wrap.

^PI Insert a proper tab (ASCII code 9);
^PL Insert a formfeed (ASCII code 12).

QUICK MENU

^QF Find specified text string;
^QA Find and replace text string;
^QG Find next occurrence of character;
^QH Move cursor to last occurrence of character;
^QI Find specified page number;
^QL Repeat last find or find and replace operation;
^QV Go to last string found.

^Kn Set marker 0 through to 9 at cursor;
^Qn Go to marker 0 through to 9 in text.
^QB Go to beginning of block marker;
^QK Go to end of block <K> marker.

^QP Go to last cursor position;
^QM Display Math calculator menu;
^QQ Repeat next character until any key pressed.

^QL Check spelling of document;
^QO Check spelling of word;
^QN Check spelling of word at cursor.

before anything else, you can type in backslashes to your heart's content. Therefore, by typing '\M, you actually get \M appearing in the cell.

On the subject of macros, if you name a macro \0 it will operate automatically as soon as the worksheet is loaded – just as AUTOEXEC.BAT behaves with MS-DOS. Similarly, if you call your worksheet AUTO123.WK1 (or WKS, depending on which release of 1-2-3 you are using), that worksheet will load itself automatically whenever you load the spreadsheet program. Combine the two – put a macro called \0 in a worksheet called AUTO123.WK1, in other words – and loading the spreadsheet package will automatically execute the macro. Great for user-friendly front-ends and the like.

153. Configuring PC Write

The shareware program *PC Write* can be configured by adding character strings to the file RULER.DEF. To stop the screen blanking when scrolling text, put %D at the start of a new line immediately after the ruler itself; to display in colour, add %C. See *PC Write's* manual for further details.

154. Hard carriage returns

A little tip for all users of *NewWord 3* and *WordStar 4* who have been bugged by all those hard carriage returns that appear at the end of ASCII files when you load them. Try keying [Ctrl] [N] and you will find them magically transformed to the soft carriage return we all

know and love. Some versions of *WordStar* can be configured to support this feature from WSCHANGE – see elsewhere in this section for details.

```
P05 L04 C01 Insert Align
!-----!
users of ^SNeuWord 3^S and ^SWordStar 4^S who have
rd carriage returns that appear at the end
load them. Try keying [Ctrl] [n] and you
y transformed to the soft carriage return
ome versions of ^SWordStar^S can be configured
from WSCHANGE - further details are
US Help Screen Collection^S (see the back of
s). Some versions use [Ctrl] [6] instead.

ordStar
ar^S know, the command ^D^KF^D takes you out to an
h you can run an MS-DOS command. It is not
s once the command has executed you must
However, if you enter ^Dcommand^D at the prompt
ill be loaded and you can carry on for as
```

■ Reading an ASCII file in *WordStar* – note the hard carriage returns signified by the '<' characters in the right-hand column

156. WordStar and ASCII

One of the oldest problems around in the world of computers is the incompatibility between *WordStar* text files and straight ASCII. The ASCII code is a 7-bit code, but *WordStar* uses the eighth bit to indicate the end of words and the like. Try TYPEing a text file created under *WordStar* and you will see what this means.

However, thanks to the PIP command available under DOS PLUS, Amstrad PC1512 owners have a simple solution. PIP is a more powerful program than MS-DOS's COPY, and allows a lot more options. The one that concerns us is Z, which sets the high-bit of the destination file to zero. The format would be:

```
PIP a:ascifile=a:wsfile[z]
```

Remember that PIP is a DOS PLUS external command, so the disk will have to contain a copy of the utility program PIP.CMD. Remember also that PIP works the other way round to COPY, in that the destination is specified first, followed by the source filename.

157. The Amstrad clock

There is a problem that can occur when some programs are used with the Amstrad mouse driver. The problem is that the Amstrad mouse driver uses the Amstrad clock 'tick' to see if the mouse has moved. In order to get a good response the mouse driver sets the clock running at three times its normal speed, passing

155. NewWord in monochrome

Owners of a monochrome PC1512 who use *NewWord* will find that areas of the screen that are highlighted suffer from a strange fuzzy effect. The following patch can be applied to *NewWord 2.18* in order to correctly display marked blocks:

1. From the system prompt, enter winstal nw.exe
2. Select option H from the menu (special patches).
3. Type an equals sign (to Enter Hex Address).
4. When prompted for the address, type 02D2 [enter].
5. A row of numbers will appear on screen:

```
02D2 34 08 C3 8A E0 8A E9 B2 08 43 D0 E9 73 07 2E      22.....C..s.."
```

and the cursor will sit below the 34. Type 0000 and then x to change the 34 08 to 00 00, and exit.

6. Type x, x and n to exit the install program and implement the changes.
7. *NewWord* will now correctly display a marked block.

If you are running *NewWord 3.0* the the operation remains the same, but you should run enter nwchange nw.exe and the address is 02CE.

only every third 'tick' on to the operating system so the clock routine can operate correctly, oblivious of what the mouse driver is doing.

Some Microsoft programs reset the clock to its proper speed for their own purposes, but unfortunately the mouse driver isn't told and continues passing on only every third 'tick'. The result is the operating system clock runs at a third of the right speed.

This problem only occurs when the MOUSE driver is loaded in conjunction with certain software, notably Microsoft's *QuickBASIC* and *GW-Basic*, and can be avoided simply by not loading the mouse driver. An alternative is to modify the Amstrad mouse driver, MOUSE.COM as shown below.

Once this is done, re-boot your system. The modification does not alter the mouse when used under GEM, but apparently some PC1512's actually have the mouse driver usually supplied with the PC1640. This can be confusing, particularly as they are both labelled version 5.0! If the PC1512 hack does not work on your mouse driver, try the 1640 version.

Incidentally, a [return] by itself is the way you tell DEBUG to stop assembling, which is why you have to press [return] twice after entering some of the DEBUG commands above. Make sure you use a copy of your master disks before doing anything with DEBUG – the results can be unpredictable unless you know exactly what you're doing!

158. Running Too Fast

A side effect of the problem of the Amstrad mouse

driver running the clock three times too fast is that it can fool speed measuring programs such as System Information from *PC Tools*, or SI from *Norton Utilities*.

These programs work by reading the system time, doing a huge number of tasks designed to simulate what a PC's chip does in normal use, and then reading the time again. It then compares the time your PC took with the time 'the average PC' – typically a creaky old 4.77MHz 8088 IBM – would take to do the same tasks. That's what the rating is actually measuring.

But with the PC1512 mouse driver installed, system time is running in slow motion. If the tasks take six seconds, the system time will only have advanced two seconds – making your PC look phenomenally fast as far as the speed-rating program is concerned.

If you divide the ratings you get in this situation by three you get a much more believable picture of your PC's performance.

159. Border Colour

Some applications change the colour of the border of the screen, and forget to change it back again when you leave. The easiest way of re-setting the border colour, and indeed the whole screen, is with the MS-DOS command CLS. An alternative is `mode 80`, but this is an external command and so requires the presence of MODE.COM on the disk.

Another way of controlling the border colour is with DEBUG. Entering `o 03D9 1` on the DEBUG prompt sets the border to blue (1512 only), using a different hex digit instead of the figure 1 alters the colour.

Modifying the Amstrad mouse driver with DEBUG

On a PC1512

`debug mouse.com`

`A 1AFB`

`MOV AX,FFFF`

`A 0910`

`JMP 0912`

`W`

`Q`

On a PC1640

`debug mouse.com`

`A 1B3B`

`MOV AX,FFFF`

`A 0948`

`JMP 094A`

`W`

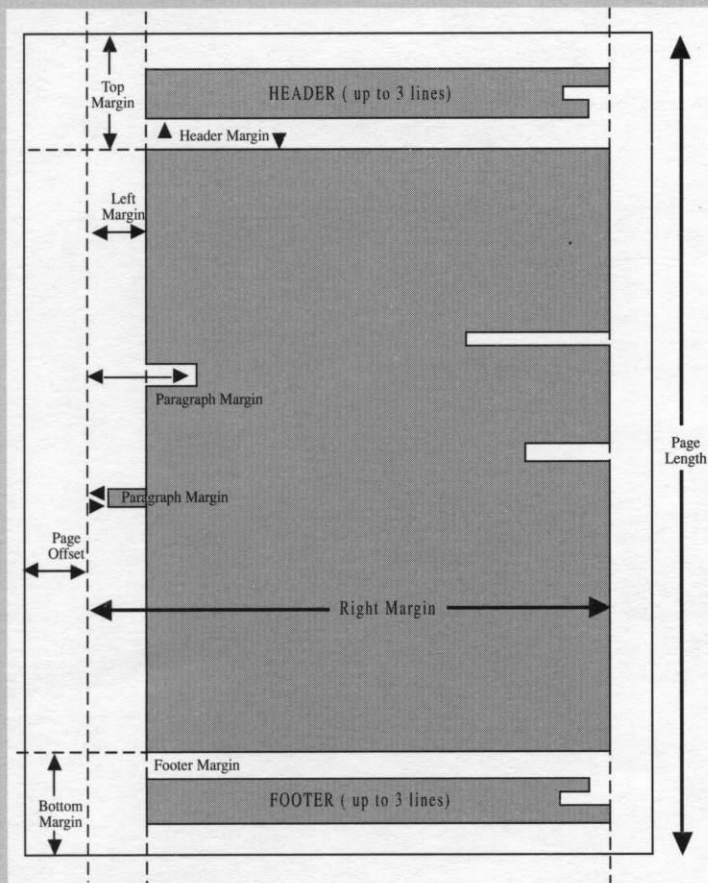
`Q`

160. WordStar Dot Commands

In addition to the 'real time' commands invoked by the [Ctrl] key, *WordStar* supports a range of dot commands that can be used in the document itself. Dot commands, so called because they start with a full stop, must be entered immediately after a hard carriage return (or at the very start of a document) so they always fall in column one. They can be in upper or lower case.

Some take a numeric parameter, in which case a typical example of its use is shown here. Some take *on* or *off* as a parameter; these are shown *on*. Others take *text* strings or a *filename*, as shown.

Page format parameters



■ All the margins are measured in columns or lines. Note that the number of lines of on-screen text per page is the Page Length minus both Top and Bottom Margins. Note also that the number of lines in the header, plus the Header Margin, should be less than or equal to the Top Margin. The same applies to the footer, as well.

Commands which only affect the printed result, and not the on-screen appearance of the document, are indicated with a colon in the far right column of the screen. Commands which affect both screen and printing are shown with a full stop.

PAGE FORMAT

See diagram for a definition of terms:

- .H1 *text* or .HE *text* Text for first line of header;
- .H2 *text* for second line, .H3 *text* for third;
- .HM 3 Sets number of lines for header margin (default is 2);
- .MT 2 Sets top margin (default is 3 lines);
- .F1 *text* or .FO *text* Text for first line of footer (by default this is the page number);
- .F2 *text* for second line, .F3 *text* for third;
- .FM 3 Sets number of lines for footer margin (default is 2);
- .MB 5 Sets bottom margin (default is 8 lines);
- .PO 3 Sets page offset from left edge (default is 8);
- .LM 4 Sets column number for left margin (default is 1);
- .RM 54 Set column number for right margin (default is 65);
- .PL 72 Sets page length (default is 66);
- .PM 6 Set left margin for first line of each paragraph;
- .PM or .PM 0 Set paragraph margin same as normal left margin.

PAGE NUMBERING

- .OP Turns off printing of page numbers;
- .PG Turns printing of page numbers back on;
- .PN 54 Continue page numbering from specified number;
- .PC 40 Specifies column for page numbers (default 28);

GENERAL COMMANDS

- .PA Force a page break;
- .CP 10 A conditional page break that will occur if necessary to keep next 10 lines on one page;
- .LS 2 Sets line spacing from 1 to 9 lines;
- .. *text* or .IG *text* A comment line that is not printed;
- .OJ *on* Turns right justification on/off;
- .RR *text* Insert ruler line;
- .IX *text* Use *text* as an index reference;
- .TC *text* Use *text* in the Table of Contents.
- .AW *on* Turns word wrap and paragraph aligning on/off;

PRINT FORMATTING

These take affect when the document is printed:

- .FI *filename* *c* Insert file (the *c* is optional and allows you to change disks);
- .XE 1B 0F Define hex string sent by a ^PE character;

.XQ 1B 32 Define hex string sent by ^PQ;
.XR 1B 31 Define hex string sent by ^PR;
.XW 1B 0E Define hex string sent by ^PW;
.XL text Define form feed string (default hex 0C);
.CW 10 Sets character width in 1/120th inch,
default is 12;
.LQ on Turns NLQ printing on/off;
.PS on Turns proportional spacing on/off;
.SR 4 Sets superscript/subscript carriage roll in 1/48th
inch, default is 3;
.LH 10 Sets line spacing in 1/48th inch (default is 8);
.BN 4 Selects sheet feeder 1 through to 4
(0 disables feeder);
.UL on Underlining of spaces on/off.

MERGE PRINTING

These commands are used in mail-merge operations. *var*, and *var1*, *var2* and so on, are variable names that would be written as &*var*&, or &*var1*& in the text itself:

.AV "text", *var* Use the *text* (optional) to prompt for
the contents of *var*;
.MA *var*=*equation* Specify an equation to determine
the contents of *var*;
.SV *var*=*text* Set contents of *var* to *text*;

.DF *filename* Read specified data file;
.RF *var1*, *var2* . . Read variables from file;
.GO top Skip to the top/bottom of the file;

.IF *condition* Start an IF statement;
.EL Begin an ELSE statement;
.EI End an IF statement;

.PF *dis* Sets paragraph alignment of text during
printing on/off/discretionary (only when a variable is
used in the paragraph).

161. Printing a Spreadsheet

Successfully printing a wide spreadsheet on standard 8-inch or A4 paper is one of the great challenges of PC computing. The following suggestions apply to a Lotus 1-2-3 compatible spreadsheet and an Epson FX dot matrix or compatible printer. Whichever combination you own, resign yourself to a little experimentation before you get usable results.

1. It is best to start by just having a go and see what you get. Enter /PPO to summon the Printer Options menu. This is where you control much of the appearance on the page, but for now simply press [H] for Header and enter the line of text you want to appear on the top of each page.

2. Press [Esc] to move up to the previous menu, and

press [R] to define the range (some compatibles may expect the command /PPR to select Printer Range). Range to Print [Wide 0:Pages 0]: B2 or something like it, may appear in the top window.

3. Specify the range you want printed, for example A1..M27.

4. Select Range again, and this time

Range to Print [Wide 124:Pages 1]: A1..M27 appears. This informs you that the selected range is 124 columns wide and one page deep.

5. Press [Return] to accept, and select Go from the PrintTo Printer menu.

What you get depends on the printer, but with a reasonably Epson compatible model you should get a respectable result but probably over several sheets of paper. Go back to the PrintTo Printer Options menu and note that the Left and Right Margins are set to zero and 80 respectively, while Pglines (the number of lines on a page) is set to 66. Most compatibles are quite intelligent when it comes to printers, and are able to work out that they will never get 124 columns on a sheet with margins set at 80, so print the last 44 on a separate sheet.

It would be possible to get more on one page if you set the printer to a condensed type face. This is usually done by sending the printer the appropriate ASCII codes first, usually detailed in your printer manual. On the Epson FX it is either ASCII code 15 or ASCII code 27 (frequently called ESC) followed by 15. Fortunately many Lotus compatibles provides for printer Setup codes, so proceed as follows:

1. Enter /PPO to get to the printer options menu, and press [S] for Setup.

2. Enter \027\015 as the setup code. The slash indicates an ASCII code rather than the actual characters zero followed by two and seven.

3. This puts the printer into condensed mode, however the right margin is still set to the default 80 characters. On an Epson printer normal character pitch is 10/inch while condensed packs 17/inch. This means that we can get 80*17/10, or 136 characters across the page – quite enough to fit in our spreadsheet. So press [M] to select Margins, [R] to select Right, and enter 136.

4. Press [Esc], check that the range is as previously set, and press [G] to try again!

If you are lucky, and your printer is reasonably Epson-compatible, you should get something quite usable. If

not, these are some common problems:

LINES WRAPPING ROUND

Usually because the margins set incorrectly on the /PPO menu. Make sure Left Margin is set to zero, and Right Margin to the number of characters that can fit across the page in the required typeface. You can find this either by multiplying the specified character pitch by the width of the page in inches, or simply by counting characters.

INCORRECT PAGE LENGTH

1. The number of lines on a page is specified on the /PPO menu under PgLines. Default is 66, which is suitable for a line pitch of 6 on standard 11-inch paper. You can find out the line pitch either from your printer manual or by measuring off an inch and counting.
2. If the printer has a DIP switch controlling 'Skip perforation', make sure this is set OFF. Set ON it automatically advances four lines at the end of each page.
3. Make sure any DIP switch on the printer controlling 'Page length' is set correctly.

SUPERFLUOUS LINES

Most spreadsheets send a carriage return command with line feeds. Make sure any DIP switch on the printer controlling 'Automatic line feed' is set OFF.

INCORRECT TYPEFACE

Again, check your manual and the DIP switches at the back of the printer. Make sure it is set in Draft mode and not NLQ. Re-check the control codes required to select a condensed typeface.

162. SuperCalc with a Serial Printer

Although *SuperCalc 3.1* does not directly support a serial printer, here are a number of suggestions that might help:

1. It is worth trying the MODE command in MS-DOS. Add this to your AUTOEXEC.BAT file:

```
MODE COM1: 1200,N,7,1
MODE LPT1: = COM1:
```

The file MODE.EXE must be available on a suitable path for this to work. This sets COM1 to 1200 band, no parity, 7 data bits and 1 stop bit, and then re-directs all output from LPT1 (the printer port) to COM1 (the serial port). Of course you need to match the COM1 settings to those used by the serial printer.

2. Use the command /ODD (Output, Display, Disk). This saves the file as a PRN file which can then be printed

using `copy file.prn prn`.

3. The output can be routed to the serial port via the graphics options screen. This is accessed from the spreadsheet with /GGO (Global, Graphics, Options) and by entering [S] in the plotter interface field, and COM1 under the serial options.

163. Custom Print Commands

With *NewWord 3* and *WordStar 4* certain [Ctrl] [P] print control characters can be customised for the user's own purposes. NWCHANGE (or WSCHANGE) allows up to six bytes to be allocated to the characters ^Q, ^W, ^E and ^R. The first byte simply stating how many bytes are in use, while the remaining five hold the actual characters that are to be sent to the printer when required.

For example the following sets up ^Q to send the sequence 'ESC 0' (1B 30 in hex) which would switch an Epson compatible printer to eight lines per inch. It assumes you are running *NewWord 3*, that the disk with the program NWCHANGE.EXE is in drive A, and your *NewWord 3* work disk is in drive B:

1. From the system prompt, enter:
`nwchange b:nw.exe`.
2. From the main installation Menu select B for Printer.
3. At the printer Menu choose C for Driver Patches.
4. At the printer Driver Patches Menu select A for custom print controls.
5. At that menu select A (for Print control Q).
6. On the screen will be a row of numbers:
0498 00 00 00 00 00...
and so on. The cursor will be below the first '00'. Type 02 1B 30. Then press [x] to update that line and exit. Pressing [x] repeatedly will take you back through the menus and, when asked if you want to make any more changes, typing [n] implements the changes to NW.EXE and gets you out of NWCHANGE.

You can now switch to eight lines per inch simply by typing [Ctrl] [P] [Q] within your document. In this case it would be a good idea to set up, say, ^R to switch back to six lines per inch.

It is also possible to define the effect of these control codes from within a *WordStar* document by using the dot codes .XQ, .XW, .XE and .XR followed by the control sequence. For example, to define ^PQ as above you'd use the dot command .XQ 1B 30.

164. Ghostly Images

Users of Lotus 1-2-3 graphics and AutoCAD release 2.03 may find their graphics remarkably persistent on the screen of an Amstrad PC1512, even after the user has returned to the MS-DOS prompt! The problem is connected to the non-standard colour/intensity planes of the Amstrad PC1512.

The PC1512 is a strange beast, graphically speaking. In most of its screen modes, it's a perfectly standard CGA (Colour Graphics Adaptor) machine. But its 640 by 200 pixel graphics screen, variously known as Mode 6 or Graphics Mode 2, is a little bit different. Although it mimics a standard CGA 640 by 200 monochrome screen for most purposes, with the right software this mode can actually display up to 16 colours at this resolution. The extra capability is good news as far as GEM users are concerned – they'd be stuck in mono otherwise – but it can cause problems with a few software packages.

The problem with Lotus 1-2-3 release 1A is that, though the program draws graphs perfectly well in 640 by 200 mode, it doesn't clear the screen properly afterwards. The upshot is that, as successive graphs were drawn, the new curves simply overlaid each other to produce a scrambled mess onscreen. Even quitting and re-running Lotus doesn't get rid of the built-up curves.

It looks as though what happens is something like this:

1. Lotus clears the screen memory it expects the 640x200 mode to use;
2. Lotus selects 640x200 mode;
3. Lotus draws the graph;
4. Lotus switches back to its normal 80-column text mode.

On a standard CGA machine this would be a reasonable way of doing things, but for a PC1512 the first two steps are the wrong way round. In order to allow 16 colours the 1512's 640 by 200 mode uses a great deal more display memory than other modes: a whacking 64K, compared to the 16K normally used for the same resolution in mono. The extra 48K can only be accessed once the machine is in 640 by 200 mode, so it won't get cleared in step 1. It will get altered by step 3 however, because of the way the 1512's innards are designed.

So every time a graph gets drawn, more bits of curve are added to the 48K of extra display memory, without any of it ever getting wiped off again. Simply exiting isn't enough to do the trick, because that doesn't wipe the memory.

On some machines a simple `cls` wipes this memory clean, but on other machines a special

program appears to be needed. Shown in the panel to the right is a DEBUG script for CLR1512.COM. Enter it into a text file called CLR1512.TXT, and don't forget the extra carriage return just before `rcx`. Then enter `debug <clr1512.txt` at the MS-DOS prompt, and the file CLR1512.COM will be created. You'll still need to use the CLS command to get a usable screen after running CLR1512.COM:

165. Umlauts in Microsoft Word

You can use the macro facility of Microsoft Word (at least in version 4) to generate a ü (umlaut) as follows:

1. Press [Shift][F3] to begin recording the macro;
2. Hold down [Alt] and type the number 129 on the

The script for CLR1512.TXT

```
a100
mov al,1a
mov dx,03D8
out dx,al
mov ax,0000
mov cx,B800
mov es,cx
mov di,0000
mov cx,2000
rep
stosw
mov al,09
mov dx,03D8
out dx,al
mov ax,4C00
int 21

rcx
22
n clr1512.com
w
q
```


- numeric keypad, to enter the code for ü;
3. Press [Shift][F3] again to finish recording the macro;
4. Type `uü1^` followed by [Ctrl][u] and [Return], to save the macro to the glossary.

Note that in step 4, the ^ refers to the caret key, usually [Shift][6]. The ü is now saved in the glossary under `uü1` but can also be generated by simply typing [Ctrl][u] while writing.

166. Rounding Errors

When using a spreadsheet such as Lotus 1-2-3 or one of its many work-alikes, it is important to be aware of the difference between the value actually held in a cell and the value that is shown on screen or paper. The function @ROUND has much the same visual effect as setting the format of a cell to display a fixed number of decimal places. Although both actions round the value that is displayed, only @ROUND affects the value actually held in the cell.

This can be particularly hazardous when applying multiplication or division to monetary values, when it is usual to set a global format of just two decimal places. The example used here shows a list of costs, A3 to A10, from which we wish to calculate both the VAT due on each item and the overall total due.

Column B shows the result of applying the simple

formula `0.15*A3`, and all looks well until the total appears in B12, which is blatantly wrong. This is because the values displayed on screen have been rounded to two decimal places, but the actual contents of the cells have not, and the function @SUM operates on the actual contents of the cells specified. The values in column E may look the same, but the application of the @ROUND function means that the total in cell E12 is now correct.

Less error would theoretically creep in if we found the total VAT owed from the total costs, using the formula `@SUM(A3..A10)*0.15` as in cell B14, but VAT is not calculated like that. There are often different ways of finding out the same thing, which makes it worth considering the rounding errors that might creep in before deciding which one to use.

167. Calculations in WordStar

You know that moment when, halfway through writing that urgent business letter, you need to work out what an ex VAT price would be inc VAT? Or what the length of a 5 foot 3 beam would be in metres? That's no problem if you've installed a memory-resident calculator program, can find your little Casio LCD model under all that paperwork, or have a mathematical prodigy handy.

It's also no problem if you've got *NewWord* or one of more recent, NW-descended versions of *WordStar*. Hit [Ctrl][Q] followed by [M] from the middle of a document, and up pops the built-in calculator. There's full onscreen help for this (even if you're working at help level 0) and you can easily insert equations or results in your documents. Hit [Esc][=] to insert the last result or [Esc][#] to insert the equation that produced it.

The same calculator module can perform maths on equations already in your documents. Mark the equation as a block using [Ctrl][K][B] and [Ctrl][K][K], hit [Ctrl][K][M] to find the result, and then use [Esc][=] to insert the result at the cursor position. Easier than getting the abacus out, wouldn't you say?

168. Reading from Sub-directories

Some application packages will not read data files held in sub-directories, but expect them to be held either in the same directory as the application or in the root directory of another drive. You can get round this using the MS-DOS utility program SUBST.EXE. Before using the application run a batch file containing the command:

SUBST E: \LETTERS

MS-DOS will substitute the sub-directory \LETTERS when you select drive E.

Rounding Errors in your spreadsheet

F1:Help 2:Edit 3:Macro 4:Abs Ready ! 5:Goto 6:Window 9:Calc F10:Graph									
B10 (F2): 0.15*A10									
	A/	B/	C/	D/	E/	F/	G/	H/	
1	Costs	VAT due	(formula)		VAT due	(formula)			
2									
3	12.77	1.92	0.15*A3		1.92	@ROUND(0.15*A3,2)			
4	34.77	5.22	0.15*A4		5.22	@ROUND(0.15*A4,2)			
5	12.77	1.92	0.15*A5		1.92	@ROUND(0.15*A5,2)			
6	6.77	1.02	0.15*A6		1.02	@ROUND(0.15*A6,2)			
7	34.77	5.22	0.15*A7		5.22	@ROUND(0.15*A7,2)			
8	3.77	0.57	0.15*A8		0.57	@ROUND(0.15*A8,2)			
9	10.77	1.62	0.15*A9		1.62	@ROUND(0.15*A9,2)			
10	63.77	9.57	0.15*A10		9.57	@ROUND(0.15*A10,2)			
11									
12	180.16	27.02	@SUM(B3..B10)		27.06	@SUM(E3..E10)			
13									
14		27.02	0.15*@SUM(A3..A10)						
15									
16	The effect rounding can have when adding up a column of figures.								
17	Column B has been formatted to show two decimal places, while column								
18	E has been rounded to two decimal places. The sum in cell E12 is the								
19	correct total of the figures in column E.								
20									
Free:98 x [334k] Auto [TEST-2.WKS] . Num . 10:47:10 am									

■ Using the @ROUND function to ensure monetary calculations, particularly of VAT, are calculated correctly

169. Macro Commands

Although there is a degree of compatibility between Lotus 1-2-3 and a number of other spreadsheets, not all these commands work with all packages:

MOVEMENT COMMANDS

In the following *n* can be omitted altogether if the command is only to be used once:

{**LT** *n*} and {**RT** *n*} – Move left or right *n* cells (some spreadsheets use *L* and *R*).

{**DN** *n*} and {**UP** *n*} – Move down or up *n* cells (some use *D* and *U*).

{**PGUP** *n*} and {**PGDN** *n*} – Page up or down *n* times.

{**BIGRIGHT** *n*} and {**BIGLEFT** *n*} – Move *n* screens right or left (some use **PGRIGHT** and **PGLEFT** instead).

{**HOME**} – Go to cell A1.

{**END**} – Toggle [End] on and off.

{**ESC** *n*} – Press [Esc] *n* times.

{**DEL** *n*} – Delete to right of cursor *n* times.

{**BS** *n*} – Backspace *n* times

{**GOTO** *cell*} or {**GOTO**}*cell*~ – Move to specified *cell* or range.

SOUNDS AND PAUSES

{**BEEP**} – Sound a beep.

{**WAIT** *time*} – Pause until specified *time* serial number.

{**TONE** *freq,t*} – Sounds a tone of pitch *freq* Hertz for *t* milliseconds. Set *freq* to zero to pause execution for specified time.

{**DELAY** *n*} – Hold display of a graph or the Worksheet Summary screen for *n* thousandths of a second. If *n* is zero pauses display until key pressed.

CHANGING CONTENTS

{**BLANK** *range*} – Erase specified cell or range.

{**LET** *cell*, *expression*} – Evaluate *expression* and put result in *cell* without moving cursor.

{**PUT** *range,col,row,data*} – Put *data* string or value at the specified number of *cols* and *rows* into the block of cells specified by *range*.

{**CALC**} – Re-calculate spreadsheet.

{**RECALC** *address,test,n*} – Recalculates *address* range or from *address* cell row by row. *test* and *n* are optional – if included recalculation is repeated *n* times or until *test* is true.

{**RECALCCOL** *address, test,n*} – Same as **RECALC**, but recalculates column by column.

PROGRAM FLOW

{**subroutine** *arg1,arg2,..*} – Calls the named *subroutine* macro, passing values *arg1*, *arg2* and so on. Arguments must be as listed in **DEFINE** statement (some spreadsheets use **CALL** followed by the subroutine name)

{**DEFINE** *cell1:type1,cell2:type2,..*} – Used in first line of a subroutine to define *cell* and *type* (*string* or *value*) of arguments passed to the subroutine.

{**RET**} or {**RETURN**} – Return to next cell after the **CALL** instruction which transferred execution to this subroutine.

{**EXIT**} – leave a subroutine called by **CALL** before **RET** is encountered.

{**RESTART**} – Terminate the main macro when subroutine macro has finished execution.

{**BRANCH** *cell*} – Execute macro commands at *cell* (or range) specified. Some use **JUMP** instead.

{**DISPATCH** *cell*} – Execute macro located at the address to be found in *cell*.

{**ONERROR** *cell1,cell2*} – When an error occurs, branch to macro at *cell1* and store error message in *cell2*. Will only trap one error.

{**FOR** *count,start,stop,step,macro*} – Sets the cell at location *count* to initial value *start*. Repeatedly executes *macro*, incrementing the value in *count* by *step* each time. Stops when *count* reaches value of *stop*.

{**FORBREAK**} – Prematurely terminates execution of a **FOR** command.

{**IF** *test*}*action* – If *test* is true execute *action*, else execute next cell down.

{QUIT} – End macro and return to 'Ready' mode.

USER-DEFINED MENUS

{MENUCALL cell} – Transfers execution to user-defined menu at *cell* and CALLs macro identified by user's selection.

{MENUBRANCH cell} – As above, but BRANCHs to selected macro. Some use MENUJUMP.

{MENUWIDTH n} – Specifies width of pop-up menu as *n* characters, to a maximum of 30.

{MENUXY col,row} – Specifies screen coordinates of top-left corner of pop-up menu.

TAKING INPUT

{GETNUMBER "prompt",cell} – Displays the string *prompt* in the control panel and waits for user to enter a value. Places value in specified cell. Some use INVALUE.

{GETLABEL "prompt",cell} – Much the same as GETNUMBER but inputs a label rather than a value. Some use INLABEL.

{INRANGE "prompt",cell} – Displays *prompt* string and inputs a range, either defined by cursor keys or direct entry, into specified cell.

{GET cell} or {INKEY cell} – Places next character pressed on keyboard into specified cell.

{?} – Pauses execution of macro so user can input keystrokes, returning to the macro when [return] pressed.

{LOOK cell} – Copies first character found in keyboard buffer into *cell*.

CONTROLLING DISPLAY

{PANELOFF} and {PANELON} – Disables and re-enables panels and pop-up menus. Some use MENUOFF and MENUON.

{DISPLAYOFF} and {DISPLAYON} – Disables and re-enables the display of the whole screen, leaving just the contents of the cells.

{WINDOWSOFF} and {WINDOWSON} – Disables and re-enables updating of the screen. Some use SCRNOFF and SCRNON.

{BORDEROFF} and {BORDERON} – Turns off and re-displays column and row borders.

{BREAKOFF} and {BREAKON} – Stops [Ctrl-Break] from

terminating execution.

{CLRSCR} – Clears screen.

{INDICATE "text"} and {INDICATE} – Display *text* (up to five characters) in mode indicator, or reset to normal.

170. Leather Goddesses

Here are a few hints to help you with the adventure game *Leather Goddesses of Phobos* by Infocom:

1. To enter the house – NQRFN RQ WKH GRRU.
2. To get out of the cage – HDW VRPH VXJDU WR LQFUHDVH VWUHQJWK.
3. To carry all you find – SXW WKLQJV LQ WKH EDVNH.
4. To get the headlight – PDNH D URSH IURP WKH VKHHW.

You also could try NLVVLQJ WKH IURJ.

(Answers are in code – a letter transposition so A=D, B=E, C=F and so forth.)

171. Trinity

Some clues for frustrated fans of the Infocom adventure *Trinity*:

1. The key to the exit from the ossuary is the VPDOOKROH in the ZDOO of the EDUURZ and lots of GLJLQJ in the ERQHSLOHV.
2. The lump of metal is hot, as the reader must have found! How about cooling it with an LFLFOH?
3. The toadstool doors will open at the WRXFK of a VKDGRZ.
4. To cross the grass: EUROOBDQGSUDP.
5. Survive in space: EXEEOH.
6. Get the skink: ODQWHUQDQGVSOLOQWHU.
7. Get the coconut: GROSKLQ.
8. Keep the lemming: FDJH.
9. Escape from Japan: RULJDPL.
10. Get the meteor: LFLFOH.
11. Cross the river: VSDGHDQGFUBSW.
12. Get the binoculars: ODQWHUG.
13. Avoid the snake: OHPPLQJ.

172. Hitchhiker's Guide to the Galaxy

Possibly the most infuriating, and certainly the most famous, puzzle in the history of adventure games is how to get the Babel Fish from the Babel Fish Dispenser in the Infocom adventure *Hitchhiker's Guide to the Galaxy*. It illustrates one of the most common problems that people encounter when playing

adventures, particularly Infocom adventures: you have solved a series of relatively minor problems, with some personal satisfaction, and are feeling fairly confident in the little universe you have carved out for yourself, when you realise that to get through to more of the game means solving a real humdinger of a puzzle. For days you battle with it, but get nowhere.

The answer to all adventure puzzles should be hidden in the text you have already uncovered, and possibly quite early on in the game, when you weren't looking for it. The secret is to examine very carefully the text about (in this case) the 'upper cleaning robot', and then go back – possibly to the very start of the game – looking for clues.

If you are really stuck, then remember the MXQN PDLO that you could have picked up if you'd thought about it. Here are a few more tips for this adventure:

1. The significance of the digital watch: WKHUH LV QRQH.
2. How to get the digital watch: BRX FDQQRW.
3. Where Ford, Zaphod and Trillian have gone: WR WKH VDXQD.
4. How to open the hatch: LW LV D ORQJ LQYROYHG VWRUB.

173. Cheating at Elite

Here are just a few of the many tips supplied to *Help Screen* by players of the Firebird space trading/combat game *Elite*:

1. If you press [PrtSc] while holding down [Alt] and [Shift] *Elite* saves the entire screen to disk, recording for posterity your latest victory. After you leave *Elite*, enter **screen** and your screen dumps will be displayed one after another as you press a key. Press [Esc] to exit. The screens can be printed to paper from the SCREEN utility if you have the MS-DOS utility GRAPHICS.EXE loaded.

2. As all aficionados know, if you scoop up the remains of your last victim in your fuel scoops then you receive one ton of alloy for every fragment you pick up. However this routine can be extended slightly by doing the following:

Once you have destroyed the enemy ship and scooped up its remains, go for its cargo. Approach the cargo canister and match speeds so it is dead ahead and quite large in your sights. Once you are lined up, open fire and then accelerate rapidly. You will now pick up the remains of the canister and be rewarded with two to four tons of Alloys, instead of the original one ton of unknown cargo. This technique can be applied to asteroids, boulders, missiles (a risky business!) and escape pods (so avoiding the one ton of slaves that is

normally received).

While on the subject of escape pods, if you destroy the capsule just before it is scooped up (and that means just before!) you are credited with the ton of slaves plus the two to four tons of Alloys from the debris as well.

3. A useful way of acquiring kills with no risk to yourself is as follows:

Dock at an Anarchy world. This may prove difficult for some pilots and so for the less competent hyperspace into the system and then use an escape pod. Once you have docked and saved the game, launch and hover a slight distance away from the space station, but not too far in case you leave the protection zone (denoted by the 'S' in the centre of your console).

At any Anarchy world there are more ships present in your area of space than at any other world (hence the relative peace of Corporate States as fewer ships are generated) and if you wait for a few minutes, a whole multitude of ships will appear, mostly all Wolves or Hunters but use your I.D. scanner to check.

Once you have identified an enemy, open fire on him only and dogfight him till his demise (But do NOT leave the space station zone!). Providing you have not inadvertently fired at the space station (with a missile or laser), you will be given your bounty and the Vipers will not come out and attack you as you are in the space station Zone and entitled to the protection of the Galcop police. The Pirates will not attack and since the police only take note if you attack one of their ships or the Space Station, you are free to dock at any time you choose.

4. First launch your Cobra, then pause the game with [Esc], press [Alt][W] and return to the game. If you now Hyperspace you will end up in Witch Space surrounded by Thargoids.

5. The following Commander file was donated by reader Mike Davis, and gives you a Cobra Mk III fitted with two military lasers, a docking computer, extra energy unit, fuel scoops, ecm system, cargo bay extension and four missiles. You also get three hundred credits, and the rating 'Elite', but you still have a long way to go to earn the title 'Archangel'. The ship is docked at Cerex in galaxy 2.

To use GIFT you first need a blank disk, formatted as a system disk by entering **format b:/s** at the A-prompt. Copy on to this DEBUG.EXE and ATTRIB.EXE from your MS-DOS system disk, plus all files from your current 'ELITE' disk.

You now need to create the ASCII text file shown below, called GIFT.TXT.

Now enter **debug<gift.txt** at the A-prompt.

DEBUG will load and the instructions held in the text file will automatically be executed, leaving you back at the A-prompt with an extra file called GIFT.CDR on the disk.

It is very important that GIFT.TXT is accurately typed in. By way of assistance, this is the ASCII dump of the resulting file: what you should see on the right-hand side of the display after entering `debug gift.cdr`. Careful examination might reveal how to give yourself a few more benefits!

```
Commander file..
.COMMANDER .
  300.0 Credits.
..t..'.NMNMP@NM.
CEREKE. ...D....
(....(P.CEREKE
.....(.....(
P.....
.....GI
FT.CDR.....
.....
.....;...
.....
```

This should work fine, but on occasion DEBUG is thrown by spurious `0Ah` characters (line feeds shown as `^J` on screen) when instructions are piped in this manner from a text file, and crashes. We are not sure why this happens, but it seems happiest when freshly booted from a floppy containing very basic CONFIG.SYS and AUTOEXEC.BAT files.

Before you can use your new ship, you must reset the archive bit of the new file. Enter `attrib -a *.cdr` at the A-prompt. Load up *Elite* and you'll find a new commander file waiting for action!

6. Here is a list of the attributes stored in your CDR file, their DEBUG addresses, and the values required for the given effect:

Missiles	0178	04
Cargo Bay Extension	0179	01
ECM System	017A	01
Fuel Scoops	017D	01
Escape Capsule	017E	01
Energy Bomb	017F	01
Extra Energy Unit	0180	01
Docking Computer	0181	01

The DEBUG script for GIFT.CDR

```
e0100 43 6f 6d 6d 61 6e 64 65 72 20 66 69 6c 65 1a 0f
e0110 0d 43 4f 4d 4d 41 4e 44 45 52 20 00 20 20 20 20
e0120 20 20 33 30 30 2e 30 20 43 72 65 64 69 74 73 00
e0130 1a 18 74 d7 9b 27 01 4e 4d 4e 4d 50 40 4e 4d 00
e0140 43 45 52 45 58 45 00 20 00 06 f4 44 00 00 05 08
e0150 28 ff 07 04 04 00 28 50 19 43 45 52 45 58 45 20
e0160 20 00 06 f4 00 00 00 05 08 28 ff 07 04 04 00 28
e0170 50 19 b4 d4 d2 d1 00 ff 04 01 01 00 00 01 00 00
e0180 01 01 00 00 02 01 03 0f b8 0b 00 00 00 fe 47 49
e0190 46 54 00 43 44 52 00 00 04 00 0b 00 00 00 00 00
e01a0 0e 00 11 00 04 00 16 00 00 00 04 00 01 00 0c 00
e01b0 03 00 00 00 00 00 0b 00 0c 00 00 01 3b 00 00 00
e01c0 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 f4
n GIFT.CDR
r cx
d0
w
q
```

■ Enter this script into a text file called GIFT.TXT and use DEBUG to turn it into a free commander file, courtesy of reader Mike Davis

Galactic Hyperdrive 0182 01
Cloaking Device 0185 01
(This last item makes you invisible whenever you hold
down [N], but drains energy very quickly)

Gold 01B2
Platinum 01B4
Gem Stones 01B6
Alien items 01B8

4x Military lasers 017B 00
0184 04
0186 0F
0187 FF

10,000,000 credits 0188 00
0189 E1
018A F5
018B 05
011D 31
011E, 011F, 0120, 0121, 0122 30

RATING: enter the following at address 018D:

Mostly Harmless 02
Poor 04
Average 09
Above Average 14
Competent 23
Dangerous 5A
Deadly 9B
Elite FE 03

(Elite requires the change at 018E as well.)

CARGO: enter number of units in hex (up to FF) at:

Food 0198
Textiles 019A
Radioactives 019C
Slaves 019E
Liquor/Wines 01A0
Luxuries 01A2
Narcotics 01A4
Computers 01A6
Machinery 01A8
Alloys 01AA
Firearms 01AC
Furs 01AE
Minerals 01B0

These assume a virgin commander file, so start by loading *Elite* and immediately saving the start position as a new commander file (called HACKED.CDR, for example). Now load the commander into DEBUG using the command `debug hacked.cdr` (assuming you have copied DEBUG.EXE onto a copy of your *Elite* play disk).

This loads the file into memory from address 0100 (remember to subtract 100 hex from the values above if you are using a different file editor). You can look at its contents by entering `d100` to display the first 128 bytes, and just `d` to display the next. Edit the contents of an address with `e address data`, and save it to disk when you have finished by entering `w` followed by `q`.

Elite will only recognise commander files that have had the archive attribute set OFF. This can be achieved using the MS-DOS command ATTRIB (an external utility) by entering `attrib -a hacked.cdr`. If you do not have MS-DOS version 3.2 you may not have the ATTRIB function, you can reset the archive attribute using DEBUG.

To do this you first have to load the Directory sectors 5 through to 12 of your *Elite* disk into DEBUG. Assuming again that you have DEBUG.EXE on your *Elite* disk in drive A, you achieve this with the command `1 100 0 5 7` (change the 0 to 1 if the disk is in drive B). Search through the directory until you find the name HACKED.CDR, and change the next byte after the characters 'CDR' from 20 to 00. Now write this to disk with `w 100 0 5 7` and exit DEBUG with `q`.

These addresses may not be correct for all versions of *Elite* so if some of them don't work you'll have to experiment (don't call us!). And don't try this on your only copy of the game either – DEBUG is capable of causing considerable damage to a floppy disk if misused! ●

T H E
**HELP
SCREEN**
C O L L E C T I O N

121.

10

P R O G R A M M I N G

174. Machine Configuration

You can check the current status of a printer (if any), and find out (roughly) what display adaptor is fitted from within your programs by using MS-DOS interrupt calls. Interrupts are a means of 'low-level' communication between a program and the operating system, and they're normally used by programmers working in assembly language. *QuickBASIC*, gives you a access to interrupts with the INT86 command, described in the manual. Most other high-level languages provide other means of access.

An interrupt works by accepting parameter data in the 8086 processor's registers, performing whatever tasks are required, then returning replies in these

same registers. *QuickBASIC* doesn't give you direct access to the processor registers but instead accepts two arrays of integers, one representing the data you want placed in the registers before the interrupt and the other the reply values returned by MS-DOS.

Two values determine which MS-DOS function is performed; the interrupt number, and the value set in the AH register (i.e. the most significant eight bits of element zero of the input array in *QuickBASIC*).

To check the printer status set AH to 02 and call Interrupt 17 (Hex). The bits of the byte returned in AH have the following meaning when set on:

- 7 Printer busy
- 6 Printer acknowledge
- 5 Out of paper
- 4 Printer on-line
- 3 Printer not connected/unavailable
- 0 Printer timed out

Checking the display adaptor is slightly more complex. First, call interrupt 10 Hex with AH = 12 Hex and BL = 10 Hex. If, on return, BL is still equal to 10 Hex, then you don't have an EGA adaptor fitted.

Next, call interrupt 10 Hex with AH = 0F Hex. On return, AH contains the width of the screen in columns (40 or 80), and AL the screen display mode. If this value is seven, and you don't have an EGA fitted, then you have a monochrome display adaptor fitted; if it's not seven, then you have a CGA adaptor.

On the PC1512 there's no way of telling whether the monitor fitted is colour or monochrome as both types use exactly the same CGA-emulation mode. You can however check whether the program is running on a 1512 by using a special function which on the 1512 returns the version number of the ROM, but to an IBM PC looks like an illegal cassette interrupt. Call interrupt 15 Hex with AH = 06 Hex. If Carry is set it is not a 1512.

As a further check (This might after all be a legal call on some other clones), load BX with zero beforehand: a 1512 will return BX with the version number, a clone would probably leave it containing zero.

176. Checking Printer Status

Another way of checking the status of a printer from within a program is to read the status from the appropriate I/O port. Many high-level languages supply functions that can read and write to the processor's I/O ports, such as the BASIC functions INP and OUT.

The PC architecture supports three printers in this way at port addresses 3BDh, 379h and 279h. The first corresponds to the printer port fitted to the old MDA display adaptor, The second is usually the one

122.

175. Fast Sort Routine

One of the fastest string sorting routines around is called the Shell-Metzner sort. The following is a BASIC listing which first creates 2500 strings filled with random numbers, sorts them into numeric order, and then prints out how long the sort took. It may require a little modification but should run fine under *GW BASIC*, *Turbo BASIC* or *QuickBASIC*:

```

10 ' Sorts 2500 strings filled with random numbers
20 RANDOMIZE TIMER
30 MAX% = 2500
40 CLS:PRINT "2500 strings being created"
50 DIM D$(MAX%)
60 FOR I = 1 TO MAX%
70   D$(I)=""   ":RSET D$(I) = STR$(INT(100000!*RND))
80 NEXT
90 PRINT "Strings created, starting sort"
100 BEEP:START! = TIMER
110 GOSUB 200:FINISH! = TIMER:BEEP
120 FOR I = 1 TO MAX%
130   PRINT D$(I);
140 NEXT:PRINT
150 PRINT "Time taken:"
160 PRINT FINISH!-START!
170 END
180 ' MAX% = Number of items to sort
190 ' D$(max%) = string array to be sorted
200 Z = MAX%
210 Z = INT(Z/2)
220 IF Z = 0 THEN RETURN
230 Y = MAX%-Z:ZZ = 1
240 X = ZZ
250 XX = X+Z
260 IF D$(X) <= D$(XX) THEN 290
270 SWAP D$(X),D$(XX):X = X-Z
280 IF X >= 1 THEN 250
290 ZZ = ZZ+1
300 IF ZZ <= Y THEN 240
310 GOTO 210

```


corresponding to the parallel printer port fitted to the back of most modern compatibles, although you might need to experiment. In *GWBasic* or *QuickBasic* you would read the port by examining the byte returned with `INP(&H379)` (or whatever). The meaning of each of the bits is the same as that returned by BIOS interrupt 17H Function 02H detailed above.

To reset the first printer port, simply use:

```
OUT &H3BE, 0
```

```
OUT &H3BE, &HEC
```

Again, experimentation might be necessary.

177. Programming the 8086

The 8086 is a little unusual in its design, but it and MS-DOS are wonderful things once you're in control, and surprisingly friendly too. Mastering them is a doddle once you've got a few basic concepts under your belt.

The 8086's vital pointer registers are all 16 bits long. The stack pointer SP is, the base registers BX and BP are, and so are the index registers SI and DI. Even the instruction pointer – the register that points at the next instruction to be executed – is 16 bits long. In other words, each of these registers can point at one of 65,536 locations (that's how many different 16-bit

numbers there are) or 64K of memory.

But if the 8086 could only keep track of instructions or access data within a 64K range, that would leave something like 90 percent of your micro's memory unusable. Clearly this isn't the case. The 8086 can get at the whole of the PC's RAM, but the way it manages this is a little unconventional.

On their own the pointer registers may only span a 64K block of RAM, but that 64K can lie anywhere in the full 640K (or whatever) of your machine's memory. A quite separate set of pointers – held in the Segment Registers – tell the 8086 where these 64K blocks start. (The blocks are properly known as Segments, hence the name.)

There are four segment registers: CS, DS, ES and SS. The segments they point to have different uses. CS points to the Code Segment, the area from which program instructions are fetched. DS points to the Data Segment, the area where program variables are kept. ES points to the Extra Segment – normally used only for certain string operations – while SS points to the Stack Segment.

The segment registers don't act like normal pointers. A pointer contains the address of the location it points to, but that wouldn't work for the segment

178. User-defined Characters

This subroutine, called DEFCHAR and sent in by Paul Mitchell of Aberdeen, should work on any PC compatible using *GWBasic* (not Locomotive's *BASIC2*). It allows users to define their own characters when using either the medium or high-resolution graphics modes (SCREEN 1 and 2). It is necessary, however, to have *GRAFTABL* loaded before it is used.

Normally, the graphics characters from 128 to 255 are undefined and cannot be used. *GRAFTABL* defines these as their proper text-mode characters. DEFCHAR then allows any of these characters to be altered by writing to the definition memory allocated by *GRAFTABL*.

To define a character, DATA lines must be inserted into the program. The first value on a line must be the character to be defined. The following eight values are the actual definition. The first number defines the top line and so on. Each line of the character will be defined as the binary equivalent of the value given. For example:

```
7 6 5 4 3 2 1 0    Value
X X X X X X X X = 1111 1111 = 255
X                  X = 1000 0001 = 129
X                  X = 1000 0001 = 129
X  X X X X        X = 1011 1101 = 189
X  X X X X        X = 1011 1101 = 189
X                  X = 1000 0001 = 129
X                  X = 1000 0001 = 129
X X X X X X X    = 0111 1110 = 126
```

To define the above symbol as character 175, the following line is used:

```
DATA 175, 255, 129, 129, 189, 189, 129, 129, 126
```

Any amount of data lines may be used and the subroutine will stop when a character value of -1 is encountered. Remember to RESTORE to the proper data line before calling the DEFCHAR subroutine.

This subroutine can also be used with compilers like *QuickBasic* assuming that *GRAFTABL* is loaded before the compiled programs are executed.

```
00000 DEF SEG = 0
10010 TabSeg% = PEEK(&H7E) + 256 * PEEK(&H7F)
10020 TabAdr% = PEEK(&H7C) + 256 * PEEK(&H7D)
10030 DEF SEG = TabSeg%
10040 READ Char% : IF Char% = -1 THEN RETURN
10050 IF Char% < 128 OR Char% > 255 THEN PRINT "Out of range!"
   : RETURN
10060 CharAdr% = TabAdr% + 8 * (Char% - 128)
10070 FOR Byte% = 0 TO 7
10080   READ CharLine%
10090   IF CharLine% < 0 OR CharLine% > 255 THEN PRINT
      "Pattern value"; Byte%; "out of range!" : CharLine% =
      CharLine% AND 255
10100   POKE CharAdr% + Byte%, CharLine%
10110 NEXT Byte%
10120 GOTO 10040
20000 DATA 255,255,129,129,189,189,129,129,255
20010 DATA -1
```

registers. They're only 16 bits long, so they can't contain addresses outside the first 64K of memory. It's the same problem as before: a 16-bit number (four hex digits) can't describe every location in the PC's memory. The full PC memory map, ROMs and all, spans 1024K – and it would take a 20-bit pointer (five hex digits) to cover that many locations completely. To get round this, the 8086 uses the contents of a segment register as the first four hex digits of a five digit number, the fifth (or units) digit being zero.

For example, if the CS register contains 2FE4 hex then the code segment starts at address 2FE40. Similarly, if the DS register contains 4800 hex then the data segment starts at address 48000. As you can see, the start address of a segment in hex always ends zero. This doesn't really create problems, because the precise starting point of a segment isn't important. The segment just puts the 8086 in the ballpark: provided the actual address you're after is somewhere in the 64K after the segment start, you'll be able to access it.

The 8086's pointers – BX, SI and all that lot – only do half the job of pointing to a location: they tell the 8086 where the location is relative to the start of the

segment. This relative position is called the Offset Address. To find the absolute position of the location the 8086 has to add the offset to the segment's starting address, or Segment Address for short.

Because there are four segment registers, the 8086 can handle four separate 64K segments at once. They can all start in the same place, all start in different places but overlap, or all be entirely separate: it's up to you. You might think you'd be best off having separate segments, or running them one after the other to form a continuous 256K block of accessible RAM, but in practice that's rarely the way you'll want things.

Typically, hobbyists and semi-serious programmers don't need to work with enormous amounts of memory. Filling a segment up with data is difficult, and isn't normally necessary when you've got a reliable operating system like MS-DOS at hand. Writing 64K of program code is harder still, and filling up a 64K stack is normally a sign of a serious bug.

The best course of action for most people, most of the time, is to have all four segments starting in the same place. This limits you to a total of 64K for program, data and workspace but that's normally ample. There'll be times when you'll want to access other areas of memory, particularly if you're writing a game or art-tool that writes directly into screen memory, but the rest of the time sticking to a single segment position makes life an awful lot easier. In particular it means you can write your programs as COM files.

So much for the basic concepts. From here on it's largely a matter of getting a decent MS-DOS programmer's guide – Ray Duncan's *Advanced MS-DOS* is particularly good – an 8086/88 reference and sticking to a few commonsense rules:

1. If your assembler wants loads of directives and you don't know which ones, examine any sample source files that come with it. If the worst comes to the worst, find a simple one and try editing it a little.
2. Shop around for assemblers. Do try getting hold of a few Public Domain or Shareware ones before shelling out for an expensive commercial effort.
3. Don't turn your nose up at DEBUG: it can be the quickest way to get things done, and it's absolutely free.
4. Don't get bogged down in all those different addressing modes. Just use simple base or index addressing – [BX], say – and then alter the value of the pointer to get at different bits of data. There'll be time enough for more exotic stuff once you've found your feet.
5. Stick to COM files wherever possible. If your

179. Hexadecimal Converter

Hexadecimal place					Conversion	
5th	4th	3rd	2nd	1st	Hex	Binary
0	0	0	0	0	0	0000
65,536	4,096	256	16	1	1	0001
131,072	8,192	512	32	2	2	0010
196,608	12,288	768	48	3	3	0011
262,144	16,384	1,024	64	4	4	0100
327,680	20,480	1,280	80	5	5	0101
393,216	24,576	1,536	96	6	6	0110
458,752	28,672	1,792	112	7	7	0111
524,288	32,768	2,048	128	8	8	1000
589,824	36,864	2,304	144	9	9	1001
655,360	40,960	2,560	160	10	A	1010
720,896	45,056	2,816	176	11	B	1011
786,432	49,152	3,072	192	12	C	1100
851,968	53,248	3,328	208	13	D	1101
917,504	57,344	3,584	224	14	E	1110
983,040	61,440	3,840	240	15	F	1111

This table is particularly useful for converting large hexadecimal numbers to decimal or binary. The first five columns give the decimal equivalent of each place in the hexadecimal number so, for example, the hex number FCBA converts to:

61,440+3,072+128+10 = 64,650 in decimal, or
1111 1100 1000 1010 in binary.

program won't fit into 64K you're probably doing something wrong.

180. Rounding Errors

Try the following program on many BASIC interpreters or compilers:

```
FOR i=0 TO 1 STEP .05
  PRINT i
NEXT i
```

And you will get the following output:

```
0      0.5
0.05   0.55
0.1    0.6
0.15   0.65
0.2    0.700000001
0.25   0.750000001
0.3    0.800000001
0.35   0.850000001
0.4    0.900000001
0.45   0.950000001
```

Changing the STEP only changes the point at which the loop becomes 'wild', and the range of the loop has no effect, so 1 to 10 gives similar results. The loop terminates before *i* reaches 1 the, because the next *i* will be greater than 1.

Add the following line before the PRINT statement, and the result will appear even more inaccurate:

```
i=INT(i*100)/100
```

This is not a bug, but simply a side effect of using binary arithmetic for floating point numbers. If you convert a decimal fraction (such as 0.05) to binary, you will of necessity get an inexact representation. If you then add together these inexactly represented fractions, any error is magnified. This is shown by the later results in the table of added 0.05's. Any programming language that uses floating point binary arithmetic would generate similar results. It is only with binary-coded decimal floating point that exact results can be obtained.

One solution is to round *i* to two decimal places, as in:

```
FOR i=0 TO 10 STEP 0.05
  i=ROUND(i,2)
  PRINT i
NEXT i
```

181. Automatic Loading Trick

The following enables a program written with the BASIC2 interpreter to run directly from the MS-DOS prompt, and the basic technique would work with many other interpreters too. Name your program MYPROG.BAT and start it with the following lines:

```
GOTO begin
:begin
CD C:\
GEM BASIC2 \BASIC2\MYPROG.BAT
LABEL begin
REM Rest of program code
```

What this does is to combine BASIC2 commands with standard MS-DOS batch commands, so the first line, GOTO begin can be interpreted in two ways. Run from the MS-DOS prompt it is interpreted as a batch file command, so execution jumps to the next line, the :begin label, from where the directory changes to the root and BASIC2 is automatically run.

BASIC2 proceeds to run the same file as a BASIC program, interpreting the first line GOTO begin as a jump to LABEL begin, where the rest of the program code is to be found.

As it stands the program is designed to run from a hard disk, however it could easily be adapted for two floppies or even a single floppy system. There is nothing to stop this method being used to copy the program into a RAM disk, and then use PAUSE to wait for the disk with GEM BASIC2 to be placed in the drive.

183. COM versus EXE

Leaving aside batch files, there are two different kinds of program you can run under MS-DOS. Simple *binary* files always have the extension COM, while the more sophisticated *executable* files have the extension EXE.

When MS-DOS runs a binary file it puts the same value in all the segment registers, loads the contents of

182. Hexadecimal Addition

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

the file in at an offset of 0100 hex and then jumps there. It follows that a COM file program can only have one segment, and so must be less than 64K in length. This is unlikely to be a problem unless you're writing an enormous application. On the plus side, the binary file's simple structure means you can easily make your own by saving a chunk of memory from DEBUG. So long as you assemble the code starting at 0100 hex and make that the start point of your save, the resulting file will do fine as a COM program.

If the length and segment restrictions of a COM file prove unacceptable you'll have to write your program as an EXE file, and things get tougher. Executable programs contain not only your program but also information on how to relocate the separate segments at load time. Putting an EXE file together by hand isn't practical, and even assemblers normally can only do half the job. You'll need to feed the assembler output through a linker to get a file you can run.

All COM files could be written as EXE files, but the reverse clearly isn't true. That said, most simple EXE files can be turned into COM files using the EXE2BIN utility on the MS-DOS system disk. If you use the Microsoft assembler *MASM*, you'll find this is the only way to get a COM file out of it. It's a galling business: you have to assemble, link and EXE2BIN your program to create even the simplest of binary files.

184. Integer Overflow

Some programming languages can appear to suffer from inaccurate mathematics, giving results such as:

$182 * 182 = -32412$

$1000 * 100 = -31072$

$200 * 200 = -25536$

$2000 * 2000 = 2304$

This is a symptom of 'integer overflow', a common phenomenon in microcomputer languages caused by a calculation resulting in a number too big to fit into the amount of RAM reserved to hold it.

Most languages store numbers in binary format, that is strings of 1's and 0's. Each byte of RAM contains eight binary digits (otherwise known as bits) so a number held in one byte can contain values in the range of binary 00000000 to 11111111, which equals 0 to 255 in decimal.

Pascal, for example, recognises two main classes of numbers – real and integer. Real numbers typically occupy five or six bytes each, and can consequently hold large numbers. They can also include fractions, e.g. 2.595, and can be used to perform floating-point arithmetic, e.g. $2.595 * 1.1211$.

However, the full five or six bytes are used to store a real number, even if the value is very small (say two or three), and complex floating-point calculation routines are invoked every time real numbers are

manipulated, so reals carry large overheads of memory and processor time.

As an alternative you can use integer variables. These typically take two bytes of memory, and can only contain whole numbers. They occupy less memory, and use much simpler calculation routines which run faster, but you are limited in the range of values you can store in them.

Two bytes (or 16 bits) of binary data can contain numbers with decimal equivalent of 0 to 65535, but this doesn't take into account negative numbers. To accommodate this Pascal uses the left-most bit of the number as a 'sign bit'. If the bit is 0 then the number is positive, if it's 1 the number is negative.

This caters for negative values, but it reduces the range of numbers which can be held in the two bytes of storage by a whopping 32768, which is the value of the most significant digit of a 16-bit binary number. This is because the sign bit is effectively no longer part of the number itself, but a 'flag'.

For this reason 16-bit signed integers can only hold values in the range – 32768 to +32767 (not – 32768 because of 'two's complement' notation, but that's another story). If you perform a calculation which generates a number outside of this range, then the result 'overflows'.

In particular, if the result is greater than +32767 then the most significant bit gets set, and Pascal thinks it is a negative number.

For this reason integers should only be used in cases where the range is strictly controlled, for example as an array subscript, or a loop counter. There are in fact plenty of useful applications – if you're working on a 600 X 350 point graphics display, then integers are the right type of variable to hold the XY coordinates of the current cursor, and if you're writing a trivia game with 1,000 questions, then an integer can always hold the current question number.

Some Pascal compilers offer the option of long integers (32 bits), which gets round this limitation.

185. AT Scan Codes

On an AT or PS/2 keyboard the function keys [F11] and [F12] generate the scan codes 57 and 58 (hex) respectively.

186. Using Bitwise Instructions

In a graphics display mode, individual bits within the bytes of display adaptor RAM are translated directly into brightness and colour values for screen pixels. In the simplest case of a display mode with just one foreground colour, each bit defines whether its corresponding pixel is lit (foreground colour) or unlit (background colour). Thus to change the status of just

one pixel without altering any of the surrounding ones you need to alter just one bit of the appropriate byte in RAM.

This can be done using those fundamental tools of assembler programming, the bitwise instructions **AND**, **OR** and **XOR**. Each has a different effect, but the principle is the same for all.

In a bitwise operation a 'source' word is matched, bit-for-bit, with a 'mask' word and a third 'result' word is produced. Using just four bits, let's take the **AND** instruction as an example:

```
Source    0101
Mask      0011
Result    0001
```

The result has a bit set to 1 only if the corresponding bits on both the source and the mask words are set to 1. For an **OR** instruction, the result would be:

```
Source    0101
Mask      0011
Result    0111
```

In this case, if a bit in either the source or the mask is set, the corresponding bit of the result is set. Finally, the more complex **XOR** (eXclusive **OR**) instruction:

```
Source    0101
Mask      0011
Result    0110
```

Here a bit is set if wither the source or the mask bit is set, but not if both are set.

Using these three instructions you can set, unset and reverse any number of individual bits in a word without disturbing the others, as follows:

Set bits on – make a mask that has the chosen bits set to 1 and **OR** with the source.

Set bits off – make a mask with the chosen bits set to 0, **AND** with the source.

Reverse bit values – make a mask with the chosen bits set to 1, **XOR** with source.

The procedure for altering individual pixel-mapped bytes within a word is:

1. Read current contents of the appropriate word from the display RAM.
2. Apply bitwise instruction with the appropriate mask.
3. Write result back to the display RAM.

187. Leaving GW-BASIC

Although the *GW-BASIC* manual doesn't make this at all clear, the **SYSTEM** command works perfectly well from within a program. In other words, a piece of *BASIC* like this:

```
1010 INPUT "Exit to DOS? (Y/N)";a$
1020 IF a$="Y" THEN SYSTEM
```

works fine. You could even combine **SYSTEM** with **KEY** and **ON KEY GOSUB** to trap a [Ctrl] [Break] or

[Ctrl] [C] if you wanted, so that aborting a program would also drop you back into *MS-DOS*. That way, if *GW-BASIC* and your program were loaded direct from a batch file, users would never have to deal with the *BASIC* side of things at all.

188. Random Numbers

There is a *DOS* function which can be used to give you a random number. If you request the system time you get hours, minutes, seconds and hundredths of a second, although the clock interrupt is not fast enough to really give hundredths. The algorithm used to supply this value does give an even distribution of values from 0 to 99, so it could be used as a random number so long as the request is fairly random, e.g. user interaction.

Here is a *C* program to check the value's randomness. You need to run it a few times to see valid results.

```
/* Random number generator (c) Colin Walls 1988 */
#include <dos.h>
int n [100];
main()
{
    union REGS regs;
    int i;
    for (i=0; i<10000; i++)
    {
        regs.h.ah = 0x2c;
        int 86 (0x21, &regs, &regs);
        n[regs.h.dl]++;
    }
    for (i=0; i<100; i++)
        printf('%4d', n[i]);
}
```

Here is another program which prints the result and returns it as an error value. You could lose the printed result by re-directing it to the nul driver (with **>NUL** on the command line).

```
/*Random number generator (c) Colin Walls 1988 */
#include <dos.h>
main ()
{
    regs.h.ah=0x2c;
    int86(0x21, &regs, *regs);
    printf('%d', regs.h.dl);
    exit(regs.h.dl);
}
```

189. Reading GW-BASIC Code

By default, *GW-BASIC* stores programs in a tokenised form not readable by text editors, or other BASIC interpreters. However saving the program from *GW-BASIC* using the command **SAVE filename,A** saves it as an ASCII file, which can then be loaded into another editor. What works for *GW-BASIC* should work for IBM *BASIC* too.

190. Reading the Mouse

As Microsoft created both *QuickBASIC* and the INT 33h mouse interface standard, it seems surprising that *QuickBASIC* doesn't provide proper mouse functions. Still, P A Thyer of Barnet came up with the program here that uses the language's CALL INTERRUPT feature.

To use CALL INTERRUPT the default library must be loaded, which is done by loading *QuickBASIC* from MS-DOS with **QB /L**. The line **'\$INCLUDE: 'QB:BI'** in the program includes a file that defines the type **RegType**, used to pass values to and from the microprocessor's registers. For instance **inreg.ax=3** passes 3 to the AX register before the interrupt, and **outreg.bx** contains the value left in the BX register after the interrupt.

CentreMouse sets the position of the mouse at the beginning of the program: note that it treats the screen

as being 640 pixels wide even in medium resolution modes.

MouseCursorOn is needed only if the position of the mouse is to be shown by a pointer on the screen.

GetMouse updates the variables which contain the position of the mouse and the state of the mouse buttons.

The main program contains the **WHILE...WEND** loop which just prints out on screen the co-ordinates of the mouse and the state of its buttons. It can be replaced by a program to do whatever the user wants, using the **SUBS** to read the mouse.

191. Recommended reading

For a relatively easy to read introduction to the subject, try *Assembly Language Programming for the IBM PC* by David J. Bradley, published by Prentice Hall on (0442) 58531.

From Microsoft Press (on 01-938 2200) comes the *Programmer's Guide to PC & PS/2 Video Systems* by Richard Wilton at £22.95, and probably the clearest description of the innermost workings of DOS available in *Advanced MS-DOS Programming* by Ray Duncan, also £22.95. Also by Ray Duncan are the somewhat cheaper slim pocket guides *MS-DOS Functions* and *IBM ROM BIOS*, both in the Microsoft Press Quick Reference Series at £4.95 each. The ultimate reference is perhaps *The Programmer's PC*

128.

'Program to demonstrate use of mouse from QuickBasic

DECLARE SUB CentreMouse (InitialX%, InitialY%)

DECLARE SUB GetMouse ()

DECLARE SUB MouseCursorOn ()

'\$INCLUDE: 'QB.BI'

COMMON SHARED inreg AS RegType

COMMON SHARED outreg AS RegType

COMMON SHARED MouseX

COMMON SHARED MouseY

COMMON SHARED MouseLeft

COMMON SHARED MouseRight

SCREEN 9

COLOR 14, 1

CentreMouse 320,175

WHILE INKEY\$ = ""

 MouseCursorOn

 GetMouse

 LOCATE 1, 1

PRINT "MouseX="; MouseX, "MouseY="; MouseY,
PRINT "MouseLeft="; MouseLeft, "MouseRight="; MouseRight
WEND

END

SUB CentreMouse (InitialX%, InitialY%)

 inreg.ax = 4: inreg.cx = InitialX%: inreg.dx = InitialY%

 CALL INTERRUPT(&H33, inreg, outreg)

END SUB

SUB GetMouse

 inreg.ax = &H3

 CALL INTERRUPT(&H33, inreg, outreg)

 MouseX = outreg.cx

 MouseY = outreg.dx

 MouseLeft = outreg.bx AND 1

 MouseRight = (outreg.bx AND 2) / 2

END SUB

SUB MouseCursorOn

 inreg.ax = 1

 CALL INTERRUPT(&H33, inreg, outreg)

END SUB

Sourcebook by Thom Hogan at £22.95, providing 500 pages of tables listing function calls, addresses and reply codes for MS-DOS, Windows and IBM PC hardware. Or for an excellent introduction to the workings of the PC, as well as detailed descriptions of BIOS and MS-DOS interrupts, there is *The New Peter Norton Programmer's Guide to the IBM PC & PS/2* by Peter Norton and Richard Wilton, at £19.95.

For a general introduction to programming techniques try *The Art of Computer Programming* by Donald Knuth. *Professional Pascal – Essays on the Practice of Programming*, by Henry Ledgard, presents the case for structured programming, and is not just for Pascal fans. It is published by Addison Wesley on (0734) 794000.

Required reading for C programmers is *The C Programming Language – 2nd Edition* by Brian Kernighan and Dennis Ritchie, published by Prentice-Hall. A clear and entertaining guide by the authors of the language, so they should know what they are talking about!

As far as assembly language books go, unfortunately there isn't one ideal book. *The 8086 Book* by Rector & Alexy (published by Osborne/McGraw-Hill) has the odd error, but gets the job done. For assemblers the best bet is the shareware assembler *A86*, available from most User Groups.

192. Using the Command Line

The ability to access the command line – the string of characters that has been entered by the user at the MS-DOS prompt – is extremely useful as it allows users to pass parameters to your programs with ease. The MS-DOS batch file language manages this quite easily by allocating ten 'replaceable variables', labelled %1 through to %9, to the first ten arguments of the command line, with the 'space' character determining where one argument finishes and the next one starts. These variables can be put to good use in batch files.

For example, the following file, called CLEAN.BAT, could be useful for cleaning up the contents of a disk:

```
ECHO OFF
del a:*.1.*
```

Entering `clean letter` at the MS-DOS prompt would result in the file LETTER.TXT, LETTER.BAK, LETTER.WKS and so forth being automatically deleted from the floppy disk in the A drive. This is a trivial example: for more details see the 'Batch Files' chapter of this book.

The situation becomes more complex when you are running a compiled program from a COM or EXE file, but thankfully most high-level languages provide some means of accessing the command line.

Borland's *Turbo Pascal* provides the `ParamCount` and `ParamStr` functions. `ParamCount` returns the

number of arguments in the command line, ignoring the first (the name of the program itself). `ParamStr` is an array with `ParamStr(1)` returning the first argument as a string, `ParamStr(2)` the second, and so on.

For example, the command line:

```
crunch file1 file2
```

would result in `ParamCount` returning the value 2, `ParamStr(1)` returning the string `file1` and `ParamStr(2)` returning the string `file2`.

The C programming language provides similar facilities, but counts the first argument – the program name itself – in with the rest. `argc` returns the total number of arguments in the command line as an integer, and in the above example would return the value 3. `argv` is a pointer to an array of character strings that contain the arguments themselves, so `argv[0]` would point to a string containing `crunch`, `argv[1]` would point to the string `file1` and `argv[2]` would point to the string `file2`. They would usually be declared as follows:

```
main(argc, argv)
int argc;
chr *argv[];
{
    /* program code */
}
```

Microsoft's *QuickBASIC* is not so helpful. It provides the function `COMMAND$` which strips the program name and any subsequent spaces from the command line, converts all remaining letters to upper case, and returns it as a single text string. It is up to the programmer to extract the arguments from this string, using the `INSTR`, `MID$` and `LEFT$` functions for manipulating strings of text.

The `INSTR` function can be particularly handy if you know the next argument in the command line should be, for example, one of four distinct three-letter commands, say DAT, MOV, SUB or ADD. If the argument has been extracted as the three-letter string `parm$`, the following returns the integer `code%` with the value 1,2,3 or 4 depending which command has been selected:

```
code% = INSTR("DAT|MOV|SUB|ADD", parm$)
IF code% = 0 THEN
    'error routine
END IF
code% = (code%-1)/4
```

Here the string `parm$` is compared with a string carefully constructed, using the `|` character although any unusual character would do, to only match on the required three letter sequences. It would, of course, also match if an argument `T|M`, but this is an unlikely combination and you could always trap a string containing the `|` character in the following IF

statement. A useful method of avoiding lengthy CASE or IF..THEN..ELSE statements.

This is all very well if you have a language that allows the extraction of the command line. If you don't then you will have to resort to directly interrogating the memory.

When a program is loaded by MS-DOS a 256-byte block of memory is set aside for what is called the Program Segment Prefix (PSP). This is located just below the program itself, at the segment address held in registers DS and ES when control is transferred to the program. Alternatively DOS function 62H (decimal 98) returns the segment address of the PSP in register BX (provided you are using MS-DOS version 3.0 or later).

The PSP contains a whole hodge podge of information, but for our purposes the interesting bit is the field starting at offset 80H (decimal 128). The first byte, at 80H, contains the length of the command line, with the string itself starting at 81H. This starts at the first character after the program name, which is usually a space, and is otherwise unchanged except that any redirection symbols, the < and >, are stripped out (this is why a program cannot know if its input or output has been redirected, or indeed its own name!). The only thing to watch for is that this field is also used as the Disk Transfer Area unless another area has been specified.



193. Screen Dumps

To get a screen dump from *GW-BASIC*, try the following. It simply invokes the system screen dump routine, so it should still work with *GRAPHICS*:

```
1000 REM Dumps screen to printer
1010 DIM a(10)
1020 b=0:b=VARPTR(a(0))
1030 DEF SEG
1040 POKE b,&HCD
1050 POKE b+1,&H5
1060 POKE b+2,&HCB
1070 CALL b
1080 ERASE a
1090 RETURN
```

Because of the ERASE you can call this routine repeatedly without any need for preparation the first time through. If you normally run your program with the function key definitions displayed along the bottom of the screen, you'd be strongly advised to add these lines unless you want the definitions in the screen dump too:

```
1065 KEY OFF
1075 KEY ON
```

This program simply creates a small amount of machine code to execute system interrupt 5 (the BIOS print screen interrupt). If you are using a more

sophisticated version of BASIC you may be able to execute this interrupt directly. *Turbo Basic*, for example, interprets CALL differently to *GW-Basic* so this program won't work. All *Turbo Basic* requires is the single statement:

CALL INTERRUPT 5

Microsoft's *QuickBASIC* provides CALL INT86 for directly calling interrupt routines – see the manual for further details.

194. Day of the Week

There are many ways of finding out what day of the week a particular date falls on, usually by first converting the date into a Julian day number – an astronomical device for turning dates after 1583 into a single integer – and using the MOD operator to convert this into a number from one to seven. There is, however, an easier way although purists might call this is cheating.

Enter **date** at the MS-DOS prompt and you will find that your computer (assuming it has a battery-backed clock) already knows not only the date but the day of the week as well. MS-DOS has already worked it out, and provides the MS-DOS function 2AH (decimal 42) so you can work it out too. This returns the month as 1 through to 12 in register DH; the day as 1 through to 31 in DL; the year in CX and the day of the week as 0 through to 6 in register AL (0 represents Sunday and 6 represents Saturday). The function 2BH (decimal 43) is used to set the date, so the two can be used together to find the day of the week on which any date falls.

195. More on COM and EXE files

Here's a little curiosity. Take a COM program file – PROGRAM.COM say – and change its name to PROGRAM.EXE. Now run it as normal, by typing PROGRAM at the MS-DOS prompt, and it works fine. The same goes if you rename an EXE file as PROGRAM.COM. How come? An EXE file is nothing like a COM file internally needs completely different treatment to load it properly.

Presumably this is a rare example of DOS being idiot-proof, but how does it tell it's looking at a COM file despite the EXE extension (or vice versa)? One way would be to check the first two bytes of the file. All EXE files start with the characters **MZ** – hex 4D 5A – which serve no apparent purpose. These translate into a pair of machine code instructions nobody would dream of starting a COM file with, **DEC BP** and **POP DX**, so that looks like a promising method.

Needless to say, things are a good deal more complex than that. Simply starting a program with **MZ** is not enough to convince DOS it's looking at an EXE file. Presumably the whole EXE header must be correct.

Since the header contains a checksum and file length information, which MS-DOS can check independently, the chances of accidentally producing an EXE file are effectively nil. We've only tested this above on MS-DOS version 3.2, so it's possible that earlier versions did actually use the COM/EXE extension. Who knows?

Another difference between COM and EXE program files is that an EXE file can specify how much working space it absolutely must have. If MS-DOS finds there isn't that much memory free, it doesn't waste its time loading the program but just returns an 'Insufficient Memory' message instead. A COM program, on the other hand, will always load if there's space to fit, and has to check for itself to see if there's enough working space left over. Where does the EXE space information live? Why, in the file header of course.

A COM file is simply a block of code and data assembled to run at memory address 0100h. There is no header information of any kind: the first byte of the file is the start of the program itself. In other words, it is a straight binary image of memory as it should be when the program runs.

Because of the way the PC's 8086 central processor is designed, programs less than 64K can be written so that they're relocatable – so that they can load and run anywhere in memory – while programs over 64K in length can't be. However, an MS-DOS

program has to be capable of running anywhere in memory, because there's no knowing where MS-DOS will want to load it. The upshot is that any program over 64K long has to be relocated by the operating system – altered to run properly in its new location, in other words – before it can run.

Unfortunately, to relocate a program an operating system needs more information than just the code of the program itself. It really needs a complete list of every byte it'll have to alter. Since a COM file doesn't come with such a list, it can't be relocated, which means it has to be less than 64K long.

An EXE file, on the other hand, does come with such a list, contained in the 512 or more bytes that make up its EXE header. This means EXE files can be as large as available memory or disk space will allow. There's also other information in the EXE header that makes EXE files a desirable thing for future-proofing or 'forward compatibility' reasons, but this is of little concern to programmers at the moment.

On the minus side that header makes an EXE file a complex thing to put together, which is why many compilers, assemblers and the like can't produce EXE files in one go, but have to settle for OBJ ('object') files instead. You then feed the OBJ file through a linker program, such as the LINK.EXE utility provided with most MS-DOS systems, and this produces your finished EXE file. As well as creating the EXE file

196. Binary from C

There's no standard C way of displaying numbers in binary form, but it's easy enough to knock something together yourself. The example here creates the function `binary()` to tackle the problem, which takes an integer parameter (signed or unsigned) and returns the address of the global string `binstr[]`.

The real meat of the function is the `while` loop. It does a bitwise AND (the `&` operator) with the 1-bit-wide `mask` to check the state of each bit in `num`. Then it uses the conditional `a ? b : c` format to set the digits of `binstr[]` accordingly. The line `mask >>= 1` is a right-shift assignment: it means 'shift `mask` one bit to the right', or 'halve it and discard any remainder' depending on how you look at it. That final `binstr[subscr]='0'` writes a null character (ASCII code zero) on the end of `binstr` so that `printf()` can handle it properly:

```
#include <stdio.h>
#include <stdlib.h>
```

```
char *binary();
char binstr[17]="";
```

```
main()
{
    int num;
    printf("Enter a number in decimal:\t");
```

```
scanf("%d", &num);
printf("That is %x in hexadecimal\n",num);
printf("That is %o in octal\n",num);
printf("That is %s in binary\n",binary(num));
}
```

```
binary(num)
int num;
{
    unsigned mask = 0x8000;
    int subscr = 0;
    while(mask!=0)
    {
        binstr[subscr++] = (mask & num)!=mask ? '1' : '0';
        /* Sets current binstr[] character to '1' or '0'
           depending on whether corresponding bit of num
           is set or not. Moves to next character. */
        mask >>= 1;
        /* Moves to next bit of num */
    }
    binstr[subscr] = '\0';
    return binstr;
}
```


197. Using Amstrad's 16-Colour Mode

Dr Alan Mynett of West Sussex supplied the following routines, written in *Turbo Pascal*, that make full use of the Amstrad PC1512's unique 16-colour graphics. The routines initialise the 16-colour mode and provide point plotting (PSET), line drawing (DRAWLINE), circle drawing (DRAWCIRCLE) and box drawing procedures (DRAWBOX, FILLBOX). In addition COLOUR can be used for coloured text.

Program HiresGraphics_Amstrad_PC1512;

```
{ $U+ }

{
  EGA1512.inc *****

  routines for using the 16 Colour mode on the Amstrad PC1512

  *****
}
```

Const

```
  BitValue : array[0..3] of byte = (1,2,4,8);
                                     {weight bits 1-4}
```

procedure Amstrad_16_Colour_Graphics (Paper, Border : byte);

```
{
  Initiates the Amstrad PC 1512 16-Colour mode with a
  border and paper colour by writing to the necessary ports.
}
```

begin

```
  Hires;           {select 640 x 200 display}
```

```
  Port[$3D9] := $0F; {enable 16 colour mode }
```

```
  Port[$3DF] := Border; {set border colour }
```

```
  Port[$3DD] := Paper; {sets which planes to be written to}
```

```
  FillChar (Mem[$B800:0], $4000, $FF)
                                     {sets screen to Paper Colour}
```

end;

Procedure Colour (Ink : byte);

```
{ Sets which colour planes to write to }
```

begin

```
  Port[$3DD] := Ink
```

end;

Procedure Pset (x,y,colour : integer);

```
{
  Sets a given pixel to the specified colour.
  Examines each colour plane in turn and uses the
  Turbo Pascal Plot statement to set or reset
  the relevant pixel in the plane.
}
```

var

```
  Plane : integer;
```

begin

```
  for plane := 0 to 3 do {each plane in turn}
```

```
  begin
```

```
    Port[$3DD] := BitValue[Plane];
```

```
    Port[$3DE] := Plane;
```

```
    if (colour AND BitValue[plane]) <> 0 then
```

```
      Plot(x,y,1)
```

```
    else
```

```
      Plot(x,y,0)
```

```
    end
```

end;

Procedure DrawLine(x1,y1,x2,y2,Colour : Integer);

```
{
  Draws a line in a given colour.
  Examines each plane in turn and uses the Turbo
  Pascal Draw statement to draw the line by
  setting bits in the relevant planes and resetting
  the bits in the other planes
}
```

Var

```
  Plane : integer;
```

begin

```
  for plane := 0 to 3 do {each plane in turn}
```

```
  begin
```

```
    Port[$3DD] := BitValue[Plane];
```

```
                                     {set the plane to write to }
```

```
    Port[$3DE] := Plane;
```

```
                                     {and read from }
```

```
    if (colour AND BitValue[plane]) <> 0 then
```

```
      Draw(x1,y1,x2,y2,1)
```

```
    else
```

```
      Draw(x1,y1,x2,y2,0)
```

```
    end
```

end;

procedure DrawCircle(xc,yc,r,colour:integer);

```
{ Uses the Michener implementation of Bresenham's algorithm:
  see Foley & van Dam (1983) Fundamentals of Interactive
```

Computer Graphics, Published by Addison-Wesley, p445

```

}

procedure CirclePoints(xc,yc,x,y,colour:integer);

begin
  pset(xc+x,yc+y shr 1,colour);
  pset(xc+y,yc+x shr 1,colour);
  pset(xc+y,yc-x shr 1,colour);
  pset(xc+x,yc-y shr 1,colour);
  pset(xc-x,yc-y shr 1,colour);
  pset(xc-y,yc-x shr 1,colour);
  pset(xc-y,yc+x shr 1,colour);
  pset(xc-x,yc+y shr 1,colour)
end;

var
  x,y,d : integer;
begin
  x := 0;
  y := r;
  d := 3 - r shl 1;
  while x < y do
    begin
      CirclePoints(xc,yc,x,y,colour);
      if d < 0
      then d := d + x * 4 + 6
      else
        begin
          d := d + (x-y) * 4 + 10;
          y := y - 1
        end;
      x := x + 1
    end;
    if x = y then CirclePoints(xc,yc,x,y,colour)
  end;
end;

```

```

procedure DrawBox(x1,y1,x2,y2,colour:integer);

```

```

{
  Draws a single line box in a defined colour.
  x1, y1 are coords of top-left-hand corner
  x2, y2 are coords of bottom-right-hand corner
}

begin
  DrawLine(x1,y1,x2,y1,colour);
  DrawLine(x2,y1,x2,y2,colour);
  DrawLine(x2,y2,x1,y2,colour);
  Drawline(x1,y2,x1,y1,colour)
end;

```

```

procedure FillBox(x1,y1,x2,y2,colour:integer);

```

```

{
  Draws a rectangle filled with specified colour.
  x1,y1 are coords of top-left-hand corner.
  x2, y2 are coords of bottom-right-hand corner.
  These will be adjusted to the nearest whole byte.
  (Uses shift instructions for speed to calculate
  screen addresses.)
}

```

```

Var
  offset,b,y,plane,length : integer;
begin
  offset := (y1 shr 1)*80 + (y1 and 1)*8192 + x1 shr 3;
  length := (x2-x1+7) shr 3;
  for y := y1 to y2 do
    begin
      for plane := 0 to 3 do
        begin
          Port[$3DD] := BitValue[Plane];
          if (colour and BitValue[Plane]) <> 0 then
            b := $FF
          else
            b := 0;
          if odd(y) then
            FillChar(Mem[$BA00:offset],length,b)
          else
            FillChar(Mem[$B800:offset],length,b);
        end;
        if odd(y) then offset := offset + 80
      end
    end;
  end;
end;

```

Note that the routine DRAWCIRCLE draws a circle only in terms of pixel count. As the pixel spacing is different horizontally and vertically the resultant screen plot is a much extended vertical ellipse. The following amendments cause the screen to plot as near to a true circle as to be acceptable, but are much slower due to the number of times the integer operator 'div' is used:

1. In Procedure CirclePoints, alter all eight calls to Procedure pset by changing shr 1 to shl 1 div 5. (Use the Find and Replace facility in the Turbo Editor).
2. In the body of Procedure DrawCircle, between begin and x:=0; add the statement r:=r*5 shr 1;.

In the call DrawCircle(xc,yc,r,colour), the variable r now represents the radius in vertical pixels, i.e. the call DrawCircle(320,100,100,colour) will draw a centred circle of full screen height.

header, the linker can also 'link in' (hence its name) other OBJ routines which have been pre-compiled and stored in library files (There are exceptions: *QuickBASIC* produces EXE files directly).

198. Sorting Dates

Apparently the Director of the Society of Genealogists, A J Camp said recently that a person would have to be a fool to express any date wholly in numbers. 'You can say 10 September 1988 or 1988 September 10 but any person who says 10.9.1988 is a fool asking for confusion, who proves immediately that he does not know what he is doing.' He was referring to the danger of accidental transposition, let alone the confusion caused by the difference between UK (DD-MM-YY) and American (MM-DD-YY) date formats.

This is all very well, but computers use numbers to aid in sorting, using the MS-DOS command SORT, for example. Using SORT on the traditional three letter abbreviations for the months of the year, 'Apr' for April, 'Jan' for January and so on, puts August before February and September at the end. April did more or less begin the year once upon a time with help from the tail end of March before 1752, but nowadays SORT is no good at all.

You could almost solve this problem using Roman numerals for months. All the numbers up to twelve SORT properly with the exception of nine (IX) which insists on squeezing in between four (IV) and five (V). Alternatives like VIV or VIIII would get round this, but they're ugly and can't be read easily. Besides, with space at a premium August (VIII) is going to cause problems of its own. What you really need is a filter

MONTH.C – a Filter to help SORT Handle Month Names

```
#include <stdio.h>
#include <stdlib.h>

char line[255];
char months[] = "JanFebMarAprMayJunJulAugSepOctNovDec";
char codes[] = "m01m02m03m04m05m06m07m08m09m10m11m12";
int find(), compare();
void fix_line(), swap();

main()
{
    while(gets(line))          /* Read line from STDIN */
    {                          /* If not EOF then... */
        fix_line();           /* swap months & codes, */
        puts(line);           /* write line to STDOUT */
        /* ...else exit. */
    }

    fix_line()
    {
        int l_ptr;
        for(l_ptr=0; line[l_ptr]!='\0'; l_ptr++)
        {
            /* For each chr in line[],
               check for month...
            */
            if(!find(l_ptr,months,codes)) /*.and if this fails */
                find(l_ptr,codes,months); /* check for code. */
        }

        find(l_ptr,search,replace)
        int l_ptr;
        char *search, *replace;
        {
            int m_ptr, match;

            for(m_ptr=0; m_ptr<36; m_ptr+=3) /*Check at line[l_ptr]*/
            {
                /*for each month/code */
                if(!(match=compare(m_ptr,l_ptr,search)))
                    continue; /* If not found then try
                                next month/code.
                                Otherwise, ... */
                swap(m_ptr,l_ptr,replace); /* .overwrite month/code
                                              with code/month, */
                break; /* and stop searching */
            }
            return match; /* TRUE if and only if*/
                          /* month/code found. */

            compare(m_ptr,l_ptr,search)
            int m_ptr, l_ptr;
            char *search;
            {
                int ch, match;
                for(ch=0; ch<3; ch++) /* Do 3 chrs at line[l_ptr]
                                         form month/code? */
                    if(!(match=line[l_ptr+ch]==search[m_ptr+ch]))
                        break; /* Stop comparison at 1st
                                non-matching chr.
            */
                return match; /* TRUE if and only if */
                               /* month/code found. */
            }

            swap(m_ptr,l_ptr,replace)
            int m_ptr, l_ptr;
            char *replace;
            {
                int ch;
                for(ch=0; ch<3; ch++)
                    line[l_ptr+ch]=replace[m_ptr+ch];
            }
        }
    }
}
```


that turns month abbreviations into codes that SORT properly, and another to turn the codes back into months again afterwards. The program MONTH.C combines both these functions into one program, MONTH.C.

Sorting a dated file with MONTH is a three-stage process. Suppose you have an ASCII file FAMTREE.DOC that reads as follows:

```
1937 Dec 14: Auntie Mavis has first child, Fred
1937 Jan 02: Auntie Mavis marries
1935 Mar 17: Auntie Mavis comes of age
1942 Feb 22: Auntie Mavis has second child, Peggy
```

To turn this into the sorted FAMSORT.DOC, you might type:

```
month <famsort.doc >templ.doc
TEMP1.DOC has the month abbreviations replaced
with the corresponding codes, thus:
```

```
1937 m12 14:
1937 m01 02:
1935 m03 17:
1942 m02 22:
```

Next you sort this using:

```
sort <templ.doc >temp2.doc
TEMP2.DOC has these encoded dates rearranged in
the correct order, as:
```

```
1935 m03 17:
1937 m01 02:
1937 m12 14:
1942 m02 22:
```

Finally, we use MONTH once again with:

```
month <temp2.doc >famsort.doc
In the finished FAMSORT.DOC, the codes are once
again replaced with the month abbreviations:
```

```
1935 Mar 17:
1937 Jan 02:
1937 Dec 14:
1942 Feb 22:
```

You can do exactly the same job using the MS-DOS piping operator '|' with the following:

```
month <famtree.doc | sort | month >famsort.doc
This automatically creates the temporary files as they
are needed. You can even turn this into a batch file
with:
```

```
month <%1 | sort | month >%2
Call this DATESORT.BAT, and run it with a line like:
datesort famtree.doc famsort.doc
```

There is one funny little side-effect of MONTH. The program is case-sensitive, so Jan is converted to m01 but jan and JAN are left unchanged. This is enough to prevent accidental replacement most of the time but does occasionally backfire, as in the case of:

```
1776 Jul 04 - Declaration of Independence
which becomes:
1776 m07 04 - m12laration of Independence
```

Not to worry: the second time through, MONTH puts the Dec back along with the Jul!

199. Saving Screens to Disk

If you've got version 4.0 or above of *QuickBASIC*, you should find that saving and reloading a screen display as a disk file is quite easy using the commands BSAVE, BLOAD and GET (graphics). If you've got a version earlier than that, you'll have to experiment with BSAVE and BLOAD on their own. This latter approach won't work for EGA or VGA screens and could have a few unwanted side effects, but basically goes like this:

```
DEF SEG=&HB800
BSAVE "SCREEN.BIN", &H0000, &H4000
```

This saves any CGA graphics or text screens currently in memory into a disk file called SCREEN.BIN. An equivalent piece of code for a Hercules screen would be:

```
DEF SEG=&HB000
BSAVE "SCREEN.BIN", &H0000, &H8000
```

In either case, a simple BLOAD "SCREEN.BIN" will reload the screen for you, though you must be in the same display mode as the screen was saved from originally.

200. Codes for Amstrad PC1512 16-colour display

Intensity Bit 3	Red Bit 2	Green Bit 1	Blue Bit 0	Hex	Colour
0	0	0	0	0	BLACK
0	0	0	1	1	BLUE
0	0	1	0	2	GREEN
0	0	1	1	3	CYAN
0	1	0	0	4	RED
0	1	0	1	5	MAGENTA
0	1	1	0	6	BROWN
0	1	1	1	7	WHITE
1	0	0	0	8	GREY
1	0	0	1	9	LIGHT BLUE
1	0	1	0	A	LIGHT GREEN
1	0	1	1	B	LIGHT CYAN
1	1	0	0	C	LIGHT RED
1	1	0	1	D	LIGHT MAGENTA
1	1	1	0	E	YELLOW
1	1	1	1	F	INTENSE WHITE

201. Using MS-DOS From GW BASIC

```
10 KEY OFF
20 CLS
30 INPUT "Enter the file specification to search for "; FSPEC$
35 FSIZE=LEN(FSPEC$)
40 IF FSIZE<1 OR FSIZE>8 THEN PRINT "1 to 8 chars only " : GOTO 30

50 'Changes lower case to upper case.
60 FOR I=1 TO FSIZE
70 J=INSTR("abcdefghijklmnopqrstuvwxyz", MID$(FSPEC$,I,1))
80 IF J>0 THEN MID$(FSPEC$,I,1) =
    MID$("ABCDEFGHIJKLMNOPQRSTUVWXYZ",J,1)
90 NEXT I

100 'Shells the appropriate command and moves the prompt
110 'so it doesn't intrude on the display.
120 SHELL "CHKDSK/V|FIND "+CHR$(34)+FSPEC$+CHR$(34)+"|MORE"
130 LOCATE 23,1
140 SYSTEM
```

A useful command for finding files that have been lost within a complex directory structure is:

```
chkdsk/v | find "filespec" | more
```

Where `filespec` is all or part of the lost file specification. A disadvantage of this routine is that you must remember to put the filespec in capital letters as FIND is case specific. You can, however, overcome this with a little routine in BASIC which converts all input to upper case and then SHELLs the DOS command.

The SHELL command is indeed quite handy as it enables you to call MS-DOS routines such as FIND and SORT – the latter being particularly useful as it is faster than any sort you could write in *GW-Basic*.

A more conventional way of executing lines 70 and 80 would be to set

```
FPART = ASC(MID$(FSPEC$,I,1))
```

and then treat it numerically using the fact that the upper case letters range from ascii code 65 to 90, while the lower case letters are 32 places above, running from 97 to 122. You can then convert back to text using CHR\$ – but this wouldn't necessarily be any quicker than the method used here. ●

T H E
**HELP
SCREEN**
C O L L E C T I O N

137.



D E B U G H A C K S

202. Using DEBUG

DEBUG is essentially a memory editor. However, by calling it with the command **DEBUG filename** the file specified is loaded into memory where it can be manipulated. Any changes can then be saved back to disk using the Write command, turning DEBUG into a disk file or sector editor as well. The file is loaded into memory from offset address 100h, unless otherwise specified.

DEBUG is command-driven from its own hyphen prompt. The commands are run from a single letter, followed by optional parameters. These parameters can be separated from the command letter by a space, and multiple parameters can usually be separated either by a space or a comma. DEBUG indicates syntax errors with an **Error** message and a marker indicating which particular bit of the line it didn't understand.

DATA & ADDRESSES

Data and address codes are normally expressed using hexadecimal (to a base of 16) numbers, as in 0,1,2...D,E,Fh (the h is used to distinguish hexadecimal from decimal). This is because each hexadecimal digit is equivalent to a 'nybble' (yes really) of four binary bits: 15 being Fh in hex and 1111 in binary. So two hexadecimal digits are required to express an eight-bit byte, or the contents of a memory address.

The 8086 processor used by the Amstrad PC is capable of addressing a megabyte of memory in all, which would need a five-digit hexadecimal address code of 00000h to FFFFFh. Just to be difficult the 8086 uses a strange system of segment pointers (or addresses) and offset addresses, each 16-bit codes (requiring four hex digits).

DEBUG uses segmented addressing, but it is fairly easy to translate into absolute address code. For example, the address **B800:212A**:

```
B800 +
  212A =
BA12A h
```

A Typical DEBUG Memory Display

segmented address	The contents in hex of the next 16 addresses, from 0100 to 010Fh	The ASCII equivalents
53DB:0100	4D 41 54 5A 27 53 20 44-49 53 4B 28 00 00 00 00	MATT'S DISK(....
53DB:0110	00 00 00 00 00 00 AF 78-97 0F 00 00 00 00 00	x.....

DISK SECTORS

When a 360K floppy disk is formatted, both surfaces are divided into 40 concentric tracks, numbered 0 to 39, which are each further divided into nine 512-byte sectors. For the purpose of storing data, sectors are grouped into 'clusters', each being two sectors on a 360K floppy and four on a hard disk. Most of these sectors are used to store your data, but the first 12 are reserved for administrative purposes.

Logical Sector 0: The 'boot' sector containing basic information about the disk.

Sectors 1 & 2: The File Allocation Table ('FAT'), containing details of the contents of each cluster.

Sectors 3 & 4: A backup copy of the FAT.

Sectors 5 to 12: The directory, containing file names and pointers to the FAT.

Next on a system disk come the hidden operating system files IO.SYS and MSDOS.SYS. Hard disk space is used in a similar fashion, but with a rather larger FAT and directory.

DEBUG INSTRUCTIONS

a address

Assembles the instructions following into machine code, storing them from the *address* specified. Enter [return] on its own to stop assembly. If *a* is used without parameters it will either store from address 100h, or from where it left off the last time it was used.

n filename

Sets the name of the file DEBUG operates on the next time you use the Load or Write commands. Can be used if you don't wish to overwrite the original.

r cx

Displays the contents of the CX register (or whatever), and a colon prompt for you to enter a new value, or hit [return] on its own to leave the contents unchanged. If *r* is used without parameters the contents of all the 8086's registers are displayed.

The contents of the CX register determine how many bytes are written by the Write command. After a file is loaded using DEBUG the CX and BX registers both hold the length of the file.

```
1 100 0 5 7
```

loads disk sectors into memory. This would load into *offset address* 100h, from the disk in *drive A* (0 for A, 1 for B, 2 for C, and so on), the contents of 7 *sectors* starting with the 5th sector. What is actually specified is not sectors, but logical records. These correspond to sectors but are counted from 0h to 2D0h, to cover the full 720 sectors on a 360K floppy.

If 1 is used on its own, it loads the file last Named into the memory from address 100h.

q
quit DEBUG and return to MS-DOS prompt.

o port value

Outputs *value* to the *port* specified.

i port

Inputs and displays one byte from the *port* specified.

u address

Unassembles, or translates, memory from *address* specified. If *address* not specified, the disassembly starts from 100h, or from where it left off previously.

d 100 f00

Displays hex and ASCII dump of memory, starting from *offset address* 100h, and ending at f00h. If the second address is omitted it displays the next 128 bytes. **d** on its own dumps the 128 bytes following from the previous dump.

e address data

Enters data into memory at the *offset address*. If the *data* is not specified, DEBUG displays the present contents followed by a stop prompt for the new ASCII value you wish to enter. Enter the new value, or string of values, and hit [return]. Press [space] to advance to the next byte, or [hyphen] to return to the previous byte. Data is overwritten, not inserted.

w 100 0 5 7

Writes the contents of memory from *offset address* 100h onto drive A (see Load command above), starting at relative sector 5 and finishing at 0Dh seven sectors later. USE WITH CARE: this process bypasses the MS-DOS file handler.

w

Writes the default file back to disk.

h value value

Hexadecimal calculator: displays first the sum and then the difference between the two hex values in hex.

s 100,ffff,1B 7E

s 100,ffff,"string"

Searches from *offset address* 100h to ffff (or whatever you specify) for any occurrence of the hex value 1B followed by 7E (as many as you wish, each separated by a space), or of the ASCII *string*. Displays the addresses where it is found.

DRIVING DEBUG

DEBUG can be used as a rather limited assembler, in which case it is rather simpler to drive it from a *textfile* containing your source code and the necessary DEBUG commands, than to enter lines individually at

the DEBUG prompt. Create a text file, using RPED or a suitable word processor, along these lines:

n filename

a

source code

source code...

r cx

filelength

w

q

Calling DEBUG with the line **DEBUG < textfile** loads DEBUG and executes the commands contained in the file, creating a COM file called *filename*. Note that it is very important to hit return twice after typing the last line of source code, resulting in a blank line before the **r cx**, as this exits the assembly mode entered with the **a** command at the start.



203. Disabling Print Screen

This little program can be used to disable the [PrtSc] key so that hitting it accidentally doesn't hang the computer if a printer is not attached.

n noprtsc.com

a

jmp 0103

iret

mov dx,0102

mov ax,2505

int 21

mov dx,11

mov ax,3100

int 21

rcx

13

w

q

To create NOPRTSC.COM enter the lines of the program into a text file called NOPRT.TXT exactly as written, using RPED or a similar text editor. Note that the blank line after the last **int 21** must be present.

Now, with a disk containing the DEBUG program, enter **DEBUG <noprt.txt**. This will load debug, execute the commands to create NOPRTSC.COM, and exit back to the A-prompt. A file called NOPRTSC.COM should have been created on the disk. Executing this file means that pressing [Shift][PrtSc] will have no effect until the computer is rebooted or GRAPHICS.COM or similar program is loaded. The command NOPRTSC can be placed in the AUTOEXEC.BAT file if required.



204. Automatic menus

This program works rather like the CHOOSE.COM utility from the *WordStar 1512* program disks in that it can be used to set up automatic menus.

To create CHOICE.COM, as the program is called, first create a text file called CHOICE.TXT. Use a straight ASCII text editor such as EDLIN or RPED, or *WordStar* in non-document mode. CHOICE.TXT could even be created using `copy con choice.txt`, but it must be identical to the following:

```
n choice.com
a
call 0119
call 012D
call 0132
jnz 0103
mov ah,4c
mov cl,[01c3]
mov al,[0142]
sub al,cl
int 21
ret
mov di,0142
mov si,0080
mov ch,00
mov cl,[si]
dec cl
mov [di],cl
inc di
inc si
inc si
repz
movsb
ret
mov ah,07
int 21
ret
mov di,0142
mov ch,00
mov cl,[di]
inc di
repnz
scasb
mov di,01c3
mov [di],cl
ret
[return]
rcx
42
w
q
```

Press [return] at the point indicated to leave a blank line between `ret` and `rcx`. Now enter:

```
debug < choice.txt
```

This feeds the instructions contained in the text file into DEBUG, creating a new file called CHOICE.COM. It requires the DEBUG utility to be in the current directory or on the current path.

To use CHOICE.COM you type `CHOICE xxxx`, where xxxx is a string of as many letters as you wish it to accept. Upper and lower case are treated as different by this program, which allows up to 127 different choices. The errorlevel returned when you press a key is the position of the character in the xxxx string, which can be checked in a batch file with something like `IF ERRORLEVEL n GOTO label`.

One peculiarity about the IF command is that it performs the GOTO if the errorlevel is above or equal to the required number. This means you must set up the series of IF statements starting with the highest errorlevel expected if it is to jump to the required label in the batch file. You can test CHOICE.COM by creating a batch file such as:

```
ECHO OFF
:start
ECHO.
ECHO Choose A, B or C
ECHO or X to exit
choice abcX
IF ERRORLEVEL 4 GOTO end
IF ERRORLEVEL 3 GOTO third
IF ERRORLEVEL 2 GOTO second
IF ERRORLEVEL 1 GOTO first
:third
ECHO You chose C
GOTO start
:second
ECHO You chose B
GOTO start
:first
ECHO You chose A
GOTO start
:end
```

When this batch file reaches the CHOICE command it waits for you to press one of the keys in the string, and returns the DOS variable ERRORLEVEL with a value corresponding to the position of the letter in the string. So in our example ERRORLEVEL would be 1 if you'd pressed [a], 2 if you'd chosen [b], and so on.

It would be advisable to copy DEBUG.EXE and your text editor onto a blank floppy disk and experiment there until you know CHOICE.COM is working properly. DEBUG can easily do a lot of damage if it is not used carefully, so don't experiment on a hard disk packed with valuable data!



205. Better Cursor for Portable

The Amstrad PPC512 and PPC640 portable, and indeed many laptop machines that use an LCD screen, suffer from a totally inadequate cursor. However you can use the following procedure to turn the cursor into a flashing block which is rather easier to see.

First, create a text file called BLOCK.TXT containing the lines:

```
n block.com
a 100
mov ah,1
mov cx,10C
int 10
int 20
[return]
r cx
9
w
q
```

Now enter `DEBUG < block.txt`. This creates a tiny file called BLOCK.COM which, when run, changes the cursor to a flashing block. Unfortunately some programs turn the cursor back into a line, but many leave it as a block making it rather easy to see!

206. Automatic Reset

It can be useful to be able reset your computer from within a batch file. The following program, for Amstrad PC owners only, use DEBUG to create a small machine code file called REBOOT.COM which allows you to do just that. Type the following:

```
debug
n reboot.com
a
out 66,AL
[return]
rcx
2
w
q
```

Note that after `out 66,AL` you press [Return] twice. You should now have a file called REBOOT.COM in the root directory of your hard disk. Try typing REBOOT – your machine should reload MS-DOS as if from [Ctrl][Alt][Del].

You could, for example, use REBOOT to swap between two separate CONFIG.SYS files. Create a file called CONFIG.RDK which, together with the usual contents of your CONFIG.SYS, contains the line `device=ramdrive.sys 100`.

Create another file called CONFIG.NDK which doesn't. Now create two batch files in your hard disk root directory, called RDISK.BAT and NODISK.BAT. RDISK.BAT should contain the following:

```
copy config.rdk config.sys
reboot
```

while NODISK.BAT should contain:

```
copy config.ndk config.sys
reboot
```

The effect of these batch files is to copy the appropriate CONFIG file (with or without the RAM disk) to CONFIG.SYS, then reboot MS-DOS.

An alternative program follows which, though longer, should reset any machine with an 8088, 8086 or 80286 processor at the very least:

```
a100
mov ax,0000
push ax
popf
mov es,ax
mov ds,ax
mov ss,ax
jmp FFFF:0000

rcx
10
n reboot.com
w
q
```



207. Double Column Directories

Just changing one byte in certain versions of COMMAND.COM is enough to produce double column directory listings for the command DIR. Find the string:

```
A8 01 B0 01 74 02 B0 05
```

and change it to

```
A8 01 B0 02 74 02 B0 05
```

This gives two columns side by side.

This almost certainly will not work on every version of MS-DOS, so try it out on a copy of your version first!

208. Blanking Floppy Disks

Emptying a floppy disk of files ready for re-use can be a tedious process if the disk is full of sub-directories or read-only files. The quickest way is often to reformat it, although even this is a time-consuming business. Here is a brutal but effective process which does the trick in a trice.

The technique uses our old friend DEBUG, and is based on the fact that to clear a disk of files and sub-directories, you don't need to erase any data clusters (the parts of the disks holding the files themselves); you just have to reset the File Allocation Table (FAT)

and Root Directory to read 'empty'. This frees all the data clusters for re-use by MS-DOS.

To do this, first make a copy of the boot sector, FAT and root directory of a newly-formatted floppy, store it in a data file on a working disk, along with DEBUG.COM. Then all you have to do is write this data to the relevant sectors of each target floppy, converting it into a blank data disk.

First, some important warnings. This technique works only with disks which have already been formatted, and does not work with boot disks (i.e. ones you can load MS-DOS from). Most important, it doesn't work with floppies which have bad sectors – if in doubt run CHKDSK on the disk in question first. Bad sectors are pretty rare on floppies, so if you find many on your machine, have your floppy disk drive checked out! Finally, writing to disks with DEBUG can be dangerous – we advise you to use only spare floppy disks for the program and work files (even if you have a hard disk), and we take no responsibility for the effects!

The following instructions make use of logical drives A and B, which means that they will work on single floppy, twin floppy or hard disk machines. On a single floppy machine, just follow the prompts to change disk:

1. We start by making the copy of a blank disk's boot, FAT and root sectors. To do this, format (or re-format) a floppy and keep it handy. Then take a spare work

disk and copy DEBUG.COM (from directory C:\MSDOS or your DOS issue disk) onto it. You'll also need a text editor such as RPED or EDLIN.

2. Put the work disk in drive A and type DEBUG. You'll get the famous user-friendly – prompt. Next load the formatted disk's starting sectors by typing:

```
L 100 1 0 0C
```

This means 'load into memory at address hex 100, reading from physical drive 1 (logical drive B), starting at sector 0 and reading 12 sectors (i.e. sectors 0 through to 11)'. On machines with one floppy drive, you'll get the 'load diskette for drive B' message.

3. Now save the data to your work disk by entering the following at the prompt:

```
n blnkflop.dat
```

```
rbx
```

```
0
```

```
rcx
```

```
1800
```

```
w
```

```
q
```

This means 'set the file name for writing data to BLNKFLOP.DAT, set the amount of data to be written to hex 1800, write the file and quit DEBUG'. Note that after the RBX and RCX commands you'll get displays showing the current values of the BX and CX registers – the 0 and 1800 you type in replace these values.

4. Next we need to create a text file which can be 're-directed' to form the input to DEBUG when you want to use the program. What this has to do is write the contents of BLNKFLOP.DAT – held in memory at hex address 100 – onto the boot, FAT and directory of the disk to be blanked. Using any ASCII text editor, create a file called BLNKFLOP.TXT containing:

```
W 100 1 0 0C
```

```
Q
```

When input to DEBUG this writes the contents of memory from hex address 100 into sectors 0 to 11 of the disk in drive B.

5. Finally, we can write the batch file that sets the whole thing going. BLNKFLOP.BAT contains just one line:

```
debug blnkflop.dat < blnkflop.txt
```

And that's it! The files needed to run BLNKFLOP are BLNKFLOP.BAT, DAT and TXT, plus DEBUG.COM. To blank your floppy disk, simply put the disk to be blanked in drive B and enter **blnkflop**, either with the disk containing the required files in drive A, or in the current directory of your hard disk.

This procedure is designed for disks formatted to 360K. It can also be applied to 1.2M disks, but

209. Reversing Caps Lock

For some reason some applications leave [Caps Lock] reversed on exit. The following text file can be run through DEBUG to create CAPS.COM, which should put the key back to normal:

```
n caps.com
a
mov ax,0040
mov es,ax
es:
mov dx,[0017]
xor dx,0040
es:
mov [0017],dx
mov ax,4C00
int 21
```

```
rcx
18
w
q
```

Once you've created this text file, type **debug < caps.txt**. As always, remember the extra [return] after **int 21** to turn the assembler off.

remember that the FAT and root directory areas are rather larger here so you would need to copy from sector 0 right through to sector 28. It even works on 3.5-inch disks, where the relevant sectors are 0 through to 13 for a 720K format, or 32 for a 1.44M format.



210. Resetting the Mouse

The Amstrad Mouse driver initialises in text mode and inserts cursor key codes into the buffer, so that when the mouse is moved the computer thinks the cursor keys have been pressed. This is very useful, since it allows the Amstrad mouse to work with almost all text-based programs.

Unfortunately, many graphics-based programs use the mouse in graphics mode, but then fail to reset it to this mode on exit. The result is that the mouse no longer works in the same way unless you re-boot the computer.

The following short program will reset the mouse. Just type in the following instructions, after running DEBUG from the MS-DOS prompt:

```
a 100
mov ax,0
int 33
int 20

n rstmouse.com
rbx
0
rcx
7
w
q
```

This will create the program RSTMOUSE.COM which restores the mouse to text mode. If you call any offending graphics programs from a batch file, including `rstmouse` at the end of the batch file will automate the reset process.

There are times when the mouse driver can become a positive nuisance, and you end up leaving the little critter lying on its back, collecting dust. However RSTMOUSE can be changed into a mouse-disabling program simply by using `mov ax,3` instead of `mov ax,0` and the command `n dismouse.com` instead of `n rstmouse.com`. You'll need to have both the files RSTMOUSE.COM and the new DISMOUSE.COM if you want to switch the mouse on and off at will.

There's one little twist here. If you're writing a batch file that switches the mouse on you have to cope with two different situations: one where MOUSE.COM has not yet been installed, and one where it has been but DISMOUSE.COM has been used to switch it off again.

The easiest way to handle this is to use:

```
mouse >NUL
rstmouse
```

If MOUSE is already in memory the first line will generate an error message, which is sent to NUL rather than displayed on screen. The mouse can then be switched back on with RSTMOUSE.

211. File Attributes

Each entry in the root directory area of an MS-DOS disk takes up 32 bytes. The first eight are reserved for the file name, the next three for the extension, while the next byte determines the attributes that have been set. Each bit of this byte is allocated a particular attribute:

Bit	Value	Meaning if set
0	01h	Read-only
1	02h	Hidden
2	04h	System
3	08h	Volume label
4	10h	Sub-directory
5	20h	Modified since last archive

Bits 6 and 7 (values 40h and 80h) are reserved. More than one attribute can be set at a time, so hidden system files would have this byte set to 06h, for example.

Note that both sub-directories and volume labels are treated in exactly the same way as files, so it is quite possible to hide a subdirectory by setting the attribute byte to 12h (setting byte 1 and 4, in other words). Such a sub-directory would not show up on a directory listing, although it would still respond to sub-directory commands such as RD and CD.

The MS-DOS command ATTRIB can be used to set the Read-only and archive bits, but the others can only be modified using something like DEBUG, *PC Tools* or *Norton Utilities*. It is fairly easy to identify the attribute byte, as it comes immediately after the filename which is usually in readable text. Not all attribute combinations are effective though, so setting a sub-directory to Read-only has no effect at all.

212. Reading the Keyboard

The IF ERRORLEVEL..GOTO structure in a batch file acts on the 'return code' left behind by the last program executed. The return code is a number passed back to MS-DOS by a program when it terminates, and is normally used to indicate whether or not the program was successful. However, since the return code can actually be any number between 0 and 255, it can clearly be used to pass on a good deal more information than this. It would, for example, be useful

for returning details of which key has been pressed by the user.

The following program sets the return code equal either to the ASCII character that the pressed key would normally generate, or to the key's 'auxiliary byte value' plus 128. The auxiliary byte value is a code used to distinguish between the many different keys that generate the ASCII code 0. When you say that a key's extended ASCII code is 0;60 its the same as saying that its ASCII code is zero and its auxiliary byte value is 60.

Adding 128 lets you tell the difference between [<], for example, which has an ASCII value of 60 and [F2], which has an auxiliary byte value of 60. By and large this is a good scheme, because ASCII values from the QWERTY key set are always less than 128 so higher return codes are obviously auxiliary byte values. Unfortunately some key-combinations generate auxiliary byte values which are over 128, so the result after the addition is over 256 and too big to be a return code.

In practice the value 'wraps round' – comes back on at the zero end of the scale, in other words – so such return codes will look like ASCII values. For example, both [Alt][=] and [Ctrl][C] produce a return code of 3, even though the former has an auxiliary byte value of 131 and the latter is a true ASCII code. In particular, if you have an AT-style keyboard with 12 function keys, you'll have problems detecting [F11] and [F12] accurately. Anyway, here is the program code:

```
a100
mov ah,00
int 16
cmp al,00
jnz 010C
mov al,ah
add al,80
mov ah,4C
int 21
```

```
rcx
10
n getkey.com
w
q
```

As ever, you can use this script two ways. You can run DEBUG and then enter the lines of the script as written, hitting an extra [Return] for the blank line after `int 21`. Alternatively you can type the whole thing into a text file called GETKEY.ASM and run DEBUG by typing:

```
debug <getkey.asm
```

Either way you should end up with a tiny program called GETKEY.COM, which waits for a key and then returns a code based on it.

This would be used in the following format:

```
GETKEY
```

```
IF ERRORLEVEL x GOTO label
```

which would detect a key with ASCII value `x` or auxiliary byte value `x-128`. To see GETKEY in action,

213. Getting ESC into Batch Files

If you want to use batch files to control your printer or to use ANSI.SYS, one of the big problems is getting the ubiquitous ESC character (ASCII code 27, or 1B hex) into the file. The following listing creates a short file called ESCOUT.COM, which works much like the batch command ECHO but automatically precedes the text with ESC. As usual, remember the extra return after the `int 21` towards the end:

```
a 100
mov bx,0080
mov cl,[bx]
mov ch,0
inc cx
inc bx
mov byte ptr [bx],1B
mov dx,bx
mov bx,1
mov ah,40
int 21
mov ah,4C
int 21
```

```
rcx
```

```
19
n escout.com
w
q
```

The usual principles apply: either run DEBUG and type this lot in, or else type it into a text file called ESCOUT.TXT and run DEBUG with the command line:

```
debug <escout.asm
```

Remember, too, the extra [return] between `int 21` and `rcx`. Get all this right and you'll end up with a tiny (25 byte) program called ESCOUT.COM. Try the following at the MS-DOS prompt (with ANSI.SYS loaded) to see how it works:

```
escout [44;37m
escout [2J
```

You could also create the batch file ELITE.BAT which would switch an Epson-compatible printer into the elite typeface with 12 characters per inch. ELITE.BAT need only contain the one line:

```
escout M >prn
```

This simply redirects the character sequence ESC 'M' to the printer. ESCOUT.COM can be a useful little program indeed!

Try a batch file like this one:

```
ECHO OFF
ECHO Hit any key
GETKEY
IF ERRORLEVEL 188 GOTO L1
ECHO That was NOT
GOTO L2
:L1
ECHO That was
:L2
ECHO function key F2
```

This is horribly unstructured, and about as useful as a "Hello World" program, but does show the general idea. When using **ERRORLEVEL** in this way always remember that the statement **IF ERRORLEVEL x** actually means 'If the **ERRORLEVEL** is x or greater'. If you have a series of possible keypresses, you must put the **IF** statements in descending order, as in:

```
IF ERRORLEVEL 64 GOTO L1
IF ERRORLEVEL 63 GOTO L2
```

Otherwise the first **IF** would trap all valid keypresses.

214. Setting Key Repeat

The following routine uses Interrupt 16h function 03h to change the key repeat and delay:

```
a100
mov ah,03
mov al,05
mov bh,00
mov bl,00
int 16
int 20

rcx
C
n keyfast.com
w
q
```

This should set the key repetition to fast, and if the values 02h and 07h are moved into BH and BL respectively, it should reset it back to near normal speed again. The value in BH controls the delay before the first repeat occurs, and can be set between 00h for a 250 millisecond delay, up to 03h for a one second delay. BL controls the rate at which keys are repeated, and can be varied between 00h for 30 characters per second, up to 1Fh for just two characters a second.

This function is only available on ATs, PS/2s and the PCjr. You simply can't reprogram the repeat rate on a standard PC or XT clone.

215. Reading the Volume Label

As you can see from a DIR display, disks (hard and floppy) have 'volume labels' – a series of characters stored on disk which you can set with the **LABEL** command (see your MS-DOS manual for details). What's more, if you're writing your own programs you can read a disk's label without too much difficulty, and therefore check that the right disk is in the drive.

Most MS-DOS commands do not use this facility, which is a shame when it comes utilities like **BACKUP** and **RESTORE**, where it would be very handy to check that the right disk was in the drive. Batch files could, however, use a separate program which checked the volume label before you called the main program, returning an **ERRORLEVEL** of zero or one depending on whether the volume label matched the one you'd supplied.

If the program was called **CHKLABEL**, then your batch file might read something like this:

```
:L1
ECHO Place disk SPREADS in drive A
PAUSE
CHKLABEL A:SPREADS
IF ERRORLEVEL 1 GOTO NOTFOUND
GOTO FOUND
:NOTFOUND
ECHO ERROR - Wrong Disk!
GOTO L1
:FOUND
ECHO OK - proceeding with program
```

The only thing missing is the program **CHKLABEL**. Here's a **DEBUG** script that produces a suitable **COM** file:

```
a100
mov bx,0080
mov cl,[bx]
add bl,cl
inc bx
mov byte ptr [bx],00
mov ah,4E
mov cx,0008
mov dx,0082
int 21
mov ax,4C00
adc al,00
int 21

rcx
1C
n chklabel.com
w
q
```

This program does have a few quirks. For example, on

THE HELP SCREEN

C O L L E C T I O N

volume labels longer than eight characters you'll need to use a full stop after the eighth character, as if the volume label was actually a filename:

ECHO Place disk LONGISHNAME in drive A:

PAUSE

CHKLBL A:LONGISHN.AME

On the plus side however, CHKLBL.COM takes wildcards so that something like:

ECHO Place any backup disk in drive A
PAUSE

CHKLBL A:BKP*.*

will accept any volume with a label that starts BKP. This suggests all sorts of intricate naming systems, if you've got the patience to set them up properly.

If you want to use this with BACKUP, note that in version 3.3 of MS-DOS the BACKUP program re-labels disks to BACKUP.001, and so on, regardless of any original label. ●

T H E
**HELP
SCREEN**
C O L L E C T I O N

147

12

T H E D I S K

On the Disk

Available with *The Help Screen Collection* is a disk containing programs and listings from the tips in this book (those marked with a disk symbol), together with a number of useful utilities. This disk comes in two versions, either a standard 5.25-inch floppy formatted to 360K, or a 3.5-inch microfloppy formatted to 720K. Both versions contain the same files. The files are organised into a number of sub-directories, corresponding to the chapters from which the listings came. The root directory is occupied by the three utility programs – a disk sector editor, a file editor and a text editor – which are based on programs that have appeared on the *PC PLUS SuperDisk* in the past.

Please note that many of the programs cannot be directly run from the disk. Some are intended for use in batch files and do not do anything on their own, while others are library routines intended for incorporation in your own programs. Remember too that, while we have checked these programs as thoroughly as possible, we cannot guarantee that they will operate properly on every type of IBM PC compatible, and cannot take any responsibility for any damage caused as a result of using this disk. Some of these programs – in particular the disk sector editor – are capable of rendering a disk unusable, so use with care and never experiment on a disk containing the only copy of vital files!

Chapter 3

TW.BAT

This is a simple batch file that turns your computer and printer into an electronic typewriter. Note that it can hang your machine if your printer is not connected and switched on-line.

WILDFIND.BAT

The MS-DOS command FIND can search a file for a specified string of characters. This batch file, based on one sent in by C A Cussen of Wincanton, Somerset, works in the same way, but allows the file specification to include wildcards so that a number of files can be searched in one go. Format is **WILDFIND "text" filespec**, but note that the "text" string must not include any spaces. The file FIND.EXE must be accessible, too.

DELB.BAT

This deletes all the files in a sub-directory except those specified, and was sent in by John Robinson of Durham. It is used in the following format:

```
DELB file1 file2 file3...
```

NEWPRMPT.BAT

A custom prompt that makes good use of the facilities offered by the MS-DOS PROMPT command to liven up the standard **C>**. Requires the ANSI screen driver, which means the file ANSI.SYS should be in the root directory and the file CONFIG.SYS should contain the line **device=ansi.sys**.

FKEY.BAT & FKEY1.BAT

These, sent in by M Fuzzey of Wallington, Surrey, again use the ANSI.SYS driver but this time to allow the function keys to be redefined to execute MS-DOS commands. The format is **fkey n /e string**, where **n** is the number of the key to be redefined, **string** is the new definition, and the optional **/e** adds a [return] to the end of the string so the command executes automatically. For example, to set [F2] to display the directory of drive A, enter **fkey 2 /e dir a: /p**.

Chapter 4

UKPND.BAT

If you are having problems getting a £ character on your printer, this batch file might help you work out what's wrong. It sends ASCII codes 9C and 23 (hex) to your printer, which are the IBM Extended ASCII codes for the £ and the # characters respectively.

FF.BAT

This short batch file simply sends the ASCII code 12 to the printer, which most models interpret as a 'form feed', and wind on to the top of the next page.

Chapter 5

BASPATCH.BAS

There are a number of bugs in version 1.12 of *BASIC 2*, as supplied with earlier Amstrad PC1512's. This *BASIC 2* program, provided by Locomotive Software who wrote *BASIC 2*, converts a copy of version 1.12 into 1.12B, fixing these bugs in the process. It should

only be run on version 1.12 – you can check the version number by clicking on 'About BASIC2...' under the 'BASIC2' option on the menu bar. This program assumes the file *BASIC2.APP* is in a subdirectory called *\BASIC2* on a disk in the computer's A drive.

Chapter 7

FINDFILE.BAT

A batch file that uses the *CHKDSK /V* option, coupled with the *FIND* command, to find files that you have mislaid on the directory tree. Use as in *FINDFILE filespec*, where *filespec* is the whole filename, or just a fragment. *FINDFILE* does not take wild cards, but can search on just part of a file name. It is a good idea to run *CHKDSK* before using *FINDFILE*, as lost clusters will throw the operation of the batch file. Enter *findfile* by itself for instructions, and remember that it won't work unless the files *FIND.EXE*,

CHKDSK.COM and *MORE.COM* are accessible through the current path.

UPDATE.BAT

Many of us only work on a few files each day, making a full MS-DOS backup regime a little unnecessary. This batch file uses the MS-DOS command *REPLACE* to update files on a floppy or hard disk so that both can be kept up to date. Enter *update* on its own for instructions, and remember that *REPLACE.EXE* must be accessible for it to work.

Chapter 9

CLR1512.TXT & COM

The Amstrad PC1512's graphics adaptor is CGA-compatible, but adaptations built in to cope with its special 16-colour mode can cause problems with some applications. In particular, users of *Lotus 1-2-3* and *AutoCAD* have reported a peculiar ghosting, where graphs are not erased properly from screen but remain, even after quitting the program. This program, executed by entering *clr1512* at the MS-DOS prompt, ensures that the screen is completely erased. The accompanying *TXT* file is a text file containing the *DEBUG* 'script' used to create the program.

GIFT.TXT & CDR

Fans of *Elite* will find in *GIFT.CDR* a *Cobra MkIII* fitted with two military lasers, a docking computer, extra energy unit, fuel scoops, ECM system, cargo bay extension and four missiles. To use *GIFT.CDR* you should first copy it on to your *Elite* play disk, and then use the MS-DOS command *attrib -a gift.cdr* to reset the archive bit of the file so *Elite* will recognise it properly. This will only work if *ATTRIB.EXE* is on the current directory path. Good hunting, and thanks to Mike Davis for the *GIFT*!

Chapter 10

These are standard ASCII text files, containing source code in *BASIC*, *Pascal* or *C* (depending on the file extension). Some constitute whole programs that can pretty well be run directly from a suitable interpreter or compiler, while others are library routines for you to experiment with. They are provided as a starting point, and may need considerable modification before they will run on your particular compiler. For full instructions on their use you will usually need to refer to the corresponding tip in the main bulk of Chapter 10, which you will find earlier in the book.

SHELL.BAS

A *GW-Basic* program, sent in by Ian Masters of Thelford, Norfolk, that demonstrates the Shell-Metzner

sort technique by generating 2500 random numbers and then sorting them into numeric order. It should run with little modification under *QuickBASIC* or *Turbo Basic* too, but can take quite some time on even a 386-based machine!

DEFCHAR.BAS

Paul Mitchell of Aberdeen supplied this *GW-Basic* program that enables users to define their own characters when in medium or high-resolution screen modes (use the *SCREEN 1* or *2* command). Note that *GRAFTABL* has to be loaded beforehand. See the tip 'User-defined Characters' for detailed instructions on using this routine.

VERIFY.C & RANDOM.C

The source code for two C routines, sent in by Colin Walls of Wokingham, which use the MS-DOS system clock to generate an effectively random number, provided the request time is fairly random. VERIFY displays the numbers generated, so you can check for randomness, while RANDOM returns the random number as an error value.

MOUSE.BAS

User's of Microsoft's *QuickBASIC* compiler may find these subroutines, supplied by P A Thyer of Barnet, useful. They use the CALL INTERRUPT feature of the language to provide programmers with a number of mouse routines, and require *QuickBASIC* to be loaded with `QB /L`. For full details, see the tip 'Reading the Mouse' in Chapter 10.

SCREEN.BAS

A short *GW-Basic* listing that evokes the system screen dump routine, and should work with GRAPHICS loaded, too. See the tip 'Screen Dumps from GW-Basic' for further details.

BINARY.C

A C program that defines a function `binary()` which takes an integer parameter and returns the address of

a global string `binstr[]` containing the value expressed as a binary number. See 'Binary from C' in the main text for further details.

1512.PAS

A whole series of *Turbo Pascal* routines, supplied by Dr Alan Mynett of West Sussex, allowing you to make use of the Amstrad PC1512's unique 16-colour graphics mode. See 'Using Amstrad's 16-Colour Mode' in Chapter 10 for details on using these routines, and modifications that can improve their performance.

MONTH.C

The source code for an MS-DOS filter that can help sort a list of dates that have been expressed using 'Jan', 'Feb', 'Mar' and so on, into month order. See 'Sorting Dates' in Chapter 10 for full details of its use.

FINDFILE.BAS

The problem with FINDFILE.BAT, covered in Chapter 7, is that the file name has to be written using capital letters if it is to work properly. This *GW-Basic* program provides a routine that allows both upper and lower case letters to be used, and demonstrates the SHELL command in action. See the tip 'Using MS-DOS From GW-Basic' for full details.

Chapter 11

All the following utilities were created using DEBUG, the assembler/debugger utility provided with MS-DOS. The TXT file is a straight ASCII text file containing the instructions for DEBUG, while the COM file is the utility itself. Most of these utilities are designed to be used in batch files, returning a series of errorlevels depending on certain conditions, such as which key was pressed, and so forth. Full details are contained in Chapter 11 itself.

CHOICE.COM & TXT

This utility, supplied by Stephen Adams, returns an error code depending on which letter key has been struck out of a specified string. It is very useful for creating on-screen menus; see the tip 'Automatic Menus' for full details on its use.

NOPRTSC.COM & TXT

Do you find yourself accidentally hitting [shift] [PrtSc] when you simply wanted a '}' character? Run this utility from Carl Phillips to disable [shift] [PrtSc] until you next reset the machine.

BLOCK.COM & TXT

Do you have problems finding the cursor on your portable's screen? Run BLOCK, written by Lt Colonel

CM Joint of Canterbury, to transform it into a full-character blob that will stand out a mile!

REBOOT.COM & TXT

A utility that will enable you to re-boot your machine from within a batch file. See 'Automatic Reset' for examples of its use.

CAPS.COM & TXT

Do some of your programs leave [CapsLock] round the wrong way? Nigel Davies of Wandsworth wrote this utility which puts it back to normal.

DISMOUSE.COM & TXT

Although a mouse can be a very useful creature, sometimes it can be a real nuisance. This utility, based on RSTMOUSE below, enables you to disable the mouse. It does not appear to work with serial mice, but works fine with the Amstrad PC1512 version.

RSTMOUSE.COM & TXT

Having disabled the mouse, it would be nice to be able to turn it back on again. Dr Alan Mynett supplied this utility which nicely complements DISMOUSE. See 'Resetting the Mouse' for full details on the use of these two utilities.

ESCAUT.COM & TXT

A handy utility that works much like the batch command ECHO, but precedes the string with an ESC character (ASCII code 27). See 'Getting ESC into Batch Files' for full instructions.

GETKEY.COM & TXT

Waits for a key press and then returns its ASCII value or its Auxiliary Byte value, making it useful for setting up on-screen menus. See 'Reading the Keyboard' for details.

CHKLABEL.COM & TXT

A handy utility that returns an error code depending on the volume label of a disk. See 'Reading the Volume Label' for instructions on its use.

KEYFAST.COM & TXT

On an AT compatible it is possible to change the length of time you have to press a key before it repeats, and the rate at which it repeats. This utility sets both to fast, although altering specific values in KEYFAST.TXT can set any speed. Further details in 'Setting Key Repeat', and remember that this utility will not work with an ordinary PC compatible.

The Utilities

ED.EXE & README

Batch files, DEBUG scripts and the like can be created using a full-blown word processor, but it is much easier to use a simple text editor. MS-DOS comes with the cumbersome and rather dated EDLIN, while Amstrad PC owners are rather more fortunate with RPED. For the rest of us, something like ED fits the bill perfectly.

ED first appeared on the *PC PLUS SuperDisk* back in February 1988, but the version on *The Help Screen Collection* disk is considerably improved, with drop-down menus and whole range of new features. To load ED, simply enter `ed` and then the name of the file you wish to edit (`readme`, for example), or `[return]` for a list of files in the current directory. Once the file is loaded, press `[F10]` for the menu, `[H]` for Help and `[G]` for General Help. This lists a selection of available commands. Experiment with the menus and you will soon get the hang of it.

ED is a Shareware program. This means that, while you are free to try out the program for nothing and to distribute it to your friends for the same purpose, if you find yourself making regular use of the program you should send the author a 'registration fee' of, in this case, £10. You will then receive a disk containing the latest version, full documentation and a utility that allows you to alter screen colours. Registrations should be sent to Barry Gilpin, 32 Owler Lane, Chadderton, Oldham OL9 9PA.

Some users have reported incidental screen corruption, which can be alarming although the program is still perfectly usable. ED writes direct to screen memory and some VGA-compatible display adaptors apparently have trouble keeping up.

FRED.EXE

Standing for File Read and Edit, FRED enables you to edit files at the machine level, directly reading and altering the value of each byte. In this sense it is rather like DEBUG, but far easier to use as it provides a full-screen display and a cursor so that you can directly

select bytes. Other features include the ability to strip the high-bit so *WordStar* files are more readable; the ability to set record and header length, making it easier to read database files; and the ability to strip the high-bit altogether, converting a *WordStar* file to ASCII. There is even a game called 'Slippery Sid' when you get bored of bits and bytes.

FRED appeared on the *PC PLUS SuperDisk* in May 1989, but once again the version on *The Help Screen Collection* disk has been enhanced. FRED is also shareware, which means that, while you are free to try it out and distribute it to friends for the same purpose, if you find it useful you should send the author a registration fee of at least £15. Registrations should be sent to the author Steve Godwin, Alexander House, 50 Cavendish Road, Sutton, Surrey SM2 5LN, and you will receive the latest version, a colour and printer customisation program, a manual and a collection of MS-DOS utilities.

COLUMBUS.COM & DOC

At first sight COLUMBUS appears very similar to FRED, but COLUMBUS is a Disk Sector Editor, enabling you to get right down to the nitty gritty at the disk surface. It can be used to examine and alter the boot sector, File Allocation Table and Root Directory, creating hidden directories with impossible names, seeing how MS-DOS stores its files and even make your disk totally unreadable. You can do a lot of damage with COLUMBUS, which is why it will only work with floppy disks.

Nevertheless, COLUMBUS is very easy to use, with clear, on-screen instructions. It displays clearly which track, head and sector is being displayed and allows you to enter values as text, in hex or decimal. You can even search for a string of up to 16 characters. Columbus was written by Mark Cummins of Aware Software. Although you are free to use it as you wish you should not sell it, or any part of it, without the author's express permission.

T H E
**HELP
SCREEN**
C O L L E C T I O N

153

I N D E X

<i>Subject</i>	<i>Tip Number</i>
\$\$\$ files	2
1.2 Mbyte disks	12, 132
1.44 Mbyte disks	12, 34, 132
1512.PAS	197
3.5-inch disk drives	30, 33, 34
3.5-inch disks	3, 12, 132
300 dpi	78
360 dpi	78
360K disks	12
4201.CPI	35
4208.CPI	35
5.25-inch disks	3, 12
5202.CPI	35
720K disk drives	30, 33
720K disks	12, 132
8086 processor	14, 177, 191
< and > command	19, 29, 192
@ECHO OFF command	34
@ROUND function (Lotus 1-2-3)	166
A4 paper	80
ACC files	2
ALT key	51, 55
AND operator	186
ANSI terminal emulation	137
ANSI.SYS	7
Changing screen colours	54
Defining function keys	56
ESC character, generating	213
Fancy prompts	53
Generating the ESC character	55
Graphics from the keyboard	58
PROMPT command, using with	53, 54
Useful books	59
APP files	2
APPEND command	34
ASCII code	13
Full table	61
Pound signs	72
ASCII code 3	51
ASCII code 10 (line feed)	65
ASCII code 12 (form feed)	66
ASCII code 13 (CR)	65
ASCII code 26	27
ASCII code 27	55, 213
ASCII codes	51, 55
ATTRIB command	20, 211
AUTOEXEC.BAT	6, 7, 34
Acorn BBC B	121
Adventure game hints	170
Alternative keystrokes	127
Amstrad CPC	121
Amstrad PC1512	
16-colour display mode	8, 164, 197, 200
640K, expanding to	120
CD model	8
Changing graphic cards	8
Changing screen colours	129
Checking machine type	174
DOS PLUS, disabling of	108
Display adaptor	136
Display problems, AutoCAD	164

<i>Subject</i>	<i>Tip Number</i>
Display problems, Lotus 1-2-3	164
Extended flow control	126
Hard disk, speeding up	112
Interrupt 15	174
MD model	8
Monochrome monitor	8
NVR bug	126
NewWord, on monochrome display	155
RAM disk, problems with	126
Reset, automatic	206
Amstrad PC1512 and 1640	
BASIC 2, using with	90, 92
Clock, slowing down	157
Disk drive, adding a	132
Fast, running too	158
GEM, how it works	93
Hard disk problems	128
Joystick problems	123, 124
MS-DOS version 3.3, upgrading to	36
Mouse compatibility	122
Mouse driver, patching	157
QuickBASIC, keyboard problems	145
Useful utilities	45, 47
Windows, compatibility problems	122
Amstrad PC1640	
Monitors	8
Internal Graphics Adaptor (IGA)	8
Amstrad PC2000	
Disk drive, adding a	141
Amstrad PCW	121
Amstrad Portable	205, 49
Apricot F, PC and Xi	121
Assembly language	177
Command line, using the	192
Day of the week, finding	194
Useful books	191
Atari ST, mouse of	130
AutoCAD, display problems	164
Auxiliary byte value	212
BACKUP command	34, 114
BAK files	2
BAS files	2
BASIC	14
BASIC 2	2
Automatic program loading	181
GOTO command	181
Indexed Files	100
Keyed files	100
MS-DOS, running from	95
Machine independence	92
Maximising available memory	89
Mouse, using the	90
Printing from	89
Screen coordinates	90, 92
User Guide	100
Version 1.12, problems with	99
BASPATCH.BAS	99
BAT files	2, 24
BBC model B	121
BIN files	2
BINARY.C	196

Subject	Tip Number
BLOCK.COM	205
BUFFERS command	34
Background printing	19
Backup files	2
Bad sectors	12
Batch files	2
@ECHO OFF	34
CALL command	34
Checking command line syntax	48
Errorlevels	204
Escape codes, generating	213
GOTO command	181
Loading BASIC 2 programs	181
Menus, creating your own	46, 204, 212
Providing instructions	52
Replaceable variables	39, 192
Reset, automatic	206
The basics	4
The batch language	39
Useful books	59
Volume label, reading the	215
Batteries, checking level	49, 135
Bin feeders	71
Binary converter	179
Binary files	2, 183
Bit maps	150
Bitwise instructions	186
Blanking floppy disks	208
Books	10, 59, 140, 191
Boot sector	32
Border colour, correcting	159
Borland	15
Bulletin Boards (BBS)	137
Byte, what it is	13
C language	
Binary, converting into	196
MONTH.C	198
Random numbers, generating	188
Useful books	191
CALL command	34
[Caps Lock], reversing	209
CAPS.COM	209
CCIT standard	137
CD command	9
CGA (Colour Graphics Adaptor)	8, 92, 136
CHKDSK command	23, 110, 113
CHKLABEL.COM	215
CHOICE.COM	41, 204
CLR1512.COM	164
CLS command	9, 55
COM files	2, 24, 177, 183, 195
COMMAND command	39, 147
COMMAND.COM, missing	16
COMSPEC variable (MS-DOS)	39
CONFIG.SYS	7, 11
BUFFERS command	18, 34
Environment space	60
RAM disks	42
Reset, from batch file	206
SHELL command	60
STACK command	17

Subject	Tip Number
COPY command	9
/B switch	27
Copying sub-directories	22
Creating text files	4, 11
File transfer	121
Printing files	5
Problems with concatenation	27
COUNTRY command	35, 72
CP/M emulation	146
CPI (Code Page Information) files	35
CR, ASCII code 13	65
CS register (Code Segment)	177
Calculation errors	184
Canon PW-1080 printer	63
Caps Lock, reversing	209
Centronics port	65, 81
Century Hutchinson	140
Changing file name	9
Clearing the screen	9
Clusters, hard disks	110
Code Pages	34, 35
Code, for adventure game clues	170
Colour screen dumps	73
Coloured disks, problems with	131
Colours, changing	54, 129
CommStar software	121
Commodore PC40	131
Compilers	15, 180
Concatenation of files	9
Control codes	55, 61, 75
Control-Z	27
Corrupted disk directories	37
Country settings	34, 35, 74
Cursor, block	205
Cut Sheet Feeders	71
[Ctrl] [C]	51
[Ctrl] [N] in WordStar	154
[Ctrl] [P]	5, 109
[Ctrl] [Z]	27
[Ctrl], in WordStar	151
Cylinders, hard disk	118
DATE command	9, 49
DATTEST.COM	49
DEBUG	
Assembler, using as	177
BACKUP, checking the directory path	114
Blanking floppy disks	208
Border colour, correcting	159
Cheating at Elite	173
Commands	202
Customising print drivers	67
Directories, double column display	207
ESC character, generating	55
File editing	202
Memory addressing	13
Partitioning a hard disk	117
Patching Amstrad mouse driver	157
Re-directing text files	29
Text files, driving with	202
WordStar 1512, customising	70
DEFCHAR.BAS	178

<i>Subject</i>	<i>Tip Number</i>
DEL command	9, 44
DELB.BAT	44
DEVICE command	141
DIP switches	62, 72, 74
DIR command	9, 22, 207
DISKCOPY command	19, 30
DISMOUSE.COM	210
DISPCHK.COM	45
DISPLAY command	8
DISPLAY.SYS file	35
DOC files	2
DOS PLUS	
Disabling of on Amstrad PC1512	108
PIP command	121
WordStar to ASCII conversion	156
DRVNT.COM	45
DTR/DSR or DCD	137
Daisywheel printers	83
Decimal converter	179
Delete, selective	44
Deleted file recovery	38
Device drivers	7
Diablo printers	83
Directories	9, 22
Double column display	207
Disabled, computing for the	134
Disk Drives	
Adding a	132
Amstrad PC2000, second drive	141
Clusters	110
Counting	45
Sectors	110, 202
Disk Manager (Ontrack)	118
Disks	
Blanking floppy disks	208
Double Density	12
Sticking	125
Volume label, reading the	215
Displays	
Checking type of	45, 174
Lack of	135
dpi (dots per inch)	78
ECHO command	39, 51, 57
EDLIN utility	55
EGA (Enhanced Graphics Adaptor)	8, 92, 136
EGA.CPI	35
EOF (End Of File) marker	27
EPSMONH6.SYS	67, 93
ERRORLEVEL	39, 212
ESC character	55, 67, 213
ESC codes for printers	75
ESCOUT.COM	213
ETX/ACK	137
EXE files	2, 24, 183, 195
EXE2BIN utility	183
Elite (Firebird game)	173
Environment space	60
Environment variables	39
Epson printers	63, 83
Character set	74
Control codes	75

<i>Subject</i>	<i>Tip Number</i>
Print drivers	67
Problems with pound signs	72
Erasing files	9
Error messages	17, 24
Escape codes	67, 75, 82, 213
Euro PC (Schneider)	130
Executable files	183
Expanded memory, with Windows	105
Expanding Amstrad PC1512 memory	120
Extended Flow Control	126
Extended memory, Windows	105
External commands	19
[F11] and [F12], scan codes	185
FASTOPEN command	34
FAT (File Allocation Table)	23, 32, 37, 38, 110, 202, 208
FDISK utility	34, 112, 117
FF (form feed), ASCII code 12	66
FF.BAT	66
FILEnnnn.CHK files	23, 110
FILEnnnn.REC files	23
FIND command	43, 113
finding sub-directories	26
FINDFILE.BAS	201
FINDFILE.BAT	113
FKEY.BAT	56
FOR command (MS-DOS)	28, 39, 43, 48
FORMAT command	7, 32, 33, 112
FX Epson printers	83
Fatal Stack Errors	17
File Allocation Table (FAT)	23, 32, 37, 38, 110, 202, 208
File attributes	20, 38, 211
File names	2, 9, 24
File transfer	121
25-pin RS232 socket	138
9-pin RS232 socket	138
Handshaking	137
Finding lost files	113, 201
Floppy disks	12
Blanking	208
Coloured disks, problems with	131
Sectors	32
Sticking	125
Tracks	32
Volume label, reading the	215
Fantasy, picture files	150
Fonts, built in	78
Form feeds, sending to a printer	66
Formatting disks	12
Frozen, unexpected hang-up	142
Function keys	21, 56
GEM	1
AES (Application Environment Services)	93
Customising print drivers	67
Desktop Accessories	2
EPSMONH6.SYS	93
Graphics drivers	8
Hard Disk installation	94
How it works	93
Maximising memory	89, 97
Output	2
Printing GEM screens	87

Subject	Tip Number
RAM Disk, running from.....	98
Removing desk accessories	89
Removing fonts.....	89
Running from MS-DOS.....	89, 96
Single disk operation	96
Stripy tiger problem.....	67
VDI (Virtual Device Interface)	93
GEM Desktop	
Configuring Applications	84
Installing disk drives.....	84, 132
Keyboard, operating from the	91
Saving preferences.....	86
Using the windows	86
GEM Paint	
On one disk.....	85
On two disks	88
Picture files	150
GETKEY.COM	212
GIFT.CDR	173
GOTO command (MS-DOS)	39, 48, 181
GRAFTABL command.....	72, 178
GRAPHICS command	7, 73, 87
GW-BASIC	2, 178
ASCII text, saving as	189
Clock, slowing Amstrad	157
Fantasy images	150
MS-DOS, returning to	187
Printing the screen	193
SAVE command.....	189
SHELL command.....	201
SYSTEM command	187
Shell-Metzner sort.....	175
User-defined characters	178
Games Adaptor	124
Games , problems with	123
General failure error	125
Graphic characters.....	58
Graphics adaptors.....	8, 136
HIMEM.SYS	104, 105
Handshaking	121, 137
Hard disks	
32 Mbyte, greater than	117
AT drive types	118
Amstrad PC1512, non-booting	128
Backing up with REPLACE	116
Backing up with XCOPY.....	111
Clusters.....	110
Cylinders	118
Drive types, AT	118
FDISK.....	34, 117
Finding lost files	113
GEM, installation.....	94
Interleave factor	112
Miniscribe models	115
Partitions	34, 117
RLL (Run Length Limited)	112, 115
Seagate models	115
Sectors	110, 112, 118
Speeding up a	112
Tracks	112, 118
Un-jamming	139

Subject	Tip Number
Useful books	119
Western Digital	115
Hard-sectored disks	12
Hercules display	8, 92, 136
Hewlett Packard LaserJet	83
Hexadecimal addition.....	182
Hexadecimal converter	179
Hexadecimal numbers	13, 179, 202
High Density disks	12
Hitchhiker's Guide to the Galaxy	172
Hugo Cornwall	140
IBM	
Character set	72, 74
Extended ASCII code	61, 72
Games adaptor.....	124
Graphics Printer.....	72, 83
Proprinter.....	83
IDC (Insulation Displacement Connector).....	141
IF command (batch files)	39
IMG files	2
IO.SYS	202
Inaccurate mathematics	184
Infocom games	170, 171, 172
Installing software	6
Integer variables, overflow	184
Intel 8086 processor	14, 177
Interleave factor.....	112
Intermediate code	15
Internal commands	9
International characters	34, 35, 74
Interpreters.....	15, 180
Interrupt 10, checking display adaptor	174
Interrupt 15, checking for PC1512	174
Interrupt 16, key repeat	214
Interrupt 17, printer status	174
Interrupt 33 (the mouse)	190
Joining files together	9, 27
Joysticks	123, 124
KEYB command	34, 35
KEYBOARD.SYS	35
KEYBUK.EXE	7, 36, 72
KEYFAST.COM	214
Kermit.....	121
Keyboard	
Alternative keystrokes.....	127
Auxiliary byte value.....	212
Generating graphic characters	58
Key repeat, setting	214
Keys stop working.....	143
QuickBASIC problems	145
Reading the	212
[F11] and [F12] scan codes.....	185
Keyboard ID	35
LCD screens	205
LCD.CPI.....	35
LF (line feed), ASCII code 10.....	65
LIM EMS version 4.0, Windows	105
LINK utility	195
LIS files	2
LQ Epson printers	83
Laser printers	78, 83

<i>Subject</i>	<i>Tip Number</i>
Leather Goddesses of Phobos	170
Line spacing	67
Lines per page, changing	64
Lost clusters	23, 110
Lost files	31, 38, 201
Lotus 1-2-3	
@ROUND function	166
Calculation errors	166
Display problems, Amstrad PC1512	164
Macro Commands	169
Macros, auto-loading	152
Printing a spreadsheet	161
Low level format	112, 117
MCGA display	136
MD command	9
MDA (Monochrome Display Adaptor)	8, 136
MODE command	121, 159, 162
MONTH.C	198
MORE utility	19
MOUSE.BAS	190
MOUSE.COM	7
MS-DOS	
< and > command	192
APPEND command	34
Batch Files	39
Books	10
COMSPEC variable	39
Code Pages	34, 35
Corrupted disk directories	37
Country settings	35
Directories, double column	207
Directory structure	32
Editing the command line	21
Environment space	60
Environment variables	39
Errorlevel	39, 212
External commands	19
FASTOPEN command	34
FDISK command	117
FOR command	28
File attributes	211
File types	2
Filename priority	24
Functions	192, 194
Hidden files	202
Internal commands	9
International settings	35
Interrupt calls	174
Keyboard ID	35
Piping command	29
Re-direction	29, 192
Version 3.3	34, 36
Version 4.00	35
MSDOS.SYS	202
MX-80 Epson printer	83
Machine code	15, 177
Books	191
Command line, using the	192
Day of the week, finding	194
Macros	152, 169
Menus, creating your own	41, 46, 204, 212

<i>Subject</i>	<i>Tip Number</i>
Microfloppy disks	12
Micronet	149
Microsoft Mouse	122
Microsoft Press	10, 59, 119
Microsoft Windows – see Windows	
Miniscribe hard disks	115
Mouse	7, 122
Amstrad PC1512 running fast	158
Amstrad driver, slowing clock	157
Cheap, for Schneider Euro PC	130
Disabling	210
Interrupt 33	190
QuickBASIC, reading the	190
Resetting	210
Multiple work disks	42
Mysterious files	29
NEC V30 processor	146
NEWPRMPT.BAT	53
NLQ (Near Letter Quality)	74
NLSFUNC command	35
NOPTSC.COM	203
NUL, redirecting to	39, 57
NVR (Non Volatile RAM)	126, 143
NVRPAT2.COM	47
NewWord – see WordStar	
Amstrad PC1512	155
Monochrome displays	155
NWCHANGE	148
RAM disk, running from	42
Surplus line feeds	65
Non-contiguous blocks	23
Norton Utilities	37, 112, 158
Nybble	202
OR operator	186
OUT files	2
OVR files	2
Offset address	177
Operating systems	1, 7
Overlay files	2
PATH command	9, 24, 50
PAUSE Command (Batch files)	57
PC-Write	55, 153
PIP command (DOS PLUS)	121, 156
PRINT command (MS-DOS)	19
PRINTER.SYS file	35
PRN device	5, 82
PROMPT command	7, 53, 54, 56
PS/2	8
PSP (Program Segment Prefix)	192
Paper length	64, 80
Pascal	192, 197
Piping (MS-DOS)	19, 29, 51
PostScript files	150
Pound signs	62, 72
Power C Works	15
Prestel	137, 149
Print Screen, disabling	203
Print drivers	67, 70, 72, 83
Printers	
24-pin printers	78
Automatic line feed	65, 74

Subject	Tip Number
Batch file control	82
Character definition	75
Checking status of	174, 176
Cut sheet feeders	71
DIP switches	65, 72, 74
ESC codes	75
Epson character set	72, 74
Epson control codes	75
Errors	81
Form feeds, sending	66
GW-BASIC, screen dumps	193
IBM character set	72, 74
Intermittent problems	69
Laser printers	78
Line feeds, surplus	65
Line spacing	75
Loose printer chips	69
Operating commands	75
Page format control	76
Page length	74, 80
Pound sign problems	72
Printing GEM screens	87
Ribbons, revitalising	77
Serial printer	162
Single sheet paper	79
Star NL-10	83
Stripy Tiger problem	68
Tab setting	76
Type styles	75
Weak pin	68
Zero character, slashed	74
Printing	
BASIC 2, printing from	89
Background printing	19
MS-DOS, files from	5
Spreadsheets	161
Problems	
Unexpected hang-up	142
Procomm	121
Program files	24
Programming	
8086 processor	177
AND operator	186
Amstrad 16-colour mode	197, 200
Automatic loading	181
Bitwise instructions	186
Books	191
Command line, using the	192
Dates, sorting	198
Day of the week, finding	194
Display adaptor, checking type	174
Inaccurate calculations	184
Integer variables	184
Masks, using	186
OR operator	186
Printer status, checking	174, 176
Random numbers, generating	188
Rounding errors	180
XOR operator	186
[F11] and [F12] scan codes	185
Protocol, communications	137

Subject	Tip Number
QuickBASIC – see GW-BASIC	2, 15
BLOAD command	199
BSAVE command	199
Clock, slowing Amstrad	157
Command line, using the	192
Fantasy images	150
GET command	150
Keyboard compatibility	145
Mouse, reading the	190
Screens, saving on disk	199
Shell-Metzner sort	175
RAM disks	6, 7, 47
Checking size of	25
GEM, running from	98
Problems with	126
Windows, using with	105
RAMDRIVE.SYS	7
RANDOM.C	188
RD command	9
REBOOT.COM	206
RECOVER command	23
REN command	9
REPLACE command	116
RESTORE command	34
RLL (Run Length Limited)	112, 115
ROS (Resident Operating System)	124
RS232 interface	121, 137, 138
RSC files	2
RSTMOUSE.COM	210
RTS/CTS	137
Random numbers, generating	188
Re-direction in MS-DOS	19, 29, 192
Read-only attribute	20
Recovering deleted files	38
Registers, 8086	177
Renaming files	9
Repeat rate, keyboard	214
Replaceable variables (batch files)	39, 52
Resetting your machine	6
Resource files	2
Ribbons, printer	77
Root directories	32
Rounding errors in programming	180
Runtime module	15
SCREEN.BAS	193
SET command (MS-DOS)	39
SETUP command	141
SHELL.BAS	175
SHIFT command (batch files)	39
SIMCGA	8
SMARTDRV.SYS	104, 105
SORT command	198
SUBST command	168
SYS files	2
Schneider Euro PC	130
Screen colours, changing	54, 129
Screen, problems with	135
Seagate hard disks	115
Searching files for text	43
Sectors, disk	32, 110, 112, 118, 202
Segmented addressing	14, 177, 202

<i>Subject</i>	<i>Tip Number</i>
Selective delete.....	44
Serial port.....	121, 137
Sheet feeders.....	71
Shell-Metzner sort.....	175
[Shift] [PrtSc].....	73, 203
Sinclair QL and Spectrum.....	121
Single floppy machines, work disks.....	42
Soft-sectored disks.....	12
Software installation.....	3
Source code.....	15
Speedstor (Storage Dimensions).....	118
Spinrite (Gibson Research).....	112
Spreadsheets	
@ROUND function.....	166
Calculation errors.....	166
Macro commands.....	169
Printing.....	161
Stacks.....	17
Star NL-10 printer.....	83
Stripy Tiger problem.....	67, 68
Sub-directories.....	22, 32
Deleting.....	28
File attribute.....	211
GEM, changing directory.....	86
Listing.....	26
SuperCalc, using a serial printer.....	162
Surplus lines when printing.....	65
System files.....	2
TIME command	9
tpi (tracks per inch).....	12
TREE command.....	109
TW.BAT batch file.....	40
TYPE command.....	9
Telecom Gold.....	137
Teletype terminal ('dumb').....	137
Temporary files.....	2, 29
Terminating batch files.....	51
The Twin (spreadsheets).....	152
Tracks, disk.....	32, 112, 118
Trinity (Infocom).....	171
Turbo BASIC.....	15
Turbo C.....	15
Turbo Pascal.....	15
Turnkey disk.....	7
UKPND.BAT	72
UPDATE.BAT.....	116
Umlauts in Microsoft Word.....	165
Undeleting files.....	38
Useful books.....	10
V21, 22, 23, 24	137
VAT, calculating in a spreadsheet.....	166
VERIFY ON.....	7
VERIFY.C.....	188
VGA (Video Graphics Array).....	8, 92, 136
VP-Planner.....	152
VT100 and VT52.....	137
Vector graphics.....	150
Video Digitiser (Electric Studio).....	150
Viewdata.....	137
Visually impaired, computing for the.....	134
Volume label, reading the.....	215

<i>Subject</i>	<i>Tip Number</i>
WD 40, revitalising printer ribbons	77
WILDFIND.BAT.....	43
Western Digital drives.....	112, 115
Wildcards.....	9
Windows	
Applications.....	103, 106
Control menu.....	101, 102, 107
Display adaptors, suitable.....	104
Graphics drivers.....	8
HIMEM.SYS.....	104, 105
MS-DOS Executive.....	102, 106
Memory, using.....	105, 107
Mouse compatibility.....	122
Notepad utility.....	107
PIF (Program Information File).....	106
RAM disk, using.....	105
SMARTDrive.....	104, 105
Version numbers.....	104
WIN.INI.....	107
Window control.....	102
Windows/286.....	104
Windows/386.....	104, 106, 107
Word (Microsoft), umlauts in.....	165
WordStar	
ASCII, converting to.....	156
Calculations.....	167
Control sequence commands.....	151, 163
Corrupted disk directories.....	37
Customising.....	148, 163
Directory listing.....	144
Directory, initial log on.....	148
Dot commands.....	160, 163
Generating control codes.....	55
Hard carriage returns.....	154
Headers and footers.....	160
Installation of.....	6
Lost files.....	144
Menus.....	151
Merge printing.....	160
Page formatting.....	160
Page numbering.....	160
Print commands.....	160
Print drivers, selecting.....	148
Print formatting.....	160
RAM disk, running from.....	42
Running MS-DOS from.....	147
Surplus line feeds.....	65
WSCHANGE.....	148, 154, 163
WordStar 1512	
Bold type, generating.....	70
CHOOSE.COM.....	41
Free batch file utility.....	41
Work disks.....	6, 42
Write protecting.....	3, 20
XCOPY command	19, 30, 111
XMODEM.....	121
XON/XOFF.....	121, 137
XOR operator.....	186
Zortech C	15

THE
**HELP
SCREEN**
COLLECTION

Distilled from three years of *PC PLUS* magazine, this book will help you understand and make the most of the MS-DOS operating system, batch files, printers, hard disks, assembly language, GEM, BASIC, C, Pascal, Windows, WordStar and much more. It is the essential guide to PC compatible hardware and software.

The Collection has been compiled and edited by Matt Nicholson, launch Editor of *PC PLUS*, and is based round the popular Help Screen correspondence section of the magazine. Every item has been re-written and updated where necessary, and there is a comprehensive index. The book can also be purchased with a disk containing many of the batch files and utilities from the text, plus other useful programs.

£10.95 for the book alone

£12.95 with 5.25-inch disk

£13.95 with 3.5-inch disk

Future Publishing Limited

ISBN 1 872666 00 0