USING AMSTRAD CP/M BUSINESS SOFTWARE

IAN SINCLAIR

Using Amstrad CP/M Business Software

Using Amstrad CP/M Business Software

Ian Sinclair

COLLINS
8 Grafton Street, London W1

Collins Professional and Technical Books
William Collins Sons & Co. Ltd
8 Grafton Street, London W1X 3LA

First published in Great Britain by
Collins Professional and Technical Books 1986

British Library Cataloguing in Publication Data
Sinclair, Ian R.
Using Amstrad CP/M business software.
1. Amstrad PCW8256 (Computer)—Programming
2. Amstrad PCW8512 (Computer)—Programming
3. CP/M (Computer operating system)
I. Title
005.4′465 QA76.8.A4/

ISBN 0-00-383359-3

Typeset by V & M Graphics Ltd, Aylesbury, Bucks
Printed and bound in Great Britain by
Mackays of Chatham, Kent

Contents

Preface

The Amstrad PCW machines, the 8256 and the 8512, are, at the time of writing, the best-selling machines for both business and hobby use in the UK, and probably soon in the world. The reasons are not hard to see because the combination of price, availability and the software base would make these machines unbeatable even if the printer and VDU had to be bought separately. Because the machines are sold as a package with a heavy emphasis on their use for word processing, however, many users are uncertain of how to use these excellent computers for other types of business programming. One factor that deters the wider use of such programs is unfamiliarity with the operating system of the machine. Though this is a very well-known and well-established system, its reputation is higher among programmers than among users, and the Manuals for the PCW machines are not very helpful unless you have access to other information, or previous experience.

This book is therefore written for the computer user rather than for the enthusiast or the programmer, and it is concerned with how to make the most effective use of the PCW machines for business purposes. This entails making use of the utility programs that come with the machine, and understanding the operating system sufficiently to avoid the problems that beset the user who lacks such knowledge. Computers are not simple machines, and the better you know their quirks, the more effectively you can use them. This is particularly true when an old-established operating system like CP/M is in use, because this system was never intended to be used by owner-drivers, only by professional programmers and computer operators. This book assumes only that you possess and use a PCW machine, and that you have probably used the machine for word-processing only. From that point on, everything is explained in detail, with examples that make the processes easier to follow.

To emphasise the commitment to the business user, several standard business programs have been featured in this book. The aim has been to show what these programs consist of, how they are used, and to provide a few tips that will get you working with these programs sooner than would otherwise be possible. It's important to realise that many of these programs are very sophisticated indeed, and that they require considerable understanding to use. The manuals are not always models of clarity, and the sections of this book, though brief, that deal with these programs are intended to form a background, on which you can build, to the use of the manuals to gain a better understanding in a much shorter time. Many of these programs sell at very high prices to users of other computers, and are accompanied by tuition sessions. The Amstrad user is usually on his/her own in this respect, because the price of the software reflects the price of the computer, and trimmings such as free advice are not generally available, though a few firms do offer a telephone hotline service.

I am most grateful to William Poel and Chris Laing, of New Star Software, for the loan of the dBASE II, SuperCalc II and New Word programs to illustrate the latter chapters of this book. I am also considerably indebted to Mr A.D. Goldman of Sagesoft for the loan of the excellent suite of business software in its Amstrad version, and to Caxton Software for copies of their spreadsheet, database and Brainstorm programs. These also have been featured in this book. At Collins, Richard Miles, Janet Murphy and Sue Moore have, as usual, worked wonders with my discs and made a real book out of some rather chaotic prose. The last vote of thanks must go to Alan Sugar of Amstrad for providing real computers at sensible prices, so restoring the faith of millions of people in 20th century technology.

Ian Sinclair

Chapter One
The Disc System

What is a disc system?

I am assuming right from the start of this book that you have by this time obtained some experience with the keyboard, as you would have by using the machine as a word-processor, and that the actions of setting up the Amstrad computer, PCW 8256 or PCW 8512, will have been carried out. You should have the units of the system connected up together, as guided by the manual. There is not a great length of connecting cable between the units, and you will need to place the keyboard, monitor and printer fairly close to each other. This is made much easier if you can use a computing desk, such as the type that is illustrated in Figure 1.1. Either the monitor or the printer can be placed on the higher shelf, and the low shelf can be used for paper, discs and other accessories. If you are working on a plain desk, it's easier to place the monitor behind the keyboard, and put the printer to one side. Make sure that the connectors are firmly in place, but do not attempt to force them. The large printer plug will fit only one way round in its socket, and this is also true of the plug on the end of the cable from the keyboard, which fits only one way round in the socket at the side of the monitor. The connection point on the PCW 8256 that is marked 'EXPANSION' is for use with an additional disc drive, and you can ignore it if you have no second drive or if you are using the PCW 8512. You can now buy monitor-to-printer extension cables and printer-power extension cables from Amstrad dealers if you need to position the printer further away from the rest of the equipment.

Once you have fitted all the pieces together, you have an empty computer. I use the word empty because until you put a program into it, a computer is as useless as a record-player with no records. The main programs for the computer are contained on the two discs that are provided in the package. Several of these programs are very precious, and unless you have bought several of the computers, your first action must be to make copies of the discs. The most important disc to copy is the one which is marked LOCO SCRIPT on one side, and CP/M PLUS on the other. If anything happens to erase the programs on this disc, particularly on the LOCO SCRIPT side, you will find it expensive to replace. Of the two sides, the one that you particularly need to copy, as far as the material of this book is concerned, is the CP/M PLUS side.

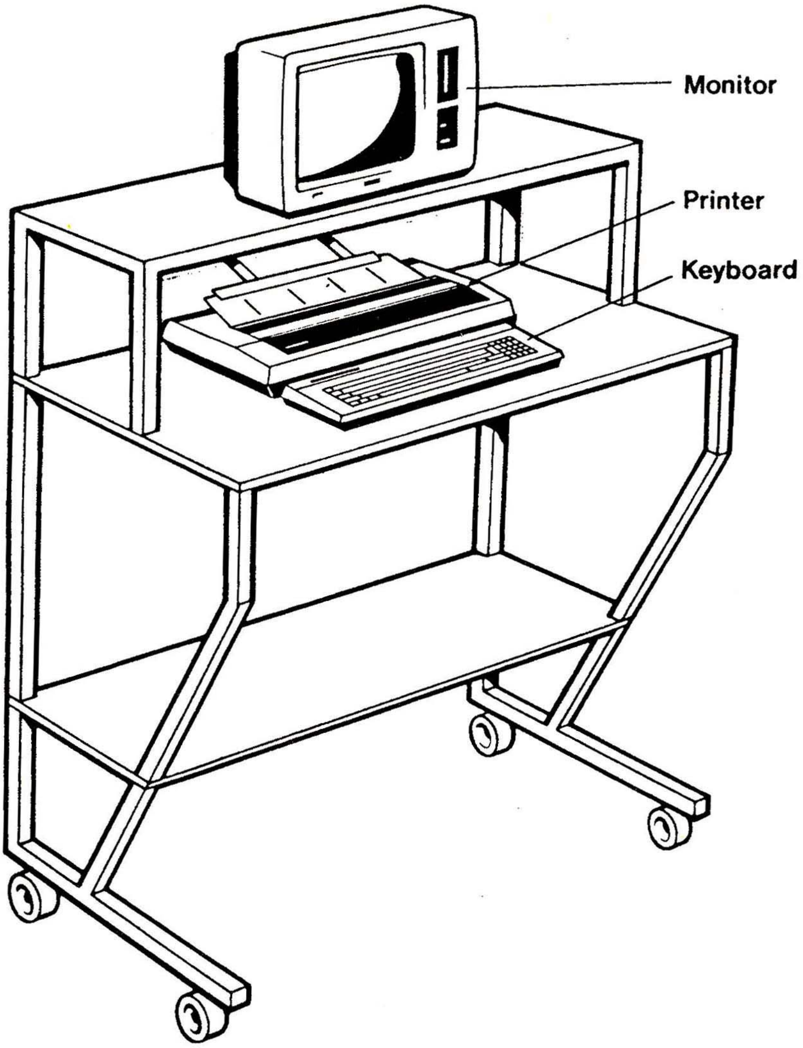


Figure 1.1 A computing desk, of the type manufactured by Selmor. This is extremely useful, because it allows the equipment to be easily moved.

The most important point to understand at this stage is that you must always *remove any disc from the drive before switching on or off, or resetting*. The act of switching on or off can generate enough magnetism near the disc to alter some stored codes. This can make the disc 'corrupted' and therefore unusable. This is one of the main reasons for needing to take a copy of very important discs. Note that if you use a PCW 8512 that you must take great care to segregate different discs. The discs for Drive B of this machine should be double-sided double-density discs, and the discs that you use in Drive A will not operate in Drive B. All program discs that are supplied are intended for use in Drive A of the PCW 8512.

Copying CP/M PLUS

You *must* make a copy of CP/M PLUS for the reasons that have just been stated. Normally, copying discs is something that you would attempt after mastering some of the essentials of using discs, but it's so important in this case that we simply have to take the subject out of sequence. Figure 1.2, then, shows the steps in making a copy of the CP/M disc side. Once you have made a copy of CP/M PLUS on one side of a disc, you can put the original master disc away in a safe place. Your copy is the one that will be used from now on. Figure 1.3 shows the precautions that you should take with discs to prevent damage. The most serious damage is caused by invisible magnetism, so the discs must never be kept on top of the monitor, or near any device (other than the disc drive itself) which contains an electric motor. In particular, never switch on, switch off, or reset (with the SHIFT-EXTRA-EXIT keys) with any disc in the drive(s).

When you have made a copy of CP/M PLUS, you can check it by typing **DIR** (or **dir**), and then pressing RETURN. This should provide the list of programs that are on this side of the Master disc, as illustrated in Figure 1.4. These programs are designed to allow you to control the action of the computer, program it to carry out your instructions, and monitor what it is doing. The initial letters of these actions, control, program and monitor, are the letters CP/M that are used for the operating system. The point at which we take up the reins, then, is with the use of the disc system software, CP/M PLUS. To understand how you can more effectively use this system, we have to start by explaining what a disc system is and what it does. You may very well point out that you don't need to understand what the parts of a car do in order to drive one, which is perfectly true in 1986. However, computers in 1986 are at the stage that cars were when Henry Ford introduced the Tin Lizzie, and in those days you certainly *did* need some understanding.

1. Switch on. When the screen lights up, insert the Master disc in Drive A, Side 2 facing left. Wait until the CP/M system loads, when you will see the 'prompt' **A>**.
2. Type **DISCKIT**, or **disckit**, RETURN, and wait for the display.
3. Press the **f5/f6** key for a COPY action.
4. Answer Y to question on disc (8256) or choose disc type for 8512.
5. Insert discs as required – Master disc is source disc and copy disc is destination disc.
6. Leave the program as advised by the screen messages.

Figure 1.2 The steps that are needed to make a copy of the CP/M Master disc side. This should be done before any attempt is made to use any of the programs. Make sure that the drive is empty when you switch the machine on or off, or reset.

Care of discs

1. Keep discs in their protective boxes when they are not inserted in the drive. If you drop a box, it will chip, but this is better than chipping the disc jacket.
2. Buy discs from a reputable source, such as Amsoft or one of the large disc suppliers. At these prices, you can't afford to take risks.
3. Never pull back the protective shutter unless you need to – which is normally never!
4. Never touch any part of the inner disc.
5. Keep your discs away from dust, liquids, smoke, heat and sunlight.
6. Avoid at all costs magnets and objects that contain magnets. These include electric motors, shavers, TV receivers and monitors, telephones, tape erasers, electric typewriters, and many other items which have surfaces that you might lay discs onto.
7. Label your discs well. If the label on the disc is not large enough, use self-adhesive labels in addition – but don't cover any of the shutters.
8. Remember that the disc is read and written from the *underside.*

Figure 1.3 Precautions that you should take with discs. Magnetic damage is the most worrying, and the golden rule is never to have a disc in the drive when switching on, off, or when resetting.

```
A: J11CPM3  EMS : BASIC    COM : DIR      COM : ED       COM : ERASE    COM
A: KEYS     WP  : LANGUAGE COM : PALETTE  COM : PAPER    COM : PIP      COM
A: PROFILE  ENG : RENAME   COM : SET      COM : SET24X80 COM : SETDEF   COM
A: SETKEYS  COM : SETLST   COM : SETSIO   COM : SHOW     COM : SUBMIT   COM
A: TYPE     COM : RPED     BAS : RPED     SUB : DISCKIT  COM
```

Figure 1.4 A listing of the files in the Master disc directory. The file J11CPM3.EMS is the CP/M operating system file – some versions will be numbered J14CPM3.EMS.

Disc System

Disc system is the name that is given to a complicated combination of hardware and software. A disc system comprises the disc drive (or drives), the disc controlling circuits, both of which are hardware, and the disc operating system, which may be in software, firmware (ROM), or a combination of both. Different manufacturers approach the design of disc systems in different ways. The approach for the later Amstrad computers has been to build at least one disc drive into the casing of the computer, and, if only one is internal, to make provision for another to be connected separately, using a connector at the back of the computer. The software which makes the disc system work is partly in the form of a chip which holds a small program that is needed to make the disc drive read the main disc program. Software in chip form cannot be altered except by plugging in other chips, and for that reason is often referred to as 'firmware'. The main parts of the control programs are held on the Master discs (the System discs) that come along with the drive. The disc system is, in fact, a miniature computer in its own right, using some of the memory of the main computer.

When you start with the CP/M PLUS disc put into the drive, the Disc Filing System (DFS) is put into the memory from the Master discs. A 'file' in this sense means any collection of data which can be stored on the disc. The important file for the PCW machines is the kind referred to as the Early Morning Start file (EMS), and you will find such a file occupying a lot of space on the Master discs. This very important file contains the whole of the CP/M operating system and without it, the computer cannot run CP/M programs. Any corruption of this program will therefore mean that you no longer have the use of your computer for these purposes, though you might still be able to use it for LOCO SCRIPT. This is why so much emphasis is placed on making backup copies of the Master discs.

Tracks, Sectors and Density

The language of disc recording is another novelty that you may have encountered for the first time with the Amstrad computer. If your sole concern is to make use of ready-made programs, you may possibly never need to know much about these disc terms. A working knowledge of how disc storage operates, however, is useful. At the simplest level, it makes it easier to understand how the disc stores data, so that you can understand why it should be that the disc has lots of unused space, but cannot accept any more data! At a more advanced level, it can allow you to extract information from damaged discs, and to make changes to information that is stored on discs.

Unlike tape, which is pulled in a straight line past a recording/replay head, a disc spins around its centre. When you insert a disc into a drive, the protective shutter is rotated so as to expose part of the disc. When the drive is activated a hub engages the central hole of the disc, clamps it, and starts to spin it at a speed of about 300 revolutions per minute. The disc itself is a circular flat piece of plastic which has been coated with magnetic material. It is enclosed in a hard plastic case to reduce the chances of damage to the surface. The hub part of the disc is also built up in plastic to avoid damage to the disc surface when it is gripped by the drive. The surface of each disc is smooth and flat, and any physical damage, such as a fingerprint or a scratch, can cause loss of recorded data. The jacket has slots and holes cut into it so that the disc drive can touch the disc at the correct places. The slot and one hole (one of each on each side) are covered by a metal shutter when the disc is withdrawn from the drive. You can see the disc surface if, holding the 'A' or '1' side uppermost, you insert your thumbnail into the slot at the right-hand side of the disc casing, near the front. By sliding your nail back, you engage the sliding peg which acts on the shutter, and you can turn the shutter until the disc surface is visible. Do *not* touch the disc surface, sneeze on it, or do anything which could leave any marks on the surface.

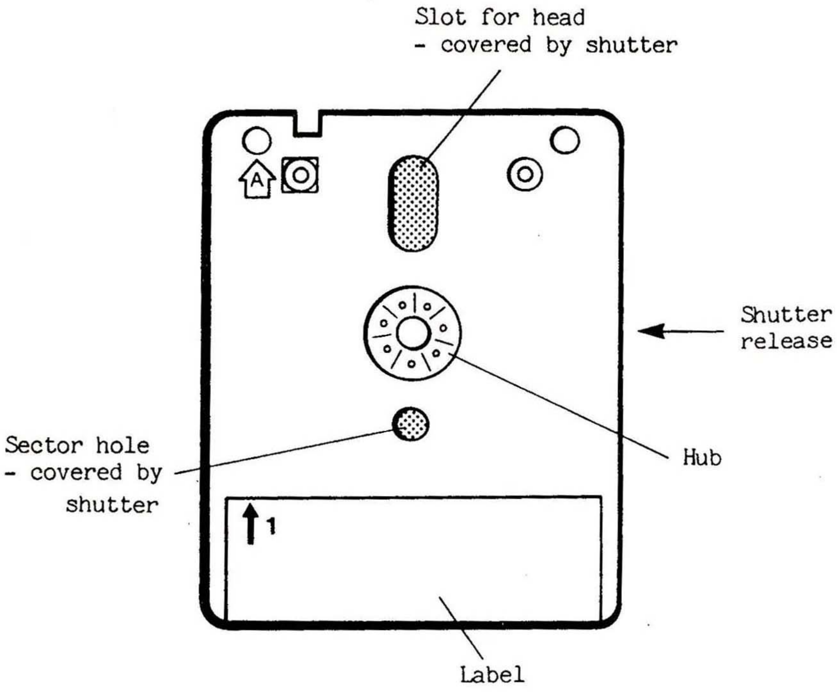


Figure 1.5 The slots that are cut into the disc casing for the disc head and the sector detecting beam.

Through the slot that is cut in the casing (Figure 1.5), the head of the disc drive can touch the surface of the disc. This head is a tiny electromagnet, and it is used both for writing data and reading. When the head writes data, electrical signals through the coils of wire in the head cause changes of magnetism. These in turn magnetise the disc surface. When the head is used for reading, the changing magnetism of the disc as it turns causes electrical signals to be generated in the coils of wire. This recording and replaying action is very similar to that of a cassette recorder, with one important difference. Cassette recorders were never designed to work with digital signals from computers, but the disc head is. The reliability of recording on a disc is therefore very much better than could ever be obtained from a cassette, which is why computers for serious use never feature the use of ordinary audio cassettes. Drive B of the PCW 8512 uses two heads, one on each side of the disc, but Drive A of either model has one head only.

Unlike the head of a cassette recorder, which does not move once it is in contact with the tape, the head of a disc drive moves quite a lot. If the head is held steady, the spinning disc will allow a circular strip of the magnetic material to be affected by the head. By moving the head in and out, to and from the centre of the disc, the drive can make contact with different circular strips of the disc. These strips are called 'tracks'. Unlike the groove of a conventional record, these are circular, not a spiral, and they are not grooves cut into the disc. The track is invisible, just as the recording on a tape is invisible. What creates the tracks is the movement of the recording/replay head of the disc drive. A rather similar situation is the choice of twin-track or four-track cassette tapes. The same tape can be recorded with two or four tracks depending on the heads that are used by the cassette recorder. There is nothing on the tape which guides the heads, or which indicates to you how many tracks exist.

The number of tracks therefore depends on your disc drives. The vast majority of disc drives for other machines use larger discs with either 40 or 80 tracks. Forty-track discs use 48 tracks per inch, and 80-track drives use 96 tracks per inch. The Amstrad computer disc drive for Drive A uses a 3-inch disc with 40 tracks, but with 96 tracks per inch. This forces you to find a source of these special discs, which at the time of writing were in very short supply. Though many of the independent suppliers were advertising these discs, very few had any in stock, so that delivery times of three months were being quoted when I started to write this book, and I was forced to re-cycle several older discs to use in the examples. Be very careful when you send for discs that you specify the 3-inch type. Most business computers, such as Apricot, have standardised on the Sony tape of 3½-inch disc, and this size is quite easy to find, and rather cheaper. It is, however, *not* suitable for your Amstrad computer disc drive. You can use either the Amsoft disc or the Maxell (also labelled 'Tatung') disc. They are practically identical except for the write-protect shutters, of which more later. At the time of writing, the price of a ten-pack was about £40, which compares with about £10 for a ten-pack of the normal 5¼-inch floppy discs for other machines. Some advertisements are now appearing for 5¼-inch disc units to add to the Amstrad machines in order to get around the problems of availability of the 3-inch discs. If you use the PCW 8512, you have the problem that Drive B uses 'double-density' discs, each side of which stores double the amount of data as compared to a Drive A disc. In addition, both sides of the disc can be used *without turning the disc over*, because there is a head on each side. This means that the two sides of the Drive B disc can be treated by the computer as if only one single large side was present – you don't, for example, have to refer to the other side as Drive C. Discs for the two drives are *not* interchangeable, and you must keep them apart and suitably labelled.

Once you have accepted the idea of invisible tracks, it's not quite so difficult to accept also that each track can be invisibly divided up. The reason for this is organisation – the data is divided into 'blocks', or sectors, each of 512 bytes. A byte is the unit of computer data; it's the amount of memory that is needed for storing one typed character, for example. Each track of the disc is divided into a number of 'sectors', and each of these sectors can store 512 bytes. Conventional 40- or 80-track discs use ten sectors per track, but the Amstrad system uses nine sectors per track on a single-density disc, allowing $512 \times 9 = 4608$ bytes to be recorded on each track. Two tracks are reserved

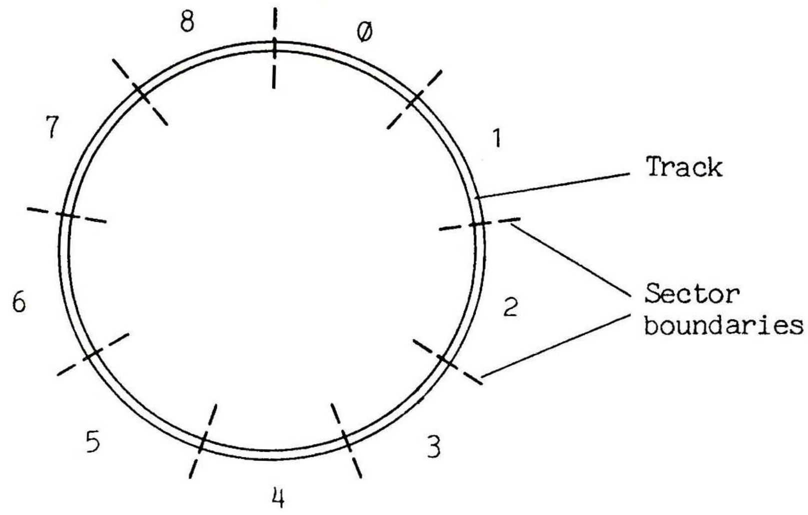


Figure 1.6 Tracks and sectors on the disc – these do not show visibly in any way.

for holding essential data, leaving 38 tracks for your use. Of these tracks, four sectors (2K) are reserved for the 'directory' entries, leaving the rest free. This corresponds to a total of 173056 bytes free, which is 169K.

The next thing that we have to consider is how the sectors are marked out. Once again, this is not a visible marking, but a magnetic one. The system is called 'soft-sectoring'. Each disc has a small hole punched into it at a distance of about 14 mm from the centre. There is also a hole cut through the disc jacket, so that when the shutter is swung aside and the disc is rotated, it is possible to see right through the hole when it comes round. When the disc is held in the disc drive, and spun, this position can be detected, using a beam of light. This is the 'marker', and the head can use this as a starting point, putting a signal on to the disc at this position, and at eight (single density) others, equally spaced, so as to form sectors (Figure 1.6). This sector marking has to be carried out on each track of the disc, which is part of the operation that is called 'formatting'.

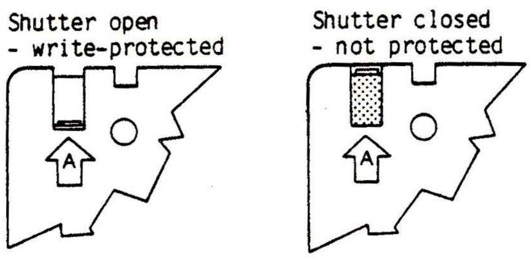
Using CP/M

CP/M is a relatively old (about 1972) and well-established operating system for microcomputers. Its purpose is to allow you to make effective use of the machine for running programs, and in order to do this it controls the action of the keyboard, the monitor screen, the printer and, of course, the disc system. The use of programs along with CP/M is delightfully simple. You start up your computer by switching on and then putting the CP/M PLUS disc side in place, so that the CP/M system is put into place (or 'booted up', in computer jargon). When the machine is ready, the prompt symbol **A>** will appear on the screen, either at the top line in an otherwise clear screen, or as the last of a set of lines on the screen. The **A** part of this means that disc drive A is in use. From then on, you can put in other discs and find out what programs are on them by typing **dir** and pressing RETURN. A word such as **dir** (or **DIR**, because you can use either upper- or lower-case) is a command word for CP/M, and in this case it's the word that requests a directory of the disc, a list of what programs are present. Once you know what programs are present, you can run a program by typing its name and then pressing RETURN. The usefulness of CP/M lies in the fact that once you have become accustomed to its actions, they will be the same for any computer that uses CP/M (the jargon phrase is 'running under CP/M'), so that you will not have to learn any new tricks if you change to another computer that uses CP/M, even if it is another variety of CP/M like CP/M 86. A further point is that other letters and words can be used along with the name of a program. Instead of simply having a command word to load a program into the memory of the computer and run it, you can have a 'command line', consisting of the program name and several other items, all separated by spaces. This action is used when you need other information at the start of a program, and that's something that we'll come to shortly.

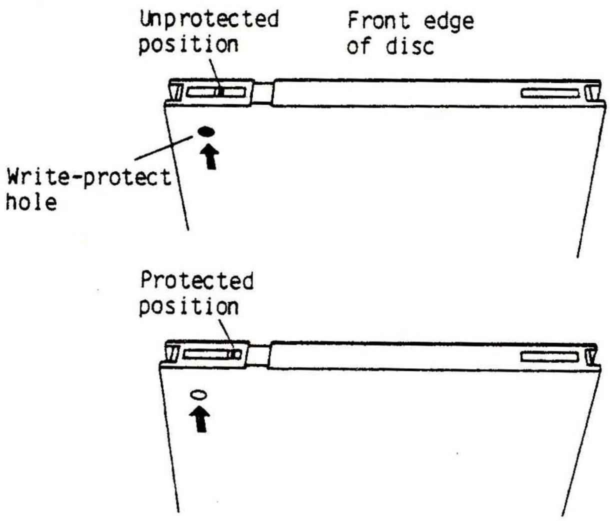
Formatting Discs

Formatting discs, as we have seen, consists partly of the action of 'marking out' the sectors on a disc. The formatting action also tests the disc. This is done by writing a pattern to each sector, and checking that an identical pattern is read back later. Failure to do so indicates a faulty sector, and a disc with such a fault should be returned to the supplier with a request for a replacement. These small discs cost more than three times as much as a conventional floppy disc, and they *ought* to be perfect. The prices would probably come down from their present very high levels if more manufacturers used them, but there is little sign of this happening.

Formatting, then, consists of marking out sectors and testing them. This takes about half a minute, and will normally end with a message which asks you if you want to format another disc. You *don't* have to format a new disc if you are going to use the DISCKIT program to make a copy of the System disc, because this particular copier formats the disc as it goes. Any fault that is found at the formatting stage will be reported. This does not necessarily imply a *disc fault* however. All 3-inch discs make use of a small sliding shutter (Figure 1.7) which exposes or covers a hole at the front left-hand side of the disc casing. If this hole is exposed, the disc is 'write-protected' which means that it cannot be formatted. There are small differences in this respect between the discs labelled as Amsoft and those labelled as Maxell. The Amsoft discs use a shutter that is operated from the flat surface of the disc. The Maxell type has a shutter that is moved by inserting the tip of a ballpoint pen into the small lever which is at the front edge of the disc. Now if the disc is protected by



(a) Amsoft discs



(b) Maxell or Tatung discs

Figure 1.7 The arrangement of sliding shutters for write-protection, as used on two varieties of discs.

opening the hole that is normally covered by this shutter, it's probably because you want to preserve some information that has been recorded on it. Since formatting will wipe the disc clean, the message is a warning to you that you might want to think again. If you really want to format, then you have to slip the plastic cover over the write-protect hole. There are *no* plastic shutters on the System discs, so as to avoid any danger of erasing them by re-formatting. This makes it impossible to carry out some actions, particularly setting keys, etc., on startup, so that you *must* make a copy of at least Side 2 of the System discs if you want to make really effective use of the CP/M facilities. Note that if you have been used to ordinary floppy discs of 5¼-inch size, that the protection action works in the *opposite* way. On the ordinary floppy, a slot has to be covered to protect the disc; on the 3-inch disc, the hole has to be *open* for protection. It's a particularly important point to remember if your second drive on a PCW 8256 is a 5¼-inch type.

```
A: BASIC    COM : DIR      COM : ED       COM : ERASE    COM : LANGUAGE COM
A: PALETTE  COM : PAPER    COM : PIP      COM : RENAME   COM : SET      COM
A: SET24X30 COM : SETDEF   COM : SETKEYS  COM : SETLST   COM : SETSIO   COM
A: SHOW     COM : SUBMIT   COM : TYPE     COM : DISCKIT  COM
```

Figure 1.8 The utilities on Side 2 of the Master disc set. BASIC is included in the listing, but you are not likely to use it as a utility for CP/M.

The formatting action is carried out when a set of instructions have been typed and entered. First of all, you will need the correct System disc side. The machine is supplied with two System (Master) discs, which are referred to as *Sides* 1 to 4, since each disc contains programs on each side. Side 2 is the most important for you at this stage. Side 2 contains the CP/M PLUS system itself, and programs, of the type that are called 'utilities' (Figure 1.8) that are designed to make your use of discs as trouble-free as possible. On the second disc, Side 3 contains more advanced utilities, mainly intended for writers of programs, and Side 4 bears a program for a computing language called LOGO which you are most unlikely to need. Note that these disc sides are specific to the 8256 and 8512 machines, and you cannot use the Master discs that were issued with earlier machines such as the CPC6128. You must also be careful about the second disc drive on the 8512, which takes 'double-density' discs that store more information than the ordinary type. To format or copy material on this drive, see your PCW 8512 manual for details.

Keeping with Drive A, then, first of all you need to select Side 2 of the System discs. You cannot take risks with these discs – they are *valuable*, and if you don't believe me, then just try getting replacements; it's bad enough trying to get blank discs! Insert the System disc in the drive, making sure that it is the correct way round. The discs are double-sided, with the sides labelled as 'A' or '1' and 'B' or '2' at the front left-hand side, next to the write-protect hole. The side number is at the right-hand side of the disc, on the Amsoft label next to the arrow. The actual recording and reading is carried out on the *opposite* side of the disc, but the labelling is placed so that side B or 2 is the one being used when this number is *facing left*. The flap on the front of the drive unit is opened when you push the disc into the slot. Hold your System disc with the 'B' or '2' side facing left. Slide the disc into the slot for the main drive. Don't use any force to do this, because you can jam the disc if you do so. Press the disc in firmly until it stays put and clicks into place. If you have been using LOCO SCRIPT, then you will first need to reset by pressing the SHIFT, EXTRA and EXIT keys together. If you have just switched the machine on, then inserting the disc is all you need to do. The System disc is now ready to control formatting. Make sure at this stage that you have a new blank disc ready to format, with the write-protection shutter over the hole.

To format a new disc you must now type the word **disckit**, and press RETURN. This will bring up the message:

One drive found

unless you happen to have a second drive connected, in which case you will see the message:

Two drives found

This is a check which can be very useful, as it will remind you if a second drive is incorrectly connected. At the bottom of the screen, you will now see a menu of the choices that are open to you. These are illustrated in Figure 1.9. The numbers refer to the *function keys* which are on the number-key pad on the right-hand side of the keyboard. If you press any of the ordinary number keys, you will get a squeak from the built-in loudspeaker to remind you of the difference. To format, then, you must press the key that is labelled **f4/f3**, and you will then see a new menu appear. This asks if you want to format the CF2 disc, the only type suitable for the machine with Drive A, and your choice is the **Y** (or **y**) key to go ahead, or any other key to cancel the action. If you are using twin disc drives, you will get another menu choice, asking in which drive the disc to be formatted is placed.

At this stage, remove the System disc from the drive, and replace it by the disc that you want to format. If you are formatting a disc for Drive B in the PCW 8512 or a twin-drive PCW 8256, the System disc need not be removed. You now start the formatting action by pressing the **Y** key. This additional action is needed because formatting a disc wipes it *completely* clear of all data, whereas other processes do not. The extra step makes you think again before wiping a disc clean, in case you have valuable data on the disc. When you press this key, the formatting will start, and the screen will show the track numbers, from 0 to 39, as the formatting proceeds. In the event of a track being impossible to format, you will be informed by a screen message, and the number of the faulty track will be displayed. Note this, so that you can quote the information when you return the disc. If the write-protect shutter has been slid aside to reveal the hole, then formatting will be impossible and you will get a message to that effect.

Once a side of the disc has been formatted, you are asked to remove the disc from the drive, and to press any key to continue. Any key in this sense should

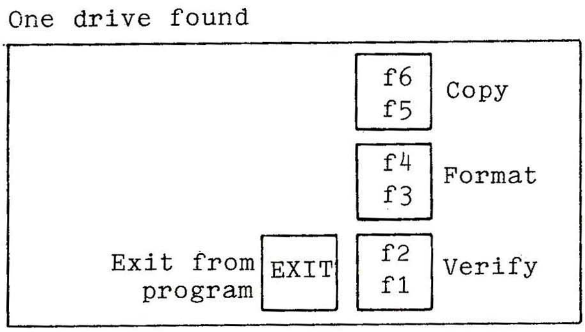


Figure 1.9 The DISCKIT main menu.

be taken as any ordinary key, number or letter, spacebar or RETURN. You can then format the other side of the disc (for Drive A) by putting it into its drive, correct way round, and pressing the Y key again. Pressing any other key at this stage will return you to the first menu again. When formatting is completed, you will probably want to return to CP/M by pressing the EXIT key.

The formatting action reserves four sectors on the third track of the disc. This portion is reserved as a way of storing information about the contents of the disc. To put it crudely, the disc system reads the first few sectors of this track to find if the filename for a program is stored on the disc, and then to find at which sector the program starts. With this information, the head of the disc drive can then be moved to the start of the program, and loading can start. This part of the track is known as the 'directory'; it keeps a record of what tracks and sectors have been used, and which of them are free for further use. The directory entries consist of filenames and numbers that indicate which track and sector is used for the start of each program or other file that is stored on the disc. The filename for a disc consists of up to eight letters for the main name, and (optionally) three letters for the 'extension'. When you 'wipe' a program or some data from the disc, all you do is to remove its directory entry – the data remains stored on the disc until it is replaced by new data. This can sometimes allow you to recover a program that you thought you had erased if you have a 'disc editing' program. Another important point about the directory is that it cannot take an unlimited number of program names, but we're coming to that!

Storage Space

How much can you store on a disc? The Amstrad single-density system uses nine sectors on each track, and a maximum of 40 tracks, 38 tracks free for your use on discs which contain the CP/M system program. This makes a maximum of 360 sectors, if the system tracks are counted as well. Of these, up to four sectors will be used for the directory entries, and this leaves 356 sectors free for you to use.

Each of these sectors will store 512 bytes, which is half of a kilobyte. If you take 356/2, then, giving 178, you end up with a figure of 178K on a single side of a 40 track drive. Not all of this will normally be usable, however, because data is not stored at every possible point on the disc. This is because the disc operating system works in complete sectors only. Suppose you have a program that is 1027 bytes long. The disc operating system will split this into groups of 512 bytes, because it can record 512 bytes on one sector. When you divide 1027 by 512, you get 2 and a fraction – but the DFS (Disc Filing System) does not deal with fractions of a sector. Three sectors will be used, even though the last sector has only three of its 512 bytes recorded. When the next program is saved, it will start at the next clear sector, so that the unused bytes are surrounded, and there is no simple way of making use of them. If you save a lot of short programs on the disc, you will find that a lot of space may be wasted in this way. There is another way in which space can be wasted if you keep a very large number of very short programs on a disc. Each program will have a separate directory entry, and when the directory track is full, no more entries can be accepted. The system, however, allows up to 64 directory entries on a disc, so you would have to be very fond of short programs to run out of directory space!

The large amount of storage space, theoretically up to 178K, on a disc contrasts with the 61K or so which you normally have available for CP/M programs on the computer. For long programs, then, a disc system can be used as a form of extra memory (called 'virtual memory'). If a long program is split into sections, the sections can be recorded on a disc, and a master program entered into the computer. This master program can then call up different sections from the disc as needed, giving the impression that a very large program is, in fact, operating. This method is extensively used for the commercial programs that are available for these machines and which we shall look at in this book. The use of a disc system therefore does not only allow you to load programs more quickly and store a lot of data, it allows you to use the computer as if it had a very much larger amount of memory.

You can print out the DIR display of any disc by loading the printer with paper, selecting the print style (draft or high quality) with the [+] or [-] keys in the usual way, and then pressing EXIT. With a disc in place, if you type DIR, and then follow it with ALT P (press ALT and P keys together) before pressing RETURN, then the directory will be printed on paper as well as appearing on the screen. You must then press ALT P again to disconnect the printing command, otherwise everything you do from then on will be printed. It's very convenient to keep printouts of your directories because you can then find what is on each disc without having to insert the disc and use DIR.

Remember that each disc has two sides, but only one side can be read by Drive A at a time. You will have to turn the disc over to 'DIR' the other side. If you have only one drive, the commands that we have looked at are all you need, but with two drives, you must specify which drive you want to use. The built-in drive on a single-drive machine is A, and this is the drive which is selected automatically at switch-on. Any actions that use Drive A can be done as before, but if you want to use Drive B then you have to type **B:**. This will select Drive B for all further use until you reselect Drive A by using **A:**. As an alternative, you can keep one drive selected but make temporary use of another. This is done by making the drive letter *part* of the filename. If, for example, you had Drive A selected, and wanted to load a program called MYFIL from Drive B, you could use:

```
A>B:MYFIL
```

Most CP/M users do not write programs, but will create and record a lot of files of information. You should note at this point that each file must carry a name, and that the disc drive will quite happily replace one recorded file with another of the same name. If you have just completed a file and you want to save it, then, always use DIR to read the directory to find if you have used a filename already. The old file is *not*, however, wiped from the disc. Instead, it is renamed with the extension label BAK. For example, suppose you wanted to save a data file called 'EFFORTS'. If there is another file of this name on the disc, it will appear in the directory as EFFORTS .DAT – the extension name of DAT has been placed there *automatically* by the action of the system. When you record your new file using the same name, it now gets the name of EFFORTS .DAT, and the other file is renamed EFFORTS .BAK. You can do this only once, though. If you save yet another file with the file name of EFFORTS, then the first one will be wiped from the disc, the second one will be renamed EFFORTS .BAK, and the latest one will be labelled EFFORTS .DAT. You can load EFFORTS .BAK only by using the LOAD command with the full filename of 'EFFORTS .BAK'.

If you really want to prevent the replacing of a program, however, this can be done, and the method (using SET) is detailed in Chapter 8. Any attempt to save a file with the same name will then bring up an error message which shows the file name followed by the message 'is read only'. This useful protection should be applied to all valuable files. Note that there is nothing on the ordinary CP/M DIR display to mark these protected files. When you have a number of valuable files on a disc, you should write-protect the whole disc by flipping back the small shutter with the end of a ballpoint pen or a nail-file. This will protect all of your files on that disc against being written over. It can't protect them from coffee or from stray magnetic fields, though!

Software on CP/M

Software on CP/M is widely available for all the computers that run CP/M, but mainly for machines that use 8-inch or 5¼-inch discs. Many of the most popular programs, however, have been put on to the Amstrad 3-inch discs, and are being sold at much lower prices than their counterparts for other machines. This does not mean that you get 'cut-down' versions of the programs, just that program prices are geared to the cost of the machine rather than to the cost of producing software. In this book, we shall look at some of the excellent business software that is available for the Amstrad PCW machines. Much of this requires the use of twin disc systems when it is used on other machines. Even the single-drive PCW 8256, however, can make use of such software, because the memory can be used as a form of 'disc', and this is referred to as Drive M, RAM disc, or as the 'silicon disc'. If you need to add another drive to the PCW 8256, it's worth considering the use of a 5¼-inch drive, so allowing you to work with cheap and readily-available discs. At the time of writing, BOX Ltd, of 22 Hendred St, Oxford OX4 2ED (Tel: 0865 717968) were advertising such disc systems in the magazine *Personal Computer World*. The same firm was offering a utility program for reading disc files produced on the IBM PC machine and machines of similar specification.

Magazines such as *Personal Computer World* are by far the best source of information on software and hardware that is available for your computer. In particular, you should look for well-established suppliers, such as New Star, Sagesoft, and Caxton whose software is featured in this book. Software of this type will have been supplied on CP/M discs for many other machines and will have been thoroughly tried and tested. New software is like a new car – it might look better than anything else you have ever read about, but until it has a good track record you can't be really sure. The main benefit of well-tried software is that every problem has been at some time investigated, and you can usually depend on getting answers to your questions if you run into difficulties. A helpful voice saying: 'Well, that's a new one – we'll ask the designer', is not a great comfort when you learn that the designer comes into the office just once a week.

Another source of answers to problems is the CP/M Users' Group. This tends to attract a more technically knowledgeable group of users than you find in computer clubs, which means that they generally know the answer to almost anything you care to ask, but you might not understand the reply first time round! The membership is, however, excellent value for money if you become really hooked on computing and in particular on exploiting the CP/M system.

Chapter Two
Drives and Files

Drives

The disc drives of the Amstrad machines are identified by letters, and on a machine with only one disc drive, such as the unaugmented PCW 8256, there will be the disc drive A and the 'silicon disc' M. It's easy and tempting to regard the M drive as another disc, but you should not do so except as a method of temporary storage. The M drive is simply another piece of computer memory. It does not hold as much data as a disc drive, particularly as compared to the double-density drive that can be attached, and its memory is volatile. Volatile means that when you switch off the computer, or if there is a momentary interruption in the power supply, enough to make the lights flicker in the room, the contents of this memory will be lost. Once lost, the memory content is gone forever, unless you have the same data stored on a real disc. The M drive should therefore be used only for convenience, to hold files that have been read in from a disc and which will later be put back on a disc. The purpose of the M drive is to allow you to use, with a single disc drive, programs such as the Sagesoft RETRIEVE which needs two drives. The Sagesoft programs will, in fact, operate without using the M drive, and the manuals show how to make use of the disc drive alone, changing discs as required. For many business purposes, in which any loss of data could be disastrous, it's better to avoid storing any data in the M drive, and use this drive only for rapid access to programs which are safely kept on a write-protected disc.

Suppose, for example, that you are using the Sagesoft database program RETRIEVE. This has the main database programs on one side of the disc and utilities on the second side. Now, the main programs take up too much space to fit into the M drive memory, but the utilities can all be placed on the M drive. You do this by the following sequence of commands.

1. Place the CP/M PLUS disc in the drive, with the CP/M side (Side 2) facing left.
2. Type **pip** and press RETURN. Wait until an asterisk appears on the screen.
3. Now remove the CP/M disc and put the RETRIEVE disc in the drive, with Side 2 facing left.
4. Type, *exactly as shown here*, **m:=a:\*.*** and then press RETURN. The files from the disc will copy into the M drive.

5. When the asterisk appears again, press ALT and C keys together. You now have the Sagesoft utilities in Drive M, and they can be called up as needed.

This allows these utilities to be accessed very rapidly, and allows you to work with a data disc in the drive. You must still change discs if you want to use the main database program, but if you are using the word-processing and mailmerge facilities of RETRIEVE, then this can be done from the M drive very conveniently. The program does not allow you to use the M drive for data, so that you cannot lose data by forgetting to transfer it from the M drive to a disc. Remember when you use the Text Editor portion of RETRIEVE that you must type fairly slowly, otherwise alternate letters will not appear – the program is not geared to fast typists! The use of the M drive is very limited when you are working with lengthy programs like the Sagesoft series, because M drive of the PCW 8256 simply does not have enough room for the program files. Some other programs written specially for the Amstrad machines will make more use of the M drive (see Chapters 9 to 11), but you should always remember that this drive should be reserved for programs and any data that is not being changed and which also exists stored on a disc.

Files

Any program or collection of data stored on a disc is called a 'file', and each file is referred to by its name, the filename. These filenames are already fixed for the programs that you buy, but in the course of using programs you will often be asked to provide a filename. The CP/M operating system has rather strict rules about such names, and if you don't obey the rules you will find problems developing. This applies just as strictly if you are using a commercial program as when you are working with CP/M on other projects. Taking the Sagesoft RETRIEVE program once again as an example, when you are asked for a filename for a document you will be presented with a set of fourteen dots to indicate the *maximum* space that your answer should take up. This does not mean that you are allowed to use a filename of fourteen letters, and failure to realise this can lead to a lot of frustration. You get a hint of this from the prompt in RETRIEVE, which asks you to enter **drive:filename**, meaning that the drive letter should be typed, then a colon and then the filename. The drive letter will be A if you are using an unexpanded PCW 8256, and A or B if you are using a PCW 8256 with second drive, or the PCW 8512. Note that RETRIEVE will not accept **M:** as a drive letter. Specifying the disc drive in this way uses two of the dots, leaving twelve to fill – but you can't use a twelve letter filename! The maximum length of filename that you can use is eight letters, the minimum is one, and the filename must start with a letter of the alphabet. This, and the other letters of the name can be in upper-case (capitals) or lower-case, and the name can contain digits provided that the name does not start with a digit. The following characters must not be used as part of a filename:

< > = , ! * ? & / $ [ ] ( ) . : ; \ + -

because they are used with special meanings. The colon, for example, is used to separate the drive letter from the name, so you are not allowed to use a colon as part of the name also.

Using eight characters for a filename leaves four places which can be used, or not, as you wish. These four places are for an 'extension' to the filename, and an extension consists of a full stop (which cannot therefore be part of a filename) followed by up to three letters. When you are using some programs these letters will usually be allocated automatically, no matter what you type. For example, if you typed a drive: filename such as **a:polymorp.hic** this would be accepted because it's within the fourteen-space size limit, with the colon and the full-stop in the correct places, but the **hic** would not appear. When the file is recorded on the disc, you would find this file appearing as **POLYMORP.DAT**. This is not inevitable, and many programs, RETRIEVE among them, accept whatever you type provided that it fits the acceptable form of drive:name.extension using one letter for drive, up to eight for name and up to three for extension. These extension letters are used to identify a type of file, and Figure 2.1 shows some of the common extension names that appear on files that you are likely to find on discs. Most of the programs that you buy for business purposes will allow you to take another attempt at

.ASC	File of ASCII text
.ASM	Assembly language program file
.BAK	Backup file
.BAS	BASIC program file
.COM	CP/M transient program file
.DAT	Data file
.DOC	Document or text file
.HEX	Machine code file
.LIB	Library file
.OBJ	Machine code (object code) file
.PRN	Assembly language listing file
.REL	Relocatable machine code file
.SUB	SUBMIT file
.TEX	Text file
.TXT	Text file
.$$$	Temporary file. Also used for an unreadable file

Figure 2.1 Extension letters that are commonly found on filenames. The list is by no means exhaustive, and commercial programs often add their own extensions.

typing a filename if you make a serious mistake that infringes the rules. If you type a name that is not one that you intended, but is valid in the sense that it doesn't infringe the rules, then you will find that this filename will be used – see Chapter 3 for details of how to retitle a file. Do *not* use the extension EMS for any files, because this is reserved for the CP/M file that runs when the CP/M master is put in just after switch-on.

The directory

It's important to know what files exist on a disc, and it's often important to know how much space a file will need on a disc. CP/M+ allows you to find this information easily, and because the Amstrad business machines have printers attached, the data on filenames can be printed out as well as observed on the screen. It's always a good idea to keep a printout of all the filenames on each disc, so that you can keep tabs on what you have without needing to place the disc into the computer each time you need the information. This is particularly important if you are using programs that read data from disc, because you will have to specify the filename for the data, and it is not always possible to obtain a directory readout while you are using some programs. It can also be useful to have a printout of how much of the disc space has been used by each file.

This information is provided by one of the CP/M utilities, called **dir**. With CP/M+ running and a disc in the drive, you can obtain the straightforward disc directory by typing simply **dir** and pressing RETURN. Unlike many other CP/M programs, **dir** is built-in, meaning that it is put into the memory of the computer when you load in the CP/M+ system. You cannot necessarily get this to work when you are in the middle of using a business program, however, so that it's best to do this kind of work either before you start using the computer for accounts, data filing, word-processing or whatever, or after a day's work on these topics has been filed and finished. If at any time you find that typing **dir** (RETURN) has no effect, you will need to load in CP/M+ by removing all discs, pressing the SHIFT, EXTRA and EXIT keys all at the same time, and then putting in the CP/M disc (Side 2 facing left).

dir, used alone in this way, provides a display such as the example in Figure 2.2. This, incidentally, is taken from the Sagesoft Popular Accounts disc. The **dir** display is printed by pressing ALT-P (ALT and P keys together) just after typing **dir** and before pressing RETURN. This causes everything that appears on the screen to be copied to the printer, and you must remember to press

```
A: INSTALL  COM : BRUN     COM : ACCOUNTS COM : UTILITY  COM : NOMINAL  COM
A: REPORTS  COM : STATMENT COM : COMPANY  DTA : CODELIST DTA : POST     COM
A: CONTROL  DTA : ALLOCATE COM : POSTINGS DTA : ACNTLIST DTA
```

Figure 2.2 A typical **dir** listing, in this example from the Sagesoft Popular Accounts program.

ALT-P again after the printing has been completed, otherwise every command will cause something to be printed. The loudspeaker will squeak when ALT-P is pressed to start printing, and will be silent when you press again to end printing. Remember that you need to feed paper into the printer, and to set items such as print quality before pressing the EXIT key to return to the computing actions. A directory display like this shows the files with their extension names but no dots. The files with the **.COM** extension are CP/M program files that will be used in the course of running a program. There are usually several of these, called up as required under the direction of the main program.

For more information on the contents of a disc, you can use an extended form of **dir**, but not quite so easily. The form is **dir**[**ful**], and the snag is that this is not a built-in action, but one that requires a program to be read from the CP/M System disc copy. This is straightforward if you want to show the results for the CP/M System disc, but not so easy if you want to show what is on another disc, unless you are using two drives. Whether you have one drive or two, the methods are the same. You place the CP/M+ disc in the drive or in Drive A if you have two drives, with the disc whose directory you need (if it is the correct type of disc) in Drive B. You then type: **b:dir**[**full**] and press RETURN. If you have two drives, the procedure is automatic, and you will see the full directory display appear. If you are using one drive, then you will be asked to change discs by a message that scrolls sideways at the bottom of the screen. Another option that you can use is to copy the DIR.COM file from the CP/M disc into the M drive, and use it from there by typing **M:** RETURN to select the M drive, and then **dir a:**[**full**] to request the directory. Either way, if you press ALT-P just before finally pressing RETURN, then you can get a printout of the full directory. Such a printout (for the Sagesoft Popular Accounts disc) is shown in Figure 2.3. The filenames are sorted into alphabetical order, read from left to right across and down the page. For each file, the number of bytes of storage space is shown in units of 'k', with k meaning 1024 bytes. A file that shows as 0k is, in fact, any short file whose name is entered in the directory but which has not been used. In this example,

```
Directory For Drive B:  User  0

    Name       Bytes   Recs   Attributes       Name       Bytes   Recs   Attributes
------------ ------ ------ ------------ ------------ ------ ------ ------------
ACCOUNTS COM    24k    190 Dir RW       ACNTLIST DTA     0k      0 Dir RW
ALLOCATE COM    16k    128 Dir RW       BRUN     COM    16k    122 Dir RW
CODELIST DTA     0k      0 Dir RW       COMPANY  DTA     1k      2 Dir RW
CONTROL  DTA     1k      6 Dir RW       INSTALL  COM     1k      8 Dir RW
NOMINAL  COM    27k    209 Dir RW       POST     COM    19k    147 Dir RW
POSTINGS DTA     0k      0 Dir RW       REPORTS  COM    22k    169 Dir RW
STATMENT COM    15k    117 Dir RW       UTILITY  COM    16k    128 Dir RW

Total Bytes     =    158k  Total Records =     1226  Files Found =    14
Total 1k Blocks =    158   Used/Max Dir Entries For Drive B:    18/   64
```

Figure 2.3 A full directory listing, obtained by using **dir b:[full]**. This is also from the Sagesoft Popular Accounts, and shows the files in alphabetical order.

DIR[ATT]	Shows attributes f1 to f4 (seldom used).
DIR[DATE]	Displays files that have date/time stamping.
DIR[DRIVE=ALL]	Shows files on all drives.
DIR[EXCLUDE]	Shows files that do not match specification.
DIR[FF]	Sends form-feed out at start of display and for each page, if LENGTH option used.
DIR[FULL]	Shows full information on file.
DIR[LENGTH=n]	Shows n lines of output before each heading – default is one screen.
DIR[MESSAGE]	shows drive names and user numbers, with 'file not found' message where no files exist.
DIR[NOPAGE]	Scrolls continuously.
DIR[NOSORT]	Shows files in order of finding them on the disc.
DIR[RO]	Shows only the read-only files.
DIR[RW]	Displays on the read-write files.
DIR[SIZE]	Shows the size of each file in k.
DIR[SYS]	Shows only the hidden SYS files.

Figure 2.4 The other form of DIR that require the DIR transient program to be read.

these are data files that will be used when the program runs. The number of bytes is important, because the total permissible for a single-density disc is 180k, and this particular example shows 158k of that space already used with programs. This is why it is important to place data on to separate discs.

Of the other information that the **dir[full]** display shows, the Recs (Records) number is less important – it relates to the way that data is grouped on the disc. For some discs, the Attributes columns may be important. For most of your discs, the Attributes column will show **RW** against each entry. This indicates read-write status, meaning that the file can be read or written. This in turn means that the files are unprotected, and can be altered by careless use. This is why you are advised to make a copy of these discs as soon as you receive them, and to use only the copies in day-to-day work. Though **dir** can be used in a vast number of other combinations, the straightforward **dir** and **dir[full]**, with or without another drive letter, form the main applications. In case you should need the other forms of DIR, they are listed in Figure 2.4.

Wildcards

At times, it can be a nuisance to type the complete name of a file. It's essential when you have to enter a filename for data, or when you want to recover data, but for some other purposes, of which DIR is one, the full name isn't necessarily needed. Suppose, for example, that each data file you had on disc used the extension letters DAT, and that these data files were mixed, for convenience, with program files on the same disc to avoid the need for disc changing. Now it's possible to use DIR and many other commands with shortened versions of the filenames, using what are called wildcards. A wildcard is a character that can take the place of another character or group of characters. To use the example, typing **dir*.dat** RETURN would give a display of all files that used the extension of DAT. This might include LIST.DAT, CASH.DAT, DEBT.DAT, CRED.DAT and so on. The asterisk in this example is a wildcard character that can stand in for any of these filenames, and only the **.dat** part is specific, resulting in only files with this extension being listed. You can, of course, use the wildcard asterisk character on either side of the dot, so that **dir*.*** will list all files, but since **dir** does this in any case there is little point. You might, however, have a set of files that were labelled ACCOUNTS.COM, ACCOUNTS. VDU, ACCOUNTS.PRP, ACCOUNTS.DAT and so on. You could list these selectively by using **dir accounts.*.** If these were the only files that started with the letter A you could also list them by using **dir a*.***, with the wildcard character substituting for the rest of the word. This would not be so useful if you had files AUDIT.COM, ADDIT.COM, ALPHA.DAT also on the disc, because these also would be listed. The question mark **?** also acts as a wildcard, but substituting for one character only. This makes it much less generally useful than the asterisk. Figure 2.5 shows examples of the correct use of the two wildcard characters.

File copying

The disckit utility of CP/M+ is very useful for copying the whole of one disc on to another (blank) disc. Another utility, called PIP, provides for copying individual files or groups of files as one of its many actions. The name is an acronym for Peripheral Interchange Program, which hints that it's use is for transferring data from one place to another. PIP is a program that has to be read in from the System disc (Side 2), and which can be used from then on, since it sits in the memory until you press ALT–C (ALT key and C together), or RETURN without having typed an instruction. Because PIP can be used in so many different ways, it's full description can be very intimidating, and so we'll look at the most common use, rather than explore every byway.

PIP is the method by which you can transfer any file from one disc to another, or to and from the M drive. By way of the wildcard characters, you

FILE.BA?	will get FILE.BAS, FILE.BAK, FILE.BAT etc.
F?LE.*	will get FILE.BAS, FOLE.TMP, FULE.COM etc.
F*BAS	will get FILLET.BAS, FILE.BAS, FOLLY.BAS etc.
***.?**	will get any file with a single letter extension.
.	will get any valid file on the disc.

Figure 2.5 Illustrating the correct use of the ? and * wildcards.

can make PIP work on more than one file, so that you can transfer a group of files with some common characteristic from one disc to another, of if you like, all the files on a disc. You start PIP by inserting the System disc (copy), Side 2, and typing **pip** RETURN. The disc will spin, and the presence of PIP is marked by the appearance of an asterisk, with a block cursor next to it. When this appears and the disc stops spinning, you can take out the System disc copy, and work with whatever discs you want to use. Suppose, as is usually the case, that you want to transfer a file called ACCOUNTS.DAT from one disc, the source disc, to a blank formatted disc, the destination disc. If you have two drives, you will place the source disc in drive A and the destination disc in drive B, then make sure that Drive A is engaged by typing **A:** RETURN. If you have only one drive, you will place the source disc in Drive A and make sure that this drive is engaged. You then type the command line, which for this example is:

```
b:=a:accounts.dat
```

and press RETURN. Note the syntax is **destination drive:=source drive:filename. extension.** If you are using twin drives, the process will be automatic, but if you are using only one drive, you will be prompted to put in the disc for Drive B. This means that the destination disc has to be put into the drive in place of the source disc. Once this has been done, pressing a key (it's a good habit to use the RETURN key for this purpose) will transfer the file. You will be advised by screen messages that copying is taking place, and the filename(s) will be shown as each file is copied. This is particularly important if you are using a wildcard in the filename. The action leaves the single drive labelled as B: however, and you should now type **dir** RETURN. This will bring up another prompt, asking you to place a disc in Drive A. Since you already have the disc that you want to check, the destination disc, in the drive, you simply press RETURN here, and you will see that the destination drive now has the file that you transferred.

The straightforward file-copying action is the main use of PIP for most purposes, because a lot of its alternative actions can be carried out in other ways. You can use this PIP action as described with wildcards, if you want to copy all the files with a specific extension, like *.COM, or even to copy all of the files on a disc, using *.* as the filename. You can use PIP, in particular, to copy files from disc to the M drive, or from the M drive back to disc. For example, to copy all .COM files from Drive A to the M drive, you run PIP, and use the command: **m:=a:*.com.** If the M drive does not have room for all the files, you will get the error message: DISK WRITE NO DATA BLOCK, accompanied by the name of a file, usually with the extension **.$$$.** This means that space has run out while the named file was being copied, and as a result the copying is not complete. This often happens when you try to use the M drive for programs, unless you can be selective and load in only the programs that you know you really need. Another way of filling the M drive, of course, is to copy programs or data into it when it already contains several files. The following chapter shows how the M drive can be cleared so as to allow more data to be stored in it.

To store files from the M drive to a formatted blank disc, you can invoke PIP as usual, and then type:

```
a:=m:*.*
```

showing wildcards being used in this instance so that all files are copied, since you would normally want to copy all files out of the M drive to a disc. Note that when you have used PIP to make a transfer like this, the PIP program is still in place at the end of the action. To remind you of this, the screen shows an asterisk in place of the usual A> or B> to indicate the drive when no program is in place. This means that you can carry out PIP actions one after the other, returning to the CP/M+ waiting state only when you have carried out all of the copying. To leave PIP completely, press RETURN by itself, or ALT-C.

Other pips

The PIP command of CP/M+ is a very extensive one, and it can perform tasks that for the most part only a professional programmer is likely to be interested in. This includes all types of file transfer from disc to printer, disc to screen, and disc to disc. One item that can sometimes be useful, however, is the ability to join two or more files into one single file. Suppose, for example, that you have two files called FILE1 and FILE2, and you want them joined into a file called JOINED. The simplest form of PIP command to perform this action is:

```
joined=file1,file2
```

which will join the files to make a file called JOINED on the same disc side. As before, you would need to have PIP loaded from the CP/M System disc before you changed discs and carried out this action.

The joining action can extend to more than one disc, however. If, for example, you used:

```
b:joined=a:file1,a:file2
```

then the two files would be read from Drive A and put together under the filename of JOINED to a disc in Drive B. If you use one drive only then the screen prompt line will tell you when to change discs. In addition, the files that you are joining need not all be on the same disc drive or disc side, but if you want to join a lot of files from different discs with only one drive you will have to carry out a lot of disc swapping, and you need to consider the form of the command carefully. For example, if you are using one single drive, the command:

```
b:cat=b:joined,a:bilge
```

will allow you to juggle with three discs, providing that you start in Drive A. When you press RETURN in this example, you will be prompted to put the disc for Drive B in place, and you can then put in the disc that contains the JOINED file. When this has been read, you will be asked to put in the disc for Drive A. you can then put in another disc that bears the BILGE file. This too will be read, and since the destination file is on a different letter, another drive change will be needed allowing you to use a third disc or side for the final file, CAT. The logic of this procedure is to have your starting drive A, usually after reading in PIP. You then make the files that are to be joined use the letters B, A, B... in sequence, starting with B and alternating. The destination file must then carry a drive letter which is not the same as the one used for the last of the joined files. This makes certain that the discs can be swapped again. If you use two drives, it can be more difficult to get the discs swapped around, and you might find it more useful to join the files in the M drive (if there is space), and transfer them in a separate command to the disc drive after you have changed discs.

Using SHOW

SHOW is another CP/M utility program that is concerned with files on the disc, but giving information that supplements DIR. To start with, you must have the CP/M System disc Side 2 in place when you type SHOW. If you simply press RETURN at this point, you will get a report on the System disc itself, showing that the files have read-write status, and that there is about 3K of space free on the disc. If you have at any point used a Drive B: command on a single drive system, or if you have a Drive B connected with a disc in place, you will also get a report for this drive. Similarly, if you have made any use of the M drive, the free memory here will also be reported. If you are using one drive only, don't be mislead by the report for Drive B, since this will refer to a disc that was at some earlier time placed in the drive.

The SHOW command can be used to refer to a disc other than the system disc, of course, but you must start the command with the System disc in the drive that you are currently using. For a single-drive system, this will normally be Drive A. A command such as:

```
show b:
```

will allow you to reveal the usable disc space on another disc side, either in Drive B of a twin-drive unit, or by swapping discs on a single-drive system. You can also display the M drive space by using **show m:** with the CP/M System disc in the current drive. This can be a useful way of deciding if a set of programs can be loaded into the M drive.

There are, as usual, several variations on SHOW that can be summoned up, using the same methods as we have used so far but with extension words

```
B: Active User :    0
B: Active Files:    0    1    2    3    8
B: £ of files  :    6    0    1   11    1

B: Number of free directory entries:           36
```

Figure 2.6 A SHOW display for a disc containing LOCO SCRIPT files. This shows the User (Group) numbers, and the pound sign has been printed in place of the hashmark to mean 'number'.

added in square brackets following SHOW. If, for example, you type **show a:[label]** and press RETURN, you will get a listing of any labelling of the disc. This refers to a disc title or other codings put on to the disc by using SET (see Chapter 8). Many of your discs will have no such label marking, and the result of **show a:[label]** will be an error message to the effect that no disc labelling exists. You would not normally use this variant on the SHOW command unless you thought that the disc had some sort of title name or password protection coded onto it.

A more useful variation, particularly for discs that may contain LOCO SCRIPT files, is the **show b:[users]** version. I have used Drive B here simply for illustration, because it's likely that you will want to have the System disc in Drive A, and use Drive B, or swap discs, for the disc side under investigation. Figure 2.6 shows the kind of display that you will get from a disc using, for this example, a disc of LOCO SCRIPT files. The new point to note here is the 'user number', which corresponds to the 'Group number' in LOCO SCRIPT. This is simply a convenient way of grouping files, but, as used in CP/M+, it allows a user to work with one set of files bearing one User number, with other files for other users not displayed in the directory unless a full directory has been requested. In this example, the **Active user:** refers to the current User number, which will always be zero unless you have done something to change it. The second line, labelled as **Active files:** shows what other User (Group) numbers are present on the disc. Since this has been a LOCO SCRIPT disc, several other group numbers have been used, and appear here as User numbers. Under each of these is the number of files in each group. Note that the character that appears on screen as the hashmark (#) is treated by the printer as the pound sign, £. This is a common problem, and it's made even more awkward by US books which refer to the hashmark as a pound sign. Printer control methods that allow you to alter this arrangement are detailed in Chapter 6.

The use of **show b:[dir]** will give a brief message on the number of free directory entries. This is useful to know if you have been recording a lot of short files under different names, because the number of directory entries is limited to 64. By listing the number free, you can see without having to count for yourself, how many more short files can be recorded. If you use longer files, it's more likely that you will run out of storage space for the data than directory entries.

```
       A: Drive Characteristics
   1,400: 128 Byte Record Capacity
     175: Kilobyte Drive  Capacity
      64: 32 Byte  Directory Entries
      64: Checked  Directory Entries
     128: Records / Directory Entry
       8: Records / Block
      36: Sectors / Track
       1: Reserved  Tracks
     512: Bytes / Physical Record
```

Figure 2.7 Using SHOW to give drive information – this has nothing to do with the disc that is in the drive.

Finally, there's a variant of SHOW that gives more technical information. Using **show b:[drive]** gives information on the type of *drive* that is in use, in this example, the B: drive. Figure 2.7 shows a typical example of the printout from this action, which simply states how many of each grouping of data can be accommodated on a blank disc. This tells you nothing about the disc that happens to be in the drive, only about the characteristics of the drive itself. It's useful if you have two drives with different characteristics connected – you should make a printout of the characteristics for each drive and pin it to the wall space behind the machine as a reminder.

Chapter Three
Disc Operations

File retitling

A common problem when you are working with a number of named files is that a file is recorded under the wrong name. A few commercial programs for the Amstrad machines allow you to rename a file without leaving the program, but in most cases this is something that has to be tackled by the CP/M+ utility command, REN. The REN action is sometimes a very vital one, because it's one way in which you can recover a file that you thought might be deleted. This can be very important, particularly in your early days as a computer user, on Monday mornings, Friday afternoons, Christmas Eve, and other times when your concentration on the computer may be less than 100%.

Suppose, for example, that we have a disc on which there is a file called JOINED.DAT. You have entered this name from a database program under the impression that you wanted to use it, but the name you really intended was JOINERS.DAT, because this file is of the names and telephone numbers of all the local joiners, sawyers and carpenters. Perhaps this might be another good place to remind you that if you use files like this, you might need to register under the Data Protection Act – consult your solicitor if in doubt. In any case, as you leave the program, you will have a data disc on which the file JOINED.DAT is present. There are two ways in which the REN command can be used, but the more convenient one is to place the System disc Side 2 in the drive, or Drive A, and type **ren** RETURN. You will then see the message:

```
Enter New Name:
```

and at this point you type the new name that you want, **joiners.dat**. When you press RETURN, you will then see a message:

```
Enter Old Name:
```

and you can now type the name **joined.dat** that you had used in error. Now make sure that the correct disc is in place in the drive, and press RETURN. When the A> prompt appears again, you can use **dir** to check that the renaming has been carried out.

Using this style of the command is more straightforward, and is better prompted than the alternative method, which is to type:

```
ren b:joiners.dat=b:joined.dat
```

with the syntax **ren newname=oldname**. As usual, the name can include a drive letter and any extensions that are needed. Strictly speaking, if both files are to be on the same drive, the drive letter need appear in only one filename, but it's safer to put the drive letter into each one if only as a reminder to yourself. You can also use the full name **rename** rather than the abbreviation **ren** if you want to. You can also use different drive letters on the two filenames, allowing you to transfer a file from one disc or side to another and rename it at the same time – but only if you have twin drives. If you use a single drive, then you have to carry out the renaming on one disc, and then use PIP to make the transfer. This is because a command such as:

```
ren b:crab=a:lobster
```

will cause the system to look for file **lobster** on the disc in Drive A right at the start, and since you need to have the System disc in place at the start, the file is not in place, and you get an error message. If you have used REN once and have not run any other command, however, REN is usually in place, and will run each time you need it without using the System disc. This still doesn't help, though, because REN simply doesn't allow for disc swapping.

One common use for REN is to recover a .BAK file. As you know, when a new file has been recorded with the same name as an old file, the old file is not completely deleted, it is simply renamed with a .BAK extension. If, for example, you recorded a new version of JOINERS.DAT on the same disc side as the older version, then the older version appears in the directory as JOINERS.BAK. Any file with a .BAK extension is normally ignored by programs – this corresponds to the use of LIMBO files in LOCO SCRIPT. To recover such a file, all you have to do is to rename it with a more suitable extension. For example, if the file has been JOINERS.DAT and has become JOINERS.BAK, then you will want to rename it with a .DAT extension. You should *not* try to rename it as JOINERS.DAT, however, because this is the name of the file that superseded it, so you need to choose another name, like CHIPPY.DAT. Once the renaming has been done, the file can be transferred to another disc, and if you need to use it on this new disc in its JOINERS.DAT form, then it can be renamed again.

One important feature of REN is that wildcards can be used. Suppose, for example, that you have a set of files on a disc, all with the .BAK extension. These could have been transferred by using PIP with a *.BAK filename. You can then use REN on these files, giving their filename as *.BAK and using the format:

```
ren *.DAT=*.BAK
```

to rename each file to a different extension. You cannot, of course, use wildcards for each part of a filename, because *.* would mean any file, and it would not make sense to rename any file to any name! The use of the wildcard in REN can be useful for recovering a group of BAK files like this.

Erasing files

Every now and again, files need to be erased. For business purposes, this should be a regular procedure, because discs that contains changing data should be copied at frequent intervals, certainly once a week if a lot of data is changed in a week. The principle is that if anything should happen to a disc, then a copy should be available, and if anything is wrong with that copy, then a backup copy, which might not be quite so up to date, is also available. The usual system is to keep three discs, labelled as Grandfather, Father and Son. The Grandfather disc is always the one with the oldest backup data. To illustrate how this works, imagine (Figure 3.1) that the system has just started, and a Father disc has been in use for a week, carrying the data files for a small

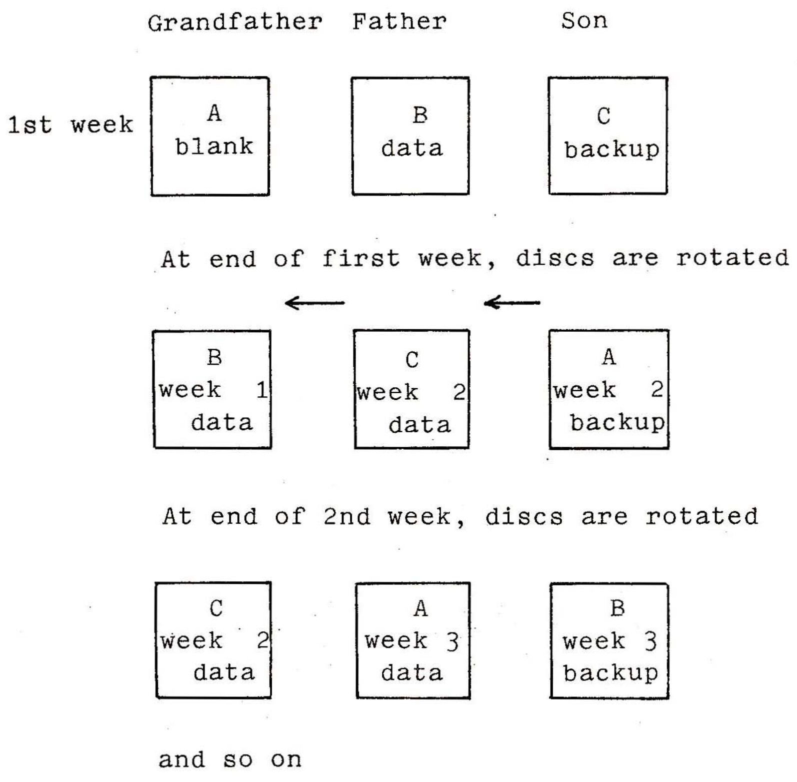


Figure 3.1 The Grandfather, Father, Son system of disc backup. This ensures that you have two discs containing current data, and one disc that is only one week old. The disc labels are rotated at the end of each week, and two discs are used for the next week's data changes.

firm, and being updated each day. Each day also, the data on this Father disc will be copied to a Son disc. In the next week, the discs are swapped round as indicated in the diagram, with the former Son disc being labelled as Father, the former Grandfather as Son, and the former Father as Grandfather. The data is once again recorded on to the Father disc, and the Son disc will again be used as a backup. By the end of the second week of the system, then, there are again two discs, Father and Son with fresh data, and Grandfather with data one week old. In the next week, the discs are relabelled. The former Son disc is now the Father, the Grandfather becomes the son, and the Father becomes the Grandfather.

A scheme like this works simply and easily when the number of data files is fixed, and when the same filenames are in use all the time, as in many accounts and database programs. For some other purposes, however, you may need to erase files from the Grandfather disc at intervals to avoid overfilling the disc. This applies only if the files can be erased, perhaps because they have been replaced or because data has been printed from them, or whatever. To erase the files, you type the command:

era filename

and press RETURN. You get no warnings, no prompts, no further messages, the file is simply erased, and there is no .BAK file to recover it from. It follows, then, that you have to think long and hard before you erase any file in this way. Though it *is* possible to recover information from an erased file, this is possible only if nothing else has been recorded on the disc, and if you have a suitable file-recovery program that you can use. CP/M+ does not come with any program of this type, but such programs can be bought and are available for CP/M machines that use 5¼-inch discs – it's only a matter of time before they become available on 3-inch discs.

The ERA command is another of the built-in commands of CP/M+, needing no System disc to be present in any drive when the command is used. The ERA command can use wildcards, so that the command **era *.bak** will, for example, erase all of the .BAK files on a disc. When the wildcard names are used with ERA, you get a warning reminder, asking for a Y/N confirmation that these files are to be erased. This is a method of reminding you that using the wildcard may erase more files than you intended. One useful application of the wildcard to ERA is the deletion of all files on a disc or on the M drive. The command **era *.*** will delete all files on the current drive, and by using **era m:\*.*** you can make sure that the M drive is specified. This action is often needed to make sure that the M drive is cleared to allow more files to be stored there.

The CP/M commands

CP/M commands are of two types, built-in and transient. The difference is important. The built-in commands (for the PCW 8256 and PCW 8512) are put into the memory of the machine at the time when you start up the CP/M operating system, and when you make use of them, you don't replace anything that is in the memory of the computer. On the older models, notably the CPC6128, these commands were permanently built in to the machine in the form of unerasable memory. The transient commands, by contrast, are carried out by loading a program (a 'command' file with the .COM extension to its name) into the main part of memory and running this program. This will replace anything else that happens to be in that part of the memory at the time, so that you have to be rather careful about how you use these commands. In particular, you cannot normally have a business applications program in the memory and then switch to using a CP/M transient command unless you have a copy of the program on disc with all of its data recorded. Since you will not normally use CP/M transient commands along with business programs, this is not quite such a problem as it might seem. We'll look at the transient commands of CP/M later in this chapter, but for the moment we'll concentrate on the built-in commands, which are listed in Figure 3.2. Each of these will be obeyed at once when its name (and any other data that is needed) is typed, followed by RETURN. Though these commands are built-in, there are versions of them, like **dir[full]** which are transient. Of these commands, you will by this time have used DIR to obtain the CP/M directory, REN to rename a file, and ERA to erase files. The USER command has already been mentioned, and the TYPE command is dealt with in Chapter 4.

The transient commands

As we saw earlier, a transient command is one which is kept stored as a file on disc, and it will be loaded and run only when called by typing its name followed by RETURN. All transient programs are stored in the memory of your computer starting at the same place each time (memory location number

DIR	Print directory
DIRS or DIRSYS	Print directory of SYS files
TYP(E)	Print out named ASCII file
ERA	Erase named file
REN	Rename file
USE(R)	Enter user number
n:	Select drive n

Figure 3.2 The built-in commands of CP/M. Some, like DIR, also exist as transient programs that have to be read in from the Master disc each time they are needed.

```
Directory For Drive A:  User  0

    Name       Bytes   Recs   Attributes      Name       Bytes   Recs   Attributes
------------ ------ ------ ------------ ------------ ------ ------ ------------
DIR      COM     15k    114 Dir RW     DISCKIT  COM     7k     56 Dir RW
ED       COM     10k     73 Dir RW     ERASE    COM     4k     29 Dir RW
LANGUAGE COM      1k      8 Dir RW     PALETTE  COM     1k      8 Dir RW
PAPER    COM      2k     16 Dir RW     PIP      COM     9k     68 Dir RW
RENAME   COM      3k     23 Dir RW     SET      COM    11k     81 Dir RW
SET24X80 COM      1k      8 Dir RW     SETDEF   COM     4k     32 Dir RW
SETKEYS  COM      2k     16 Dir RW     SETLST   COM     2k     16 Dir RW
SETSIO   COM      2k     16 Dir RW     SHOW     COM     9k     66 Dir RW
SUBMIT   COM      6k     42 Dir RW     TYPE     COM     3k     24 Dir RW

Total Bytes     =     92k  Total Records =     696  Files Found =    18
Total 1k Blocks =     92   Used/Max Dir Entries For Drive A:    27/  64

Directory For Drive B:  User  0

    Name       Bytes   Recs   Attributes      Name       Bytes   Recs   Attributes
------------ ------ ------ ------------ ------------ ------ ------ ------------
DATE     COM      3k     23 Dir RW     DEVICE   COM     8k     58 Dir RW
DUMP     COM      1k      8 Dir RW     GENCOM   COM    15k    116 Dir RW
GET      COM      7k     51 Dir RW     HEXCOM   COM     2k      9 Dir RW
INITDIR  COM     32k    250 Dir RW     LIB      COM     7k     56 Dir RW
LINK     COM     16k    123 Dir RW     MAC      COM    12k     92 Dir RW
PALETTE  COM      1k      8 Dir RW     PATCH    COM     3k     20 Dir RW
PUT      COM      7k     55 Dir RW     RMAC     COM    14k    106 Dir RW
SAVE     COM      2k     14 Dir RW     SID      COM     8k     63 Dir RW
XREF     COM     16k    121 Dir RW

Total Bytes     =    154k  Total Records =    1173  Files Found =    17
Total 1k Blocks =    154   Used/Max Dir Entries For Drive B:    21/  64
```

Figure 3.3 The transient commands of the Amstrad version of CP/M+. These consist of all of the COM files, excluding BASIC.COM. Some of these files are not standard CP/M files, and are intended for the PCW machines only.

256), and will wipe out anything else which has been stored starting at this address. Figure 3.3 is a list of the commands which come under this heading. Of this list, you have certainly used **disckit**, which carries out the formatting and disc copying actions, and PIP for copying files. A fair number of these transient programs are highly specialised, and we shall deal with a few of them in the following chapters. Figure 3.4 shows a shorter list of the transient programs that are likely to come in for most use. When you want to make use of these programs, you must have the appropriate side of the System discs in the current drive, and you need only type the name of the transient, then press RETURN. All of these program names have the extension of **.com** but this does not need to be typed in order to run the programs.

The **disckit** program is the one that we have used for formatting and copying. It has a third use, which is to check for errors after copying a disc. This is never needed in the normal course of events, but the basis of some protection systems is to cause disc errors when a copy is being made, so that the 'Verify' action can sometimes be useful if you suspect that a copy might be incorrect. It might seem odd to see **rename**, **erase** and **dir** appearing also among the transients, but these are the specially extended versions of the commands, such as **DIR[FULL]**. An even more useful action is **erase*.*[confirm]** which will erase all files, but asks for confirmation first. It's

DISCKIT	The essential copiet, formatter and checker.
PIP	For file copying.
DIR	The extended directory program.
ERASE	The extended erase utility.
RENAME	The extended rename utility.
ED	Text editor for small files.
SETKEYS	Key redefinitions.
SETLST	Printer control.
SUBMIT	For running programs in sequence from one command.

Figure 3.4 A short list of the most useful of the transient programs, the ones for everyday use.

particularly useful if you want to delete most of the files on a disc surface, but to preserve a precious few.

We have by now used examples of both the built-in and the transient commands so that you can now appreciate what the differences are, and how the programs must be used. From this point on we shall be dealing mainly with the transient commands. Note that when you load in any other programs, such as accounts or database programs, the programs normally use the same part of the memory as the transient CP/M utilities, so that you can't expect to make use of a business program and a CP/M utility at the same time. A few utilities load in to the normal part of memory and then shift themselves so that they *can* be used along with other programs. It's unlikely, however, that you will want to get involved with this type of action unless you become heavily hooked on programming with CP/M codes.

CRTL and ALT

In several places in the manual, and in many programs, you will read of the CTRL key being used along with other keys. This key exists on the CPC6128 and on most other computers, but on the PCW 8256 and the PCW 8512, this key has been renamed ALT. Because some manuals do not mention this change, confusion can be caused – old programmers may not have noticed because the key is in the same place and is hit by the same finger! The effect of this key is to turn all the other keys of the keyboard into command keys, by pressing the ALT key along with one of the other keys. When these combined keys are pressed, there is no letter printed on the screen and no effect on any waiting action, so that it's possible to press ALT and a key when you are halfway through typing an instruction, for example, or just before pressing RETURN on a command of any kind. The two main ALT key uses are the ones we have met already – the use of ALT-P (press ALT and P keys together) to switch the printer on and off, and the use of ALT-C to leave the PIP program. The use of ALT-P is what is called a 'toggling' command. This means that pressing the keys once will switch the effect on, and pressing again will switch the effect off again. For ALT-P, the loudspeaker sounds a warning and the cursor flashes once when the printer is being switched on, but it's only too easy to forget that it is on since there is no continuing reminder. You will sometimes meet these commands printed in the form ^P, because on the computers that use the CTRL key, the ^ character is printed when this key is pressed by itself. If you see anything on the screen when you have pressed ALT along with another key, however, something has gone wrong because this key does not normally make any character appear, nor does it move the cursor position when used in CP/M. The following is a list of the more useful ALT key actions, and a summary will be found in Appendix A. Many of these can be quite useful for altering a command line without having to delete the whole line, but on the Amstrad machines, most of the actions are also covered by other single keys. For that reason, actions such as ALT-A, which has the same action as the cursor-left key, and ALT-F, which has the same action as cursor-right, have been omitted. The remaining set will sometimes be referred to in instructions for programs, and have been listed even when their effect can be duplicated by other keys. The reason for these duplications is that the CP/M system was devised a considerable time ago; computers then had little more than a set of typewriter keys and a CTRL key. Modern machines have added several keys whose actions duplicate many of the ALT key actions, but the original set are still provided so as to make the system more truly universal. You might find, for example, that if you use several computers, that ALT-A is more convenient to learn then the use of the cursor-left key, because not all computers have the cursor-left key. In addition, many programs will use the ALT-key combinations in different ways, specific to the program.

ALT-C is used to stop a program running. This applies mainly to the transient CP/M command programs, because some commercial business software will disable this key action. This is done to ensure that you can leave a program only by the correct route, recording any data that has been entered.

ALT-E will cause a new line to be selected on the screen, but without causing the effect that pressing RETURN would have on a CP/M+ .COM program. Its main use is in preparing short sections of text with PIP or ED (see Chapter 5).

ALT-H has the same action as the DEL-left key of the PCW machines, so that this key combination is not needed.

ALT-I has the same action as the TAB key, and is not therefore needed.

ALT-J generates a new line and ends any input, mainly used with the ED transient program.

ALT-M is a combination that has the same effect as the RETURN key, and so is seldom used (but see Chapters 6 and 7 for SETLST and SETKEYS files).

ALT-P toggles the printer on and off.

ALT-Q starts screen scrolling. This reverses the effect of ALT-S and allows the screen display to scroll again. The use of ALT-S and ALT-Q in succession can be very useful for reading a file that consists of ASCII codes and is read using TYPE (see Chapter 4).

ALT-R makes a copy of whatever you have typed up to that point, and places the copy on the next line. The original line end is marked with a hashmark. Used in some editing actions.

ALT-S stops screen scrolling, and is particularly useful when used to examine long sections of text that have to be read from the screen. Scrolling is restarted by using ALT-Q.

ALT-Z is used in the text-editing utility, ED, to signal the end of an input.

Printer use

The use of the Amstrad printer is so easy and automatic with LOCO SCRIPT that you tend to assume it will be as simple with CP/M. In one sense, it is, because the use of ALT-P to switch the printer on and off is a very simple and flexible way of using the printer. It is not so easy, however, to exercise the same control over what is printed. We have already seen an example of this with the hashmark, #, being printed as a pound sign, £. It *is* possible to gain the same control over the printer from CP/M programs as you have with LOCO SCRIPT, but to do so involves writing a printer control program, something that we shall look at in Chapter 6. This is not always necessary, however. Many business packages, such as the Sagesoft series, will include a printer-setting program in their package, and this will operate automatically with no attention needed on your part. The snag here is that this may *not* be what you want! If, for example, you need to make your accounts up in dollars, as is possible for an importing business, then you may find that being unable to put in a dollar sign is, to say the least, a considerable frustration. The advice in Chapter 6 is therefore likely to be useful for a number of readers, even when very complete commercial software is being used.

For the more ordinary use of the printer, the procedure is much the same as it is when LOCO SCRIPT is being used. When you load the printer with paper, the bottom line of the screen indicates the available options. The choice is made in the usual way by moving the cursor, using the left and right arrowed keys, and then pressing the [+] or [–] keys to make the choice. The action is then confirmed by pressing the EXIT key. If you need to alter settings in the middle of a page, then the printer menu can be summoned up again by using the PTR key. These actions are, however, very limited. There is no provision for underlining, for bold or italic print, for altering the print size or any of these features that you become so accustomed to with LOCO SCRIPT. Once again, all of these things can be done, but not in a simple straightforward way unless the program that you are running provides for them.

Chapter Four
Instructions and Text

SUBMIT files

One of the files on the System disc is SUBMIT.COM, and its use is very widespread in commercial software. The SUBMIT.COM program exists in order to run a SUBMIT file, which is, in fact, a list of programs that have to be run or actions to be carried out. Suppose, for example, that you wanted to ensure that when you made use of a program on a disc, the printer action was altered so as to allow twelve characters per inch of bold type with double line spacing, and the keyboard was altered so that the ENTER key was disabled. CP/M programs need to be written to carry out these actions, so that the necessary programs (SETLST and SETKEYS) would need to be placed on the same disc. In order to make everything happen in the correct order, a SUBMIT file is then written, containing, in order of use, the programs that must be called. In this example, you would probably use the programs SETKEYS, SETLST and then the main program that you intended to use. When you started to use the disc, then, the keyboard and the printer would be set up for your use, and then the main program would start, all with one command. This saves having to use the alternative of typing the name of each program, and pressing RETURN, in the correct order. The actions can even be carried out in such a way that the system is auto-starting, so that the sequence in the SUBMIT file will run when you put the disc in place after switching on or resetting. This requires the main CP/M program with the .EMS extension to be present, and also a file called PROFILE.SUB. We shall look at PROFILE.SUB files later in this Chapter.

A SUBMIT file is any set of program names or commands that bears the extension letters SUB. This means that you should avoid using these particular letters for anything else, since the machine could tie itself in a considerable knot if you used the SUB extension for some file of data.

We'll leave for the moment the problem of how a SUBMIT file is created, because that requires a text-editor program, and the one that is supplied with CP/M+, called ED, is not by any means simple to use. You *cannot* use LOCO SCRIPT to make SUBMIT files, because LOCO SCRIPT files are not of the correct form to be read by the CP/M+ operating system. This is because the LOCO SCRIPT files always contain non-alphabetical characters, the codes for things like line length, print setting, line spacing, underlining, bold type and so on. Because these codes are concentrated in the header, it's usual to find that CP/M cannot read this part, and never gets to the rest of a LOCO SCRIPT file, as we can demonstrate later.

Suppose, for example, that you used ED to create a file called TEST.SUB and it consisted of the two lines:

```
dir
listdat.com
```

and you recorded this file. From then on, by commanding **SUBMIT TEST.SUB**, you would make the machine produce a directory, then load in a program called LISTDAT.COM and run it. This is the simplest application of SUBMIT, and the actions can consist of:

1. any legal command of CP/M+
2. any legal command of CP/M+ combined with the 'submit parameters' of $0 to $9 (illustrated later)
3. any line of valid input data
4. any line of valid input data with 'submit parameters'.

Each command line must consist of not more than 128 characters, which is a generous allowance when you consider that filenames are of only eight characters. The lines of commands are carried out strictly in order, and this, of course, is the main reason for using SUBMIT, to reduce the amount of typing effort that would otherwise be needed in entering a sequence of commands that is often repeated.

Like all of the CP/M+ commands, though, SUBMIT has hidden strengths. Suppose that when you start work on a disc, you want to rename a file, erase an old file and then call up a database program. The filenames for renaming and for erasing are not fixed, so that they have to be entered. The normal run of events would be that you typed something like:

```
ren newname=oldname RETURN
era backfile RETURN
fildat RETURN
```

which is a fair bit of typing and RETURN bashing to do first thing in the working day. Now this is where these 'parameters' of SUBMIT come in handy. A parameter is what an Irish friend calls a variable constant, meaning that it's something that is fixed for a time (in a SUBMIT file in this example), but whose value can be changed when it needs to be. It's easier to illustrate than to define – suppose that you had a SUBMIT file that read:

```
ren $1=$2
era $3
fildat
```

and this was recorded under the name of HELLO.SUB. In this file, the $1, $2 and $3 are parameters, they are holding a place for names that you will put in when you call up the SUBMIT file. This you would do by typing:

SUBMIT HELLO NEWNAME OLDNAME BACKFILE RETURN

and the effect is to load in the SUBMIT file HELLO, use NEWNAME in place of $1, OLDNAME in place of $2, and BACKFILE in place of $3. It's a very ingenious use of a single command line, and it allows much more flexibility in the use of a SUBMIT file than would otherwise be possible.

A further facility of SUBMIT is that you can put both program names *and* directions into the file. For example, there would be little point in putting **pip** into a SUBMIT file unless you could also specify the files on which **pip** was to operate. These can be put in preceded by the < sign. For example, you can have in your SUBMIT file:

```
pip
<b:=*.bak
<
dir b:*.bak
```

which will cause **pip** to be loaded, then copy all of the BAK files to Drive B, then show the directory for Drive B. The < sign in this application carries out the task of the RETURN key, which is why there is a < in a line of its own following the input to PIP, and used to leave the PIP program.

You may at times need to put into a SUBMIT file some commands that are carried out on the keyboard by using the ALT key plus a letter key. One obvious example is the use of ALT-P for printing data. To make use of such characters in a SUBMIT file, you can use the up-arrow character which is obtained by using the EXTRA key along with the semicolon or letter U key on the PCW machines. For example, if you wanted to print a directory with a SUBMIT file, your SUBMIT file would be:

```
^p
dir
^p
```

so as to switch on the printer, print the directory, which also appears on the screen, and then switch off the printer. The procedure is straightforward enough provided that you know how to get the up-arrow symbol from the keyboard. Using the ALT key during the creation of a SUBMIT file has no effect, since the text creating program ED does not use this key to make any mark on the screen, only as a control key.

Finally, you can use SUBMIT by itself, and you will get a prompt asking you to name the file that is to be submitted. This allows you to keep your copy of SUBMIT.COM on one disc along with other utilities, and operate on a SUB file that is held on another disc. At times, you may find that SUBMIT, either used by itself or with a filename has no effect. One thing that can happen is that after you press RETURN on SUBMIT, the prompt sign >

returns instead of the 'Enter File to SUBMIT:' message. Another problem that is encountered is that you type SUBMIT, space, filename, and on pressing RETURN the disc spins and the cursor moves to the next line on the screen, but nothing else happens. This is usually a lock-up, and no key will from that point have any effect on the machine. You will have to remove the disc(s), switch off, and start all over again loading in the CP/M system and the files. Usually when this kind of thing starts to happen, the SUBMIT.COM program on your disc has become corrupted, usually by having the disc in place when you switched off the computer. This is why it's so important always to work with copies of the System discs, and never with the originals except for copying purposes. If you find that your SUBMIT file is giving trouble, erase it and use PIP to place a new copy on the disc.

PROFILE.SUB

The PROFILE.SUB file is a special and very useful feature of CP/M. PROFILE is a form of SUBMIT file, but with the difference that you never have to type SUBMIT. The initialisation of CP/M with the .EMS file includes a search for a file called PROFILE.SUB, and if this is present on the disc, the commands in the file, which is a normal SUBMIT type of file, will be executed as part of the switch-on process. This is possible *only* if the disc is write-enabled, so that no PROFILE.SUB file can be used with the System disc, only with write-enabled copies. You *must* have the main CP/M program called J11CPM3.EMS (or J14CPM3.EMS) also on the same disc, since this is the program that is read into the memory to make the CP/M system operate. On the System disc Side 2, there is a PROFILE.ENG file which is designed to be used with your system. The contents of this file are illustrated in Figure 4.1. The first line is a set of instructions regarding searching for files. The effect of this is that files should be looked for in Drive M first and, if they are not

```
setdef m:,* [order = (sub,com) temporary = m:]
pip
<m:=basic.com[o]
<m:=dir.com[o]
<m:=erase.com[o]
<m:=paper.com[o]
<m:=pip.com[o]
<m:=rename.com[o]
<m:=show.com[o]
<m:=submit.com[o]
<m:=type.com[o]
<
```

Figure 4.1 The contents of the PROFILE.ENG file that is included on the Master disc. It cannot be run from the Master disc, only from a write-enabled copy.

present, on whatever other drives are connected. This part is dealt with by the **m:,*** portion of the command. The items enclosed in square brackets request the system to look first for any SUB files and, only after they have been executed, to look for COM files. Any temporary files are to be placed in the M: drive. What follows is then more familiar by this time. PIP is being used to transfer files into the M drive, the [o] at the end of each filename is a command that requests the system to ignore any ALT-Z symbol at the end of the file. The files that are loaded in this way are ones that you might want to use if you were carrying out your own programming using the BASIC language. This is not very likely if you are using the machine for business with purchased programs for CP/M, so it's likely that you will want to make a new PROFILE.SUB for yourself, using the commands of ED that are listed in the following chapter. Before we get to that stage, though, we need to take a look at a useful method of finding what is contained in the SUBMIT and PROFILE type of files.

The TYPE utility

The ability to read a file is something that is rather limited unless you have made some study of programming CP/M system machines. This doesn't mean that you can't find out the contents of a file, nor is it intended as a discouragement. What I mean is that many files will be unintelligible, because they consist only of coded numbers that give you nothing on the screen that you can understand unless you know the system. With that proviso, then, let's see what we *can* do. The main file-reading utility is TYPE, and it needs a filename following it, with a space between TYPE and the name of the file that is to be typed. The effect will be to print to the screen (or to paper if you have used ALT-P) the characters of the file that you specify. The snag is that the file must consist wholly of printable characters! This will be true of SUB files, and of some files with extension names of TXT, but certainly not of COM files, and probably not of several others. Even the files that have been created by LOCO SCRIPT cannot be read correctly by TYPE because of the number of 'special characters' that are embedded in them. The clue to all this is the use of ASCII codes.

The word ASCII is an acronym made up from the letters of American Standard Code for Information Interchange, and we should be glad of it because it's one of the precious few things that is standardised in computing. The code, Figure 4.2, specifies what numbers shall be used for each letter of the alphabet, digit, and punctuation mark. In the main, these codes are widely observed, but some characters have no code allocated. This is because ASCII is an American code, and since the US does not require a '£' sign, there isn't one, nor are there the accented letters for Continental alphabets, nor many of the Greek letters that are used as mathematical symbols and so on. It follows, then, that though the bulk of the ASCII code system is used as standard,

32		33	!	34	"
35	#	36	$	37	%
38	&	39	'	40	(
41	)	42	*	43	+
44	,	45	-	46	.
47	/	48	0	49	1
50	2	51	3	52	4
53	5	54	6	55	7
56	8	57	9	58	:
59	;	60	<	61	=
62	>	63	?	64	@
65	A	66	B	67	C
68	D	69	E	70	F
71	G	72	H	73	I
74	J	75	K	76	L
77	M	78	N	79	O
80	P	81	Q	82	R
83	S	84	T	85	U
86	V	87	W	88	X
89	Y	90	Z	91	[
92	\	93	]	94	^
95		96	`	97	a
98	b	99	c	100	d
101	e	102	f	103	g
104	h	105	i	106	j
107	k	108	l	109	m
110	n	111	o	112	p
113	q	114	r	115	s
116	t	117	u	118	v
119	w	120	x	121	y
122	z	123	{	124	|
125	}	126	~	127	

Figure 4.2 The standard ASCII codes as they are printed from a dot matrix – some characters appear differently on the screen.

certain codes are often re-allocated to suit the country in which the computer is used or manufactured. You will already have noticed the problem with the # and the £ signs on the printer. The range of standard ASCII code numbers is 32 to 127, and if a file has been recorded that consists entirely of these numbers, plus the use of 13 for the RETURN key action, then this file can be printed by the TYPE utility without problems. If, however, the file contains certain codes in the region 0 to 31, or codes 128 to 255 (the upper limit), then the effect on TYPE can be to print nothing, or to terminate printing, giving you the impression that there is nothing in the file. All of this is quite deliberate – it would be possible to make a version of TYPE that printed the printable letters and put spaces or dots in place of the others, but TYPE is intended for working with files of purely ASCII coded material. A lot of word-processing programs can create such files, and there is a file of this type on Side 4 of the System discs, the HELP.HLP file. This is not entirely ASCII, but the only non-ASCII codes cause no problem other than causing the beep to sound at intervals.

You can, of course, use TYPE to look at the content of the PROFILE.ENG file on the System disc Side 2, and you will find other SUB files to look at in your System discs. Even more interesting is to look at the SUB files on any commercial software that you have, or at .MSG files which carry the warning and other messages of the programs. Several commercial programs have a PROFILE.SUB that places program files into Drive M so that they can be called up quickly to occupy the transient file space. TYPE action includes paging, meaning that the text will fill up the screen, and then halt until you press a key to continue. You can avoid this by using the command in the form TYPE **filename** [**nopage**], which will allow the display to scroll continuously. This can be useful if you want a quick preview of a long file. During such a display, you can always press ALT-S to stop the scrolling, and the ALT-Q when you want to start it again. You can escape from the TYPE command by using ALT-C. The [**nopage**] option can be useful if you have continuous stationery loaded into the printer and you want to print the whole file (usually ALT-P just before you press RETURN on the TYPE command). You can ensure paging by typing TYPE **filename** [**page**].

Using DUMP

TYPE is a built-in utility, and one that is convenient for the type of ASCII-coded file that is used for SUBMIT and other purposes. There is another utility, DUMP, that can be used for files that contain non-ASCII codes, or a mixture of ASCII and non-ASCII. This utility allows you to look at the content of .COM files, but you are unlikely to get much useful information from such files. More usually, DUMP allows you to look at files created by LOCO SCRIPT without the need to switch over to LOCO SCRIPT. These files, however, are not shown in the form that LOCO SCRIPT uses, so don't expect to see them with lines arranged as they were when they were typed.

Figure 4.3 shows the result of using DUMP on part of a LOCO SCRIPT file. This is by no means such a straightforward exercise as you might think, and the setting up involves one utility, USER, that we have noted already. In the normal run of events, you would type something like:

dump b:textbit

and then put the System disc Side 3 into the drive, press RETURN, and change to the disc with the text on it when prompted. This is the normal and

```
CP/M 3 DUMP - Version 3.0
0000: 4A 4F 59 01 01 54 65 6D 70 6C 61 74 65 20 20 20  JOY..Template
0010: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0020: 20 20 20 66 6F 72 20 6C 65 74 74 65 72 73 20 6F     for letters o
0030: 6E 20 70 6C 61 69 6E 20 41 34 20 70 61 70 65 72  n plain A4 paper
0040: 20 6E 6F 20 73 70 65 63 69 61 6C 20 63 6F 6E 74   no special cont
0050: 69 6E 75 61 74 69 6F 6E 20 73 68 65 65 74 73 05  inuation sheets.
0060: 0A 2E 7F FF 00 00 00 00 00 8C 00 01 00 01 00 FF  ................
0070: 0C 0E 01 00 00 1A 00 0C 86 80 0D 00 27 00 02 02  ............'...
0080: 00 48 02 00 0A 52 06 00 00 00 0E 12 16 1A 1E 3C  .H...R.........<
0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0100: 05 00 81 00 00 00 00 00 0A 80 C8 82 00 89 00 0A  ................
0110: C8 00 88 08 86 00 00 89 07 47 A4 05 81 83 03 79  .........G.....y
0120: 6F 75 72 81 6E 61 6D 65 84 03 88 02 86 00 00 89  our.name........
0130: 07 44 6C 05 81 83 03 79 6F 75 72 81 61 64 64 72  .Dl....your.addr
0140: 65 73 73 84 03 88 02 86 A0 05 88 02 86 00 00 89  ess.............
0150: 07 4C 09 06 81 83 03 64 61 74 65 84 03 88 02 86  .L.....date.....
0160: A0 05 88 02 86 A0 05 88 02 86 A0 05 88 02 86 A0  ................
0170: 05 88 02 86 F4 04 83 03 61 64 64 72 65 73 73 65  ........addresse
0180: 65 84 03 88 02 86 A0 05 88 02 86 8C 04 44 65 61  e............Dea
0190: 72 81 83 03 81 81 81 81 81 81 81 81 81 84 03 88  r...............
01A0: 02 86 A0 05 88 02 86 DA 00 42 65 66 6F 72 65 81  .........Before.
01B0: 75 73 69 6E 67 81 74 68 69 73 81 74 65 6D 70 6C  using.this.templ
01C0: 61 74 65 81 64 65 6C 65 74 65 81 74 68 65 81 66  ate.delete.the.f
```

Figure 4.3 The action of the DUMP utility on part of a LOCO SCRIPT file. The numbers on the left-hand side form the coding of the file.

straightforward use of DUMP, but it's not always possible to use if you want to look at LOCO SCRIPT files from CP/M+, because of the grouping of LOCO SCRIPT files. In the illustration, the LOCO SCRIPT file was one that was in Group 3, corresponding to User 3 in CP/M+. This means that two changes have to be made before the dump can be carried out. One is that a copy of DUMP.COM has to be put on to the same disc as contains the LOCO SCRIPT files. This is not absolutely essential if you have twin drives or if you put DUMP.COM into Drive M, but it does form the simplest method of dealing with this. You then insert the System disc Side 2 and type:

set dump.com[sys]

which will allow the utility DUMP.COM to be loaded no matter which Group (User) number happens to be in use. Having done this, you type **user 3**, to gain access to Group 3 files of LOCO SCRIPT, and you can then use the command:

DUMP LETTER.PLP

for the particular file that happens to carry that filename. Note that it's essential to give the full filename, including extensions in these two latter commands. If you get a 'No file' error message when you know from a DIR listing that the file *does* exist on that disc side, then it's almost invariably because you have either misspelled the name of the file or have omitted its filename extension. In general, the only time you can be certain of being able to omit an extension is when you are calling a COM program by name.

The dump of the file in Figure 4.3 is of one page only, enough to get the taste of the system. The important part from your point of view is the right-hand side, which shows the characters that correspond to valid ASCII codes. On this section, you will see a lot of dots. Some of these will correspond to full-stops in the text, but most of them are placed where the file contains non-ASCII codes, the control codes of LOCO SCRIPT. The large number of these special codes illustrates why the TYPE utility is unable to make any sense of LOCO SCRIPT files. The main part of DUMP, however, consists of the numbers and letters, starting with a four-character group in each line, and with sixteen two-character groups in each line. These are the actual bytes of data in the file, expressed in a number-code called hexadecimal. This is a form of writing numbers that uses a scale of sixteen rather than the familiar scale of ten, and its advantage is that each byte of data can be expressed as a two-digit number. The letters A to F are used in place of the numbers 10 to 15 in this scale, which is why you see letters cropping up where you don't expect them. The four-digit group at the left-hand side is a total number count for the bytes in the file. Unless you are bitten very deeply by the desire to program for yourself, you don't really need to know anything about hexadecimal. If the urge should get to you, then you might like to look at my book: *Introducing Amstrad CP/M Assembly Language*, but if you don't happen to be going down that particular road, then only a few points about this number display are of real use to you. One is that the number 20 (two sixteens and a zero, equal to 32 in ordinary numbering) is the ASCII code for a space. The other is that code 2E is the real full-stop (period), 0A is a line feed (usually causing a carriage return as well), and 0D is a carriage return by itself. The ASCII code numbers are those whose first digit is anything between 2 and 7 inclusive. The dots that you see separating the words in this example use the code 81, which is the special LOCO SCRIPT separator.

There is another way of displaying the text of a LOCO SCRIPT file without showing these special characters, but the display is definitely off-putting. If you want to see it, put System disc Side 2 in place, and type, using whatever name corresponds to your own text file:

```
pip con:=letter.plp[g3fo]
```

and press RETURN. Note that the last letter between the square brackets is letter oh, not number zero. When you press RETURN after this, the PIP utility will load in and if you are using a single drive, you will be reminded to change discs; if you have twin drives then the file with the text should be in Drive B. The result will be a display in which every non-ASCII character causes a shape of some sort on the screen. There are no numbers in hexadecimal this time, and you might find the text easier to read. It's not so easy to print this lot, because the non-ASCII characters that appear on the screen do not get to the printer, with the result that all of the text appears joined together, and the printer form-feeds (so needing a new sheet of paper) at intervals. This is because the CP/M+ set-up for the printer is not the same as that for LOCO SCRIPT, and because the PIP utility does not treat the character codes in the file in the same way as LOCO SCRIPT does. Other uses of PIP are dealt with in more detail in Chapter 8.

Chapter Five
Creating Text

Using ED

The ED.COM file in your CP/M System disc, Side 2, is a text editor which can be used for creating, editing and saving ASCII text on to disc. There are no restrictions about how this text editor can be used, though its main use is in writing programs for CP/M . COM use. It is intended, in other words, to be used by professional programmers, who use it to create files of commands in what is called assembly language that can be transformed into CP/M+ .COM files. Another use is in making files that will be used by other utilities such as SETKEYS and SETLST, as we shall see in Chapter 6. You can also use ED to prepare files for other languages, however, such as Pascal and C, provided that these come in versions which can use the ED files. You can even use ED as a general form of text editor to prepare text which can later be incorporated into your own programs. Its main use for most PCW applications is in making files for key definition, printer initialisation, and SUBMIT actions, of which more later. If you find that ED is likely to be useful to you, it's a good idea to make a copy of ED, along with the CP/M system, on several disc sides. As before, you can use DISCKIT and PIP to make whatever copies you need. I must emphasise once again, however, that ED was designed as a tool to be used by the professional programmer. It lacks, therefore, the helpful features of more modern programs that are designed for the computer-user as distinct from the computer programmer, and until you get into the way of it, it can seem positively user-hostile. Unless you have an alternative, and LOCO SCRIPT isn't one, for creating such files, you are stuck with ED, and you simply have to learn its little ways. If you are likely to need to prepare a lot of text files such as SUBMIT files, and files for other languages, or for CP/M, then I can strongly recommend the Hisoft text editor ED80, which is very much easier to use. The difference between the two is that ED80 allows screen editing, so that you can see at all times what the effects of your editing commands are. This is not exactly something that could be proclaimed as an outstanding feature of ED.

When you call up ED to create or change a text file, you can follow the name ED with a space, and then with the filename that you want to use. If you have a text file on the disc, and you want to edit it further, then you will use this filename following ED and the file will be obtained and presented ready for use. Notice that this can be done only if you have ED on the same disc as the file, or if you have a second drive and can specify the second drive letter in the filename. With a single drive, you *can't* type ED, then a filename, and hope to change discs while the machine searches for the filename! You can, however, type **ed b:filename**, and swop discs when the screen message tells you to. The point is that if this filename is not found on the disc that is present in the drive, then the filename is used to make a new directory entry for a new file, one that you are about to create. This file will be given no extension unless you specify one. Another file, a temporary file, is also opened at this time, and will be erased when you finish using ED. The purpose of this second file is as a form of notebook, so as not to take up too much machine space. You never see this second file in the directory, because of the automatic erasure, but you have to be sure that there will be space for it on the disc. If, for example, you have only one directory entry left then you can't be certain of being able to create an ED file. Equally obviously, you can't use ED along with a disc that is write-protected, so you need a copy which is not on the original System disc.

The alternative way of calling up ED is simply to type the ED name and press RETURN. When you do this, you will get a message:

Enter Input File:

which requires you to type a filename that you will use. This can be the filename of an existing file, or a new filename. When you press RETURN on this, you will get the further message:

Enter Output File:

and if you are creating a file for the first time, you will simply press RETURN at this point. You will also press RETURN on this one if you have a file on the disc already and you are updating it, and saving the updated copy under the same name. You need to enter a second filename only if you want an old file read, and the amended version recorded under another name. This might be useful if you needed to keep both versions. If you simply type a drive letter for Output File, then the amended file will be recorded on the specified drive, but with the same name as before. If the filename is a new one, the message:

NEW FILE

appears. In what follows, we'll assume that a new filename will be used so that you can explore the world of ED without affecting a file that is on your disc copy.

Suppose, then, that you place in the drive a disc that contains ED (*not* the System disc, nor any write-protected disc), and then type:

ed tryout

and press RETURN. The disc spins, loading the **ed** program, and then looking for this filename. If the filename is not on the disc, then the words

nA	Append 'n' lines from file into buffer
0A	Append from file until buffer is half full
B, −B	Move to start(B) or end(−B) of buffer
nC, −nC	Move 'n' characters forward(nC) or back(−nC)
nD, −nD	Delete 'n' characters forward(nD) or back(−nD)
E	Save file and return to CP/M Plus
Fstring↑Z	Find string in text
H	Save file, stay for editing
I	Enter insert mode
Istring↑Z	Insert string at current position
Jstring1↑Zstring2↑zstring3	Find string1, concatenate string2 on to it, then delete all characters up to start of string3
nK, −nK	Kill (delete) 'n' lines forward(nK) or back (−nK)
nL, −nL	Move 'n' lines forward(nL) or back(−nL)
0L	Move to start of current line
nMcommandlist	Execute command list 'n' times
n, −n	Move 'n' lines forward(n) or back(−n) and display that line
:ncommand	Execute from present line to line n
Nstring↑Z	Extend find string
O	Return to original file
nP, −nP	Move 23 lines forward(nP) or back(−nP) and display
Q	Abandon file, return to CP/M Plus; you will be asked to confirm Y/N
R↑Z	Read LIB file into buffer
Rfilename↑Z	Read named file into buffer
Sstring1↑Zstring2	Delete all string1, substitute string2
nT, −nT	Type 'n' lines forward or back
0T	Type current line
U, −U	Translate to upper-case
V, −V	Line numbering on/off
0V	Display buffer space figures
nW	Write 'n' lines to new file
0W	Write until buffer is half empty
nX	Write or append 'n' lines to LIB file
nXfilename↑Z	Write 'n' lines to named file; append lines if previous X command applied to this same file
0x↑Z	Delete LIB file
0xfilename↑Z	Delete named file
nZ	Wait for 'n' seconds

Figure 5.1 The command letters of ED. This is a daunting list, but you don't normally need more than a handful.

NEW FILE appear as a reminder to you that you are now creating text, not editing old text. Under the **NEW FILE** message, you will see a prompt which consists of a colon and an asterisk. This type of prompt means that **ed** is waiting for a command letter. Each command is obtained by typing the letter (possibly followed by other items) and then pressing RETURN, and nothing is set into motion until you press RETURN. There is a huge list of command letters, which are listed in Figure 5.1. The list is rather intimidating, but you'll find that only a few of these command letters are used extensively, and some of them might never be useful to you. It's a good idea if you intend to use **ed** extensively to make a copy of the most useful command letters. When you are trying to create a file for the first time, the most useful is **i**. When you press **i** and then RETURN, the **:*** display is replaced by **1:**, meaning that ED is waiting for a line of text which will be numbered as line 1. This difference is *very* important, because when the number/colon prompt is on screen, you can't issue command letters, and when the colon/asterisk is on screen you can't type text. You can switch to the command mode (colon/asterisk) by pressing ALT–Z, and return to the text mode by using **i** RETURN. If you see a display that consists of a line number, a colon and an asterisk, this means that ED awaits a command, and is presently at that line number in some existing text. If you mistake this for a text line, then you will find that pressing ALT–Z to get to command mode (as you think) simply displays **^Z** – you are already in command mode and ED now takes the ALT key to mean the up-arrow symbol.

The **i** command letter means 'insert new text', and it allows you to type both upper- and lower-case letters, by the normal use of the SHIFT key, in numbered lines. You can therefore type whatever text you want, using RETURN to take a new line. The lines will be automatically numbered, but this can be turned off, if you don't want to see it, by using the command letter **-V** when you are in the command mode (colon/asterisk). The line numbers are not recorded with the file, and are a useful guide, so I generally prefer to keep them, particularly for a file of commands in assembly language, Pascal or C. Another point is that the use of **i** when you already have text in the memory can cause a new line to be inserted, and the remaining lines renumbered. It's easier to see that this has been done when the numbers appear.

It's one thing to type in text merrily, and use RETURN to give a new line, but what happens when your fingers slip, and you make a typing error? If you use the normal DEL-left key, then you will see a zero (0) appear in the *next* character position and also a copy of the character that has been deleted. The use of DEL-right produces an up-arrow-K. The delete action *has* been carried out, but it's a lot better for your confidence if you can see it happen on the screen. If you use ALT–H for deleting, you will see the action. This is a throwback to the days when CP/M was originally designed, and computers did not have DEL keys. You will find the same use of ALT–key in other programs that were originally written in the early days of microcomputers –

one well-known example is WordStar. The scheme is now so widespread, in fact, that many manuals urge you to use these key combinations rather than to change to the familiar key use of the PCW machines. The Hisoft ED80 program also makes use of ALT–key actions, but not quite to the same extent. When you have finished with a piece of sample text, you can get back to the command level of ED by pressing ALT–Z, and you can then save your text permanently on disc by typing the command letter **e** RETURN, which saves the text, abandons ED, and restores CP/M PLUS. If you want to save your text, but remain with ED for further activities, you can use the command letter **h** in place of **e**. In this case, though, it's useful to return to the CP/M level so that we can check the directory entry.

Having tried typing a piece of simple text, and saved it, the next step is to look at the disc directory entry. The filename TRYOUT should now appear on the disc, and it will have no extension unless you specified one when you called up ED or specified the Input File. You can use **TYPE** either to see the file on the screen or, by pressing CTRL–P just before you press RETURN, on the printer. Having satisfied yourself that the file is as you remember it, you can now work with this file, to see how the system treats a file that already exists. When you now type **ed tryout**, you will hear the disc spin as **ed** loads, but this time there is no **NEW FILE** message, because the file already exists. All you get this time is the prompt **:*** and the block cursor. Your file, however, is *not* shown. If you use **i** at this stage, you will be able to type lines to which your file can be added. In other words, anything you type now will be added to the start of your file. To get access to your file, you type #**a** RETURN. The **a** command means **Append**, and when this has been done, the prompt now includes the first line number of the file, but you still don't see the file listed on the screen. The hashmark in this command means 'all', as distinct from a stated number of lines. To see the file, you now need to type #**T** RETURN, when all will be displayed. It's this sort of thing which discourages you from using **ed** as a word-processor, because you do need to go through a lot of steps to get anything done, and you need to make use of a lot of commands. You can, however, put several commands into one line, so that you could have typed #**a**#**t** RETURN to have achieved the same effect rather more quickly. Like any strange system, it takes some getting used to. Remember, too, that **ed** was devised in the days when only professionals used CP/M, and no-one ever considered the possibility of owner-drivers coming along.

Having got your file of text into place using #**a** and #**t**, you can now take a look at it. All of the text should be there, because the hashmark (#) is used to specify all of the text. If you had wanted just two lines of text, you could have specified **2a** to append two lines, or **2t** to type just two. Many of these **ed** commands will take number prefixes like this, with the hashmark always meaning 'all of it'. When your text is displayed, then, you'll see the cursor on a line numbered 1. again. If you want to add text, you will have to move to the end of the file, using the command **–b** ; **b** without the minus sign means go to the start of the file. It's at this point that you can get thoroughly confused.

When you see a line number *and* an asterisk then **ed** is awaiting a command with that line ready to be worked on if that's what you want. It's only when the asterisk disappears, however, that you can type text material. Suppose, for example, you happen to be looking at the display:

1:*

which means command mode, line 1 ready. If you now type **i** for insert and type some text, like **add this**, then RETURN, you will see the words appear, and line 2 is then presented for new text. You do *not* see what has originally been typed in these lines, because you are creating a new line. If you escape again by pressing CTRL–Z, you can use commands **b#t** to display the whole of the text. You'll see now that you have created a new text line, and all the other text has been shifted down. Try again, but this time, press CTRL–Z instead of RETURN. This time, you'll see when you list (with **b#t**) that words have been inserted into a line, rather than creating a new line. This is the action of the **i** command, insertion of text. The most irritating feature of this, and all editing actions of **ed** however, is that you can't see what you are doing at the time. Unlike modern word-processor programs, where you see characters being deleted, shifted, corrected and what have you, **ed** requires you to list the text again before you can see what has been done. Let's face it, in 1964 I would have given my right arm for a facility like this, and we're getting a little bit spoilt now! You can see what I'm griping about, though, if you try to alter a line. For example, try loading in your text again if you want to start with a clean sheet. With the prompt on line 1 (meaning that you see **1:***) type **4c** to make the position pointer move along by four characters. All that you can see when you press RETURN is that you have another prompt for the same line. Now type **–4d** which should delete four characters back from the pointer, the word 'This', if you typed the text as originally indicated. Once again, when you press RETURN, you simply get another prompt. You can display your line now using **t**, and you'll see the effect of the change. Now insert a word, by pressing **i**, then RETURN. Type a word like **what**, then press CTRL–Z so as to make the word insert into line 1, rather than occupy line 1 and move the rest of the text down. Now when you use **t**, you still see the line that reads:

is a demonstration file of text

with no **what** in sight. That's because *you did not move the pointer back*. When you use **t** by itself, it prints that line, but from the pointer position only. This is what made your insertion invisible. If you had moved the correct number of characters back before using **t** on the line, you would have seen your insertion. Try it again, this time using **–4c** or **–5c** (depending on whether you put in a space following **what**), then use **t** to see the changed line. Each time you type a character you move the pointer, and any type command operates from that position. The character moving commands have the same pointer moving effect.

Once you appreciate what the character pointer of **ed** is doing, the action of **ed** becomes a lot easier to understand, if not easier to bear. The editing commands operate from the pointer position, either in characters or in lines. For example, **c** specifies character movement, like **4c** (four forward) or **–6c** (six back). The **l** (letter el) command acts in lines, so that you can have **3l** or **–2l**, for example. You can delete characters using **d** with the usual numbers and signs, and you can delete (or kill) lines using **k**, also making use of numbers and a negative sign to go backwards. Unlike modern word-processors, what you see may not be what you get, and what you do get you don't see right away. Despite all that, though **ed** is packed with features, not all of which you get on some high-cost programs. As you might expect, you can use **ed** to find the position of anything in its text. As you might also expect by now, any search will start from the character pointer position, so if you want to search through the whole text, you have to start with the **b** command. For example, if you wanted to find the position of the word **editor** you could command **bfeditor**, with the **b** getting to the start of text, the **f** meaning 'find', and the word we are looking for, 'editor'. You have to be careful to type the word just as you expect to find it – if you specify 'editor', then you won't find 'Editor'. When the command is executed, you'll see the usual asterisk prompt, and the line number will be the number of the line that contains the word. Now the position of the character pointer is *immediately following* the word that you requested. Since this is the last word in this particular line, the command **t** will not produce anything apart from the full stop at the end of the line, and to check that you have actually found the word, you need to type **–6ct**, which will shift the pointer six spaces back, and then type to the end of the line. You can, incidentally, limit the scope of **t** with commands such as **2:4t** (sounds like a weed-killer!) which will type lines 2 to 4, with one extra line always thrown in for good measure and the pointer left at the start of the first of the lines.

Suppose you want to carry out a search and replace action. The command letter for this is **s** and it has to be followed by the word you want to replace, then a CTRL–Z, and then the word you want as a replacement. For example, looking at the example text yet again, suppose you wanted to replace 'text' by 'words'. you would type:

bstext^Zwords

but when you try this, you'll see that only the first 'text' has been replaced. This is because **s** needs a specifier number. If you want only the first three 'text' words to be replaced, then you can use **3s**; if you want to replace *every* 'text' with 'words', then you need to use #s. When you do this, **ed** will inform you where that last position of 'text' was by a message such as:

BREAK "#" AT s

followed by the asterisk prompt in the line that holds the last word that was replaced. The character pointer is now placed following the 's' of the last 'words' that has been put in.

Other features of ED

A text editor the size of ED is not mastered overnight, and in this chapter, I have concentrated on the features that you are most likely to encounter as a CP/M user rather than as a CP/M programmer. Though you *can* use ED as a word-processor, it's not really ideal because of the difficulty of seeing what you are doing. In any case, you will use either LOCO SCRIPT or New Word on the 8256/8512 machines when you want the use of a word-processor. The **ed** utility comes into its own when you have to prepare material for CP/M commands, as we'll see shortly. These applications cannot be filled by using LOCO SCRIPT, because the way that LOCO SCRIPT is recorded on the disc is not the same as the way that **ed** is recorded.

You can turn the line numbering off with the **–V** command if you want to. This makes plain text look more readable, but for most applications of **ed**, the line numbering is useful, if only as a way of keeping track of what is there. Remember that ALT–P will always bring your printer into action if you want to type text, and it will be typed just as it appears on the screen when you use **t**. You can also, of course, type direct from an ED file, using **TYPE filename**.

Another convenience is paging. The editor will divide your text into pages, each of up to 23 lines. You can then display your text on the screen in pages by using **b** to get to the start of the text, then using **p** for page display, starting with **0p** to show the first page. A command that is sometimes useful is **0v** (zero, not 'oh'), which will show two figures. The first of these is the number of bytes that remain available, the second is the total that is free for use when no text exists, the maximum number of characters for **ed**. For some odd reason, even a lot of costly word-processors have no running word-count, which makes them pretty useless for authors and journalists. At least **ed** allows you to estimate a word count by using these figures. Subtract the first figure from the second one to get the number of characters used, then divide by six to get words. It's a reasonable approximation when you have a few thousand words of text, and better than just guessing! Even on the short demonstration text, it's not all that far out.

Chapter Six
Printer Control

The PTR menu

Controlling the printer from CP/M programs is one of the greatest frustrations for the 8256/8512 user, and certainly one of the aspects that gives rise to the greatest number of enquiries to dealers. The reason is that though LOCO SCRIPT and the 8256/8512 machine were designed to work with the printer that is supplied, CP/M is a universal operating system that provides for any machine (using the Z80 microprocessor chip), with any display and any printer. In other words, when you use CP/M, the program should set up the printer, and you will find that the main commercial programs will do just this. The trouble is that the settings that you get may not be to your liking, and if you are using a program in which the printer is not set, then you may find that you want to make considerable changes. To do so needs some programming; it's all a very long way from the quick and simple menus of LOCO SCRIPT. The modification of printer action requires the use of **ed**, and it's not something that you can do in a hurry when you happen to have ten minutes to spare. It helps, then, if you can formulate exactly what you want to do before you start, because you can waste a lot of time trying various things out if you have no clear plan of what you want to do.

To start with, though, we'll keep to simple matters. Even with CP/M running, you still have the familiar strip of commands appearing on the screen when you load the printer with paper or you press the PTR key. Unless you are using continuous stationery, some of these options, such as Top of Form and FF, will not be particularly useful. The choice of Draft Quality or High Quality is very useful, however. You can, for example, print out a rough copy of a trial balance using Draft Quality before you commit it to the slower High Quality print. This choice is the one that you are most likely to use, and the choice is made or altered by moving the cursor and then pressing the [+] or [–] key as appropriate. As with LOCO SCRIPT, the first part of the message line informs you about the state of the printer, and the normal message is **Online**. Page 120 of the manual, Book 1, deals with the various other messages that appear under other conditions, and we shall not repeat the advice here.

Other set-up actions

The real problems appear when you want to make choices other than the few permitted on the PTR key menu. You may, for example, want to change the print style to bold or italic, or to change the size of print. None of this is provided for in the PTR key menu, nor is there any other provision for it in CP/M. Once again, there may be some choice of options from within any program that you happen to be using, and such options will be specific to that program – they might use menus selected like those of LOCO SCRIPT, or by asking you to press certain keys. Once again, we are not concerned with these ready-provided choices, but with how you can make changes for yourself independently. I must emphasise, however, that there are limits. Whatever changes you make to printer settings in the way that will be described are permanent until overruled. In other words, if you set the printer for bold 12-point, then it will stay that way until you change it, or switch off. Similarly, if you change to italic 12-point, and then load an accounts program, don't be surprised if the print style is *not* the one that you specified. This is because, as I have said, many commercial packages set the printer for themselves, and such settings overrule any that you made earlier. What you have to do in such a case is to change the printer control program in the commercial package, using the same techniques as you would to make your own. This is, of course, yet another good reason for using only a copy of any commercial program, and for avoiding like the plague any commercial program that cannot be copied. One thing that you cannot do is to make changes to the printer action while you are in the middle of running a commercial program. This would require considerable modifications to the main program, and is something for the professional programmer. There are, however, a few programs that make provision for you to load and run another program, and then resume the action of the first program.

To start with, how would we make the printer work with 12-point italic? By working through this example, you will see the general method, and be able to apply it to any changes that you want to make. The principles are simple enough. The printer settings are altered by sending to the printer some code numbers that do not cause characters to be printed. Each of these sets of codes will cause some change to the printer setting, and this change will persist until countermanded, or until the printer is reset (using the RESET option in the PTR menu), or the whole system switched off. Each of the printer setting codes will start by sending the number 27 to the printer, and this is the number that is sent when the ESC key of most computers is pressed. On the 8256/8512 machines, the EXIT key is set up to provide this code. Unfortunately, you cannot simply press the EXIT key followed by the code number that you want to use. This is because when you are using CP/M, only CP/M instructions will be accepted. If you press keys, then RETURN, the reaction of the CP/M system is to look for a program on disc whose filename corresponds to what you have typed. If there is no such name, or, as is more usual, there could not be because no name starts with the EXIT character, you get precisely nothing. The problem, then, is how you make the CP/M system feed the codes to the printer, and one solution is to write a CP/M program. Fortunately, you don't have to go to such lengths, because one of the utility programs on your Master disc set provides for feeding codes to your printer. This is very much easier than writing a CP/M program, but because it's so unlike other programs that you use, you need to take some time to become familiar with it.

The SETLST utility

SETLST is the name of the CP/M utility program that sends codes to the printer. The program itself is one that reads characters from a file on a disc, then sends corresponding codes to the printer. In other words, you don't achieve anything just by typing SETLST. You need to have a file that contains the character codes, and the name of this file must follow SETLST. Suppose, for example, that your file of control codes is called 'italp', then you would achieve your printer setting by typing:

SETLST italp

making sure that the disc in the machine contained both SETLST and the file **italp**. Setting the printer, then, boils down to creating a file of commands, and this requires the use of **ed**. You can't get around this easily, unless you happen to have a text editor that makes files that correspond to **ed** files. Some text editors that are sold for the purpose of being used for other programming languages (like Pascal or C) can be used for this purpose, but in this section we'll stick with **ed**, mainly because it is provided as standard. What follows, then, shows how italic 12-point can be set up on the printer by making a file for SETLST.

The printer control file consists of lines of commands, which can also include an explanatory comment. This latter point is very important, because without comments, the file can look very strange and unfamiliar a few weeks after you typed it. With comments, it's easy to see what the file is intended to do, and to make changes if they are needed. Each printer command starts with the ESC character, and this has to be obtained in a rather special way, by using the ^ sign (appearing as an up-arrow on the screen). This is obtained on the PCW machines by the keys EXTRA-U. This up-arrow sign does not form any part of the command itself, it's simply used as a symbol to show that what follows it *is* a command such as ESC for the printer file. Following the up-arrow symbol, the ESC code of 27 can conveniently be represented as 'ESC'. In other words, by using ^**'ESC'** you can send the code number 27 to the printer, and this scheme of using the up-arrow character is used also to send codes like ALT-A (sent as ^ A), the up-arrow character itself (sent as ^), the double quotes (sent as ^ ") and numbers that represent computer actions (such as ^**'13'** for carriage return). Following the use of ^ ESC, you can add either the number code or letter code for the print effect that you want. These codes *do not* need to be enclosed in single quotes. For example, to set the Amstrad printer into italic, the codes ESC 4 must be used. These two code numbers can be put into the file as:

^'ESC'^4

Note that the ESC is enclosed by single quotes, because it is a computer code, but the **4** is not enclosed in quotes because it's a printer code. We can set the Amstrad printer into 12-point type by sending the codes ESC M (capital M, not small m), and this will be coded as:

^'ESC'M

but a number following the **M** would need to be enclosed in single quotes.

To make these two lines into a suitable file, we first of all choose a filename, which might be 'italp'. You must then take a disc which has a copy of **ed** and SETLST on board, and type:

ed italp(RETURN)

which will produce the usual asterisk prompt sign of **ed**. Type **i**(RETURN) to get the editor into insert mode, and type your two lines slowly and carefully. Remember that you can't use the DEL keys with **ed**, you have to use ALT-H to delete a faulty character, and EXTRA-U to get the up-arrow sign. When you have pressed RETURN on the second line, you can check the two lines by typing **#b#t** RETURN, and if they are correct, press ALT-Z to return to the control mode of **ed**. Now you can press E for Exit, and your file will be recorded on disc. It can then be run by typing SETLIST italp (RETURN).

Figure 6.1 (a) shows these two lines as put into a file with ED, and produced in this example by using TYPE. Both the file and the test lines are printed in 12-point italic because SETLST italp was used before typing the file out. The comments 'italic' and '12-point' have been put into the lines of the printer-control file, separated by a space. Normally, a complete command line to the printer will consist of a set of characters with no spaces. The use of a space forms a signal to the SETLST program that what follows is simply a comment and not any part of a command. It is therefore ignored when SETLST acts, but it remains in the file so that when you type out the file you can see the

(a)
```
^'esc'4 italic
^'esc'M 12-point
```

(b)
```
    italic 12-point
This is a set of lines intended to test the action of printer
setting files, using SETLST. By looking at more than one line
its easier to see what is happening.
```

Figure 6.1 (a) The two-line SETLST command file for 12-point italic printing. (b) The result of using SETLST on the command file.

comments that explain what the effect of each code line will be. From that, it follows that you have to be careful not to put in a space at any other part of the command.

So far, so good – we can command the printer to alter its action by means of this file which is used by SETLST. The problem now is what we do to make other alterations. Suppose, for example, that we had another file, called **biglet** that we used to put the printer into its enlarged letter mode. There are three sets of codes that can carry out this action, but the most useful one for this purpose is ESC W1, because the other options **SO** and **ESC SO** are automatically cancelled at the end of the line. The **ESC W1** is then coded as:

^'ESC'W1

and it can be typed, recorded and used in the usual way. Figure 6.2 shows the result. The large type style has been set, but the type is still italic. The reason is that a new SETLST file does not automatically cancel any existing codes. The effect of sending codes to the printer is to alter the memory of the printer itself, and sending another code simply adds another command, it does not necessarily clear the others. There are some codes which are incompatible, and for which one will always supersede another. You cannot, for example, use High Quality with Condensed type, or with superscript or subscript. In the main, though, if you want to change printer effects, it's a good ideal to start any printer file with a command line that resets all of the defaults. In other words, your first command puts the printer into the same state as when you first switch on. This is done by using ESC @, and Figure 6.3 shows a file that has been constructed to restore defaults, and then set the printer to proportional spacing and bold type. The ESC @ line will have the effect of cancelling any other actions that have previously been set, and the next two lines set the desired effects. The test lines, as usual, prove that the action has been carried out.

All of the normal printer actions can be controlled in this way, and Figure 6.4 lists the most common of these actions. Most of the actions are available as a switch-on code and a switch-off code, so that a program can call on SETLST to change codes at various stages. This, however, is not likely to be something that you will get involved in. More important, at this stage, is the

(a)

```
^'esc'W1
```

(b)

```
This is a set of lines intended to test the acti
on of printer
setting files, using SETLST. By looking at more
than one line
its easier to see what is happening.
```

Figure 6.2 (a) The command for enlarged type, and (b) its result. The italic style has not been cancelled.

```
^'ESC'@ defaults
^'ESC'p1 proportional
^'ESC'E bold
```

Figure 6.3 A command file that sets defaults first to cancel previous changes, and then goes on to specify bold type and proportional spacing.

Command	*Result*
ESC 15	Condensed on (also SI)
ESC 18	Cancel condensed
ESC M	Elite on
ESC P	Cancel Elite
ESC p 1	Proportional spacing on
ESC p 0	Cancel proportional spacing
ESC W 1	Enlarged on (also ESC SO or SO)
ESC W 0	Cancel enlarged
ESC 4	Italic on
ESC 5	Cancel Italic
ESC m 1	High quality (also ESC × 1)
ESC m 0	Cancel high quality (also ESC × 0)
ESC G	Double-strike on
ESC H	Cancel double-strike
ESC E	Emphasised on
ESC F	Cancel emphasised
ESC ! n	Set mixed mode

In this mode you can use a number n which will set more than one print mode, so avoiding multiple commands. The codes are: Enlarged=16, Double-strike=8, Bold=4, Condensed=2, Elite=1, and you need to add the numbers corresponding to the effects that you want. You cannot mix some styles, such as bold and condensed, but you will get the next-best (such as double-strike and condensed). Repeating the instruction with n=0 cancels the effects.

ESC S 0	Set superscript
ESC S 1	Set subscript
ESC T	Cancel superscript or subscript
ESC – 1	Set underline start
ESC – 0	Set underline end
ESC X	Slashed zero
ESC o	Cancel slashed zero
ESC I 1	Print extended character list
ESC I 0	Cancel extended character list
ESC @	Reset printer
ESC d	Make current settings be retained after reset
ESC 24	Clear out text in printer buffer to stop printing

Figure 6.4 The list of printer control codes – only the codes for print styles are shown here, and the full list, including tabulation and line spacings, is contained in the manual.

treatment of two symbols, the slashed zero and the pound sign. The slashed zero is sometimes wanted when the zero occurs along with the letter 'o' and the two must easily be distinguished. The slashed zero is selected by using the codes ESC X, and the unslashed zero, which is the default, is selected by using ESC o (letter lower-case oh, not zero!). The pound sign is another problem entirely. The problem is that there are only a limited number of ASCII code numbers available, and some of them have to be re-assigned when a different alphabet from another country is selected. The default set in the UK version of the alphabet assigns the pound sign to ASCII code 35, which in almost every other set except the Spanish one is used for the hash sign. This is why, when you try to type something with a hashmark using the standard UK character set you get a pound sign in place of the hashmark. The annoying thing is that the correct signs appear on the screen, but not on the printer. The correct hash sign can be restored by using the US character set for the printer and this is done by using the ESC R0 codes. The zero, or any other number, in this code sequence *must* be enclosed in single quotes, and must follow an up-arrow sign. The trouble here is that if you have a piece of typing in which you have used both the hashmark and the pound sign keys freely, they will all appear either as hashmarks *or* as pounds, depending on which character set you have chosen, as Figure 6.5 illustrates. There are two versions each of the hash and the pound signs which differ slightly, as you can see, but there isn't any neat simple way of mixing the signs if you insist on using the standard keys. One way that we'll look at later is to redefine the keyboard so that the character for the English pound is used separately from the hashmark.

The PAPER utility

The important printing items such as margins, line spacing, tab settings and so on, can all be set by using the SETLST utility, as all of these things are set by sending codes to the printer. As it happens, however, the Amstrad Master disc set contains a separate utility, PAPER, which allows *some* of these items

(a)

```
This uses the hash £ and the pound £ signs
in one £££££££££££ lot

This uses the hash # and the pound # signs
in one ########### lot
```

(b)

```
^'ESC'@
^'ESC'R^'0'  US Set
```

Figure 6.5 The appearance of the hash sign and the pound sign (a) in the default character set and the US set of the printer. (b) The file for setting the US character set. Note that this applies to the printer only, not to the screen.

to be set up with rather less strain than is needed for the use of SETLST. One of the particularly useful features of PAPER is that it allows easy changing between the two main paper sizes A4, and A5. If you have a copy of the PAPER utility on your disc, then typing **PAPER A5** RETURN will set the new paper parameters of a single sheet – six lines per inch, 50 lines per page, and a three-line gap between pages. It does *not*, however, set the number of characters per line, because that depends on the size of characters that you are using, and will have to be set, like the character size, by using SETLST. In the same way, you can use PAPER A4 to set the number of lines per page at 70, the other settings remaining as for A5. You can also set up for continuous stationery by typing a number following PAPER, the number being the length of the sheet of continuous stationery. For example, to set for the very common 11-inch sheet, you would type PAPER 11. As with all PAPER commands, the settings that have been allocated will be displayed on the screen to remind you.

PAPER also allows some free-range settings, and the permitted commands, with letter abbreviations, are illustrated in Figure 6.6. For many purposes, if you use standard A4 and A5 sheets, you may never need to specify other sizes, but if you use, for example, headed paper of unusual size then this facility could be very useful. Another point worth remembering is that if the DEFAULTS command is used, then the settings that have been specified will be retained when the printer is reset, instead of being lost. They will not, of course, be retained if the machine is switched off.

The command following PAPER must be separated from each other by commas or by single spaces. In each case, n means that a number must be specified.

FORM LENGTH n	Sets page length of n lines. Default lines per inch is 6, gap zero.
GAP LENGTH n	Sets number of blank lines at foot of page.
LINE PITCH n	Sets pitch at 6 or 8 – no other numbers allowed.
SINGLE SHEET	Sets use of single sheets and Paper-out Defeat ON.
CONTINUOUS STATIONERY	Selects use of continuous stationery, with Paper-out Defeat OFF.
PAPER OUT DEFEAT ON	
PAPER OUT DEFEAT OFF	
A4	Sets for use of A4 paper
A5	Sets for use of A5 paper
n	Sets for continuous stationery with form length of n inches
DEFAULTS	Makes current setting the defaults when the printer is reset.

Figure.6.6 The commands for the PAPER utility.

SUBMIT and PROFILE

All versions of CP/M allow what is called a 'SUBMIT' file to be run, and in fact, the files that we have used with SETLST are really a form of SUBMIT file. A SUBMIT file contains a list of programs that are to be run or actions to be carried out. Suppose, for example, that you used **ed** to create a file called TEST.SUB and it consisted of the two lines:

```
dir
ed prnt.ccp
```

and recorded this file. From then on, by commanding **SUBMIT TEST.SUB**, you would make the machine produce a directory, then load in **ed** along with the **PRNT.CCP** file. SUBMIT is a very useful type of command when a number of actions are needed, or when several programs are needed in sequence. Even more useful, however, on a day-to-day basis, is the PROFILE.SUB file. This is a form of SUBMIT file, but with the difference that you never have to type **SUBMIT**. The initialisation of CP/M includes a search for a file called PROFILE.SUB, and if this is present on the disc, the commands in the file, which is a normal SUBMIT type of file, will be executed as part of the switch-on process. This is possible *only if the disc is write-enabled*, so that no PROFILE.SUB file can be used with the System disc, only with write-enabled copies. From what has gone before, you might want a PROFILE.SUB file to contain a SETLST PRNT.CCP to set up a printer, and we'll look in the following chapter at the use of SETKEYS to alter the keyboard. All of these items can be put into a list, using **ed**. Once this file exists on a write-enabled copy of the System disc, then, when the system disc is used after switching on, or following a reset, the CP/M program itself will be loaded in, followed by the execution of the commands in the PROFILE list. This takes quite a noticeable time, incidentally, but it can save a lot of keyboard use first thing each morning. The System disc contains PROFILE.ENG, which consists of a set of **pip** actions to transfer programs to the M drive, as we have seen.

Graphics

The commands that can be sent to the printer include some that allow graphics printing. This allows the printer to produce such material as company logos, etc. as part of headings, and even to produce illustrations. The amount of work that is needed to produce more than a simple logo, however, seldom justifies the use of the graphics commands for this purpose. For illustration work, the much simpler method is to buy a graphics package which will allow the picture to be created by the use of a light-pen or similar means, and the printing then carried out by the program with no need for further effort.

The generation of a simple company logo, however, can be done comparatively easily provided that the planning is carried out well. You have to start with a planning aid, and the best by far is some old-fashioned graph paper that is divided into 8 × 8 sections. If, in these metric days, you can find some $\frac{1}{8}$-inch division, inch ruled, graph paper, then this is ideal – but for small-scale work you can use the grid in Figure 6.7. Each large square (i.e. each set of eight small squares) will take up a length of $\frac{8}{72}$ inch, equal to $\frac{1}{9}$ inch, so that you will need to work with nine large squares in each direction to create a picture one inch wide and deep. This requires a large amount of planning paper if you want to plan a large logo! To illustrate the work that is needed, then, I shall show what is involved in creating a logo shape that requires a miximum of six squares in each direction. You need not, of course, take an equal amount of squares in each direction, nor do you have to use the full size of the planning grid. Do remember, though, that a small logo will look very insignificant, because one full square represents only $\frac{1}{9}$ inch, and that's a very small piece of an A4 sheet, though it might look better on A5.

The first step, then, is to plan the shape on the matrix. For your own work, the best method of doing this is to lay a piece of tracing paper over the grid, because you will probably need more than one attempt before you get the

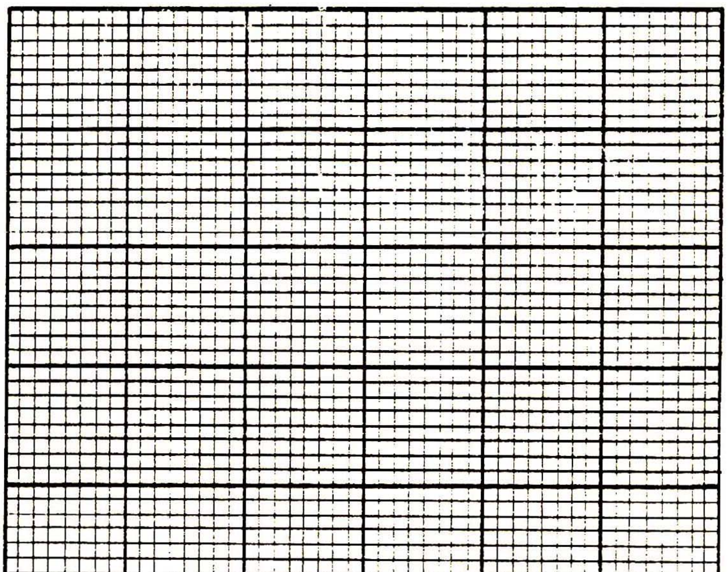


Figure 6.7 A grid for designing printer graphics. You can make these for yourself with $\frac{1}{8}$ scaled graph paper.

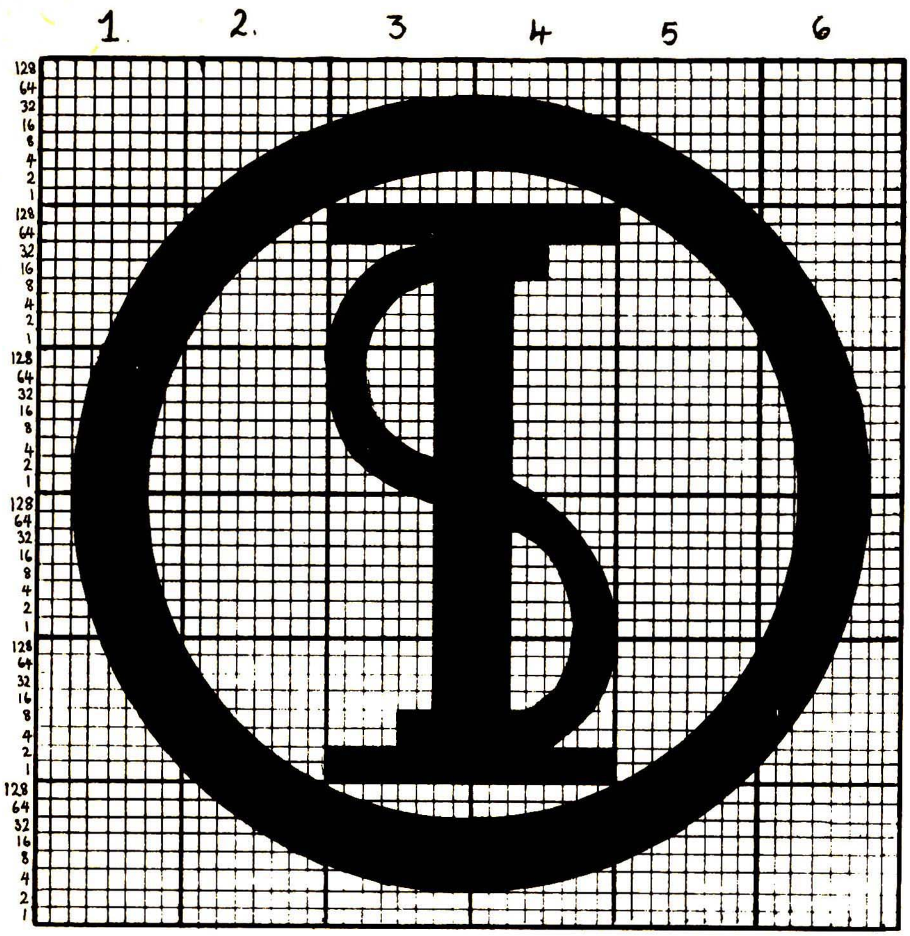


Figure 6.8 A pattern drawn on the grid, with the pixel code numbers shown at the side.

logo to your satisfaction. Avoid fiddling detail, because it probably won't be noticeable in the final effort. A good idea of how the logo will look will be obtained if you view your plan from a distance of about 15 feet away – this will give about the same impression as reading it on a normal paper sheet printed in its final size. Figure 6.8 shows the logo plan for this example. At this point, you have to decide what you will do when a portion of one of the small squares is shaded. The usual method is to count half a square or more as a shaded square, and less as unshaded. The point is that each tiny square in the pattern – and remember that there are 64 of these small squares in each main square – will be represented by a number. The number codes run vertically, with the number code 128 representing the top of a vertical column of eight small squares, and the number 1 representing the bottom of a set. These numbers have been marked in on the plan of Figure 6.8. From now on, we shall give the small squares their rightful name of pixels, so that when I say pixels, you know that I shall be talking about the small units, 64 of which make up one square on the pattern. The next step in creating the logo pattern is to find the code number for each vertical line of pixels in each line of the pattern.

Now in this example, there are six lines, because the pattern is six squares deep. In each line, there are again six squares, each of which contains eight lines of pixels, a total of 48 lines of pixels per line. Our logo will therefore be represented by six lines of numbers, with 48 numbers in each line. It's a lot of numbers, and that's why you need to avoid trying to make large logos. You now have to form a table of these numbers. This is done line by line, finding the number for each vertical row of pixels, working from left to right in each line. Keep the lines separate, because each line needs its own command in the final program. To find the number for a vertical set of pixels, look at the shaded pixels in the row. If none of the pixels is shaded, then the code for that row is **0**, no problem. If all of the pixels are shaded, then the code is **255**, again fairly easy. If some are shaded and others not, then the code number is formed by adding the numbers shown at the side for each shaded pixel. It's now that you have to decide whether a partly-shaded counts or not. If you look at the pattern in Figure 6.8, you will see that the top row contains no shaded pixels in the first square, so that the numbers for this row consist of eight zeros. The next square has a first vertical row of pixels unshaded, giving another zero code. The next vertical row, however, has the pixel mainly shaded in the position that is coded as 1. This makes the number for this row equal to 1. In the next along, the 1-position is shaded, and we can take the 2-position as being shaded also, applying the rule that half-shading counts as complete shading. Adding the 1 and 2 gives 3, the code for this row. Each row in turn has its code number determined by adding up the position numbers for the shaded pixels. After much blood, sweat and tears, you get the 48 numbers for the row, and there are only another five rows to go! The amount of work, incidentally, is pretty much the same whether the logo is simple or complex – it's the size that really counts. A logo that consists of a narrow strip, either horizontally or vertically, is much easier to program than one like this which needs most of the squares in the grid.

That's the worst part over. The next part is to write the program that will create the logo, and this is done using a SETLST type of file. The form of this file is fixed by the size of your logo, but it must always start in the same way. That is to fix the line spacing of the printer at $^{8}/_{72}$ inch. This must be done because the default setting will leave gaps between the rows of your logo pattern, rather spoiling the effect (unless, of course, you happen to like it better that way!). The $^{8}/_{72}$ inch setting is achieved by using the codes **ESC A8** in the usual way. That casual phrase conceals a lot of work, as the listing of Figure 6.9 shows. Each number in the listing has to be placed in the file using the up-arrow symbol (EXTRA-U) and enclosed by single quotes, making this a very great deal of work for what will be quite a modest logo. The main

```
     1:   ^'ESC'K^'48'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'1'^'3'^'3'^'7'^'7'^'15
'^'15'^'31'^'30'^'30'^'60'^'60'^'60'^'60'^'60'^'60'^'60'^'60'^'60'^'60'^'30'^'30
'^'30'^'15'^'15'^'7'^'7'^'3'^'3'^'1'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'
     2:   ^'CR'
     3:   ^'LF'
     4:   ^'ESC'K^'48'^'0'^'0'^'0'^'0'^'0'^'3'^'15'^'63'^'127'^'254'^'252'^'240'^'
224'^'192'^'192'^'128'^'0'^'199'^'207'^'222'^'216'^'248'^'240'^'255'^'255'^'255'
^'255'^'240'^'240'^'192'^'192'^'192'^'192'^'0'^'128'^'128'^'192'^'224'^'240'^'24
8'^'252'^'127'^'63'^'15'^'7'^'1'^'0'^'0'^'0'
     5:   ^'CR'
     6:   ^'LF'
     7:   ^'ESC'K^'48'^'0'^'0'^'0'^'15'^'255'^'255'^'255'^'240'^'128'^'0'^'0'^'0'^
'0'^'0'^'0'^'0'^'0'^'248'^'254'^'30'^'7'^'3'^'3'^'255'^'255'^'255'^'255'^'1'^'0'
^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'128'^'224'^'255'^'255'^'255'^'
31'^'0'^'0'
     8:   ^'CR'
     9:   ^'LF'
    10:   ^'ESC'K^'48'^'0'^'0'^'0'^'240'^'255'^'255'^'255'^'15'^'1'^'0'^'0'^'0'^'0
'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'255'^'255'^'255'^'255'^'192'^'224'^'1
12'^'126'^'31'^'7'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'1'^'7'^'255'^'255'^'255'^'24
8'^'0'^'0'
    11:   ^'CR'
    12:   ^'LF'
    13:   ^'ESC'K^'48'^'0'^'0'^'0'^'0'^'0'^'192'^'240'^'252'^'253'^'127'^'63'^'15'
^'7'^'3'^'1'^'0'^'0'^'3'^'3'^'3'^'3'^'15'^'15'^'255'^'255'^'255'^'255'^'15'^'15'
^'31'^'123'^'243'^'227'^'0'^'1'^'3'^'3'^'7'^'15'^'31'^'127'^'254'^'252'^'240'^'2
24'^'0'^'0'^'0'^'0'
    14:   ^'CR'
    15:   ^'LF'
    16:   ^'ESC'K^'48'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'0'^'128'^'192'^'224'^'
224'^'240'^'240'^'248'^'248'^'120'^'120'^'124'^'60'^'60'^'60'^'60'^'60'^'60'^'60
'^'60'^'120'^'120'^'120'^'248'^'248'^'240'^'240'^'224'^'192'^'192'^'128'^'0'^'0'
^'0'^'0'^'0'^'0'^'0'^'0'^'0'
    17:   ^'CR'
    18:   ^'LF'
    19:
```

Figure 6.9 The SETLST file for the logo drawn in Figure 6.8.

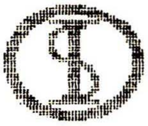


Figure 6.10 The result of all the hard work – the logo drawn by the file in Figure 6.9.

consolation is that it probably will be done only once! That once, however, entails a lot of effort, and if even the slightest item is incorrect, the whole appearance may be changed. We need now to examine the listing of Figure 6.9 in detail, because it's only from an example like this that you really get the feel of how this type of action is done. The result, incidentally, appears in Figure 6.10, and you will realise from the listing that every dot on that logo has had to be programmed into place.

In detail, then, you start by typing in a program to set the printer to 8/72 inch spacing, using **^'ESC'A^'8'**. This should be a separate program, because you will encounter problems, such as odd characters being printed following the logo, if this is made part of the same file as the logo file. You can create this file, called perhaps SPACING, using ED in the usual way. The file for the logo shape consists of six main lines of characters. Each line must start with the control characters that switch the printer to graphics mode, and prepare for the numbers that follow. In particular, there will be 48 numbers per line of the shape, and this number must be communicated to the system. The basis of the command is:

ESC K nl n2 (then data)

so that two numbers, using the up-arrow symbol and single quotes, have to follow the **K**. Of these two numbers, the second will usually be 0 unless you must use large logo shapes. The first number will normally be the count for the line, 48 in this case. The reason for the second number is that you might just possibly want to use a number higher than 255. When you do this, the two figures have to be used. Unless you are a glutton for punishment, you aren't likely to do this, but if you do, then Figure 6.11 shows how the second figure is used. In our example, though, the total for a line is 48, and so the introductory statement is:

^'ESC'K^'48'^'0'

for *each line*. This has to be done because the printer reverts to normal text printing at the end of each graphics line.

Following the introductory characters, the code numbers for each line are put in. There must be 48 of them, because this is the number that has been specified. Any mistake here, with 47 or 49 numbers, will cause trouble. Each number has to be put in with the up-arrow symbol and the single quote marks, making the entry of the number tedious and therefore liable to error. You must work slowly and carefully, checking as you go, and remembering that to delete with ED you need to use ALT-H. One good idea is to count the numbers before you press RETURN on each line. Following a line of graphics codes, you need the carriage return command and the line feed command. These should be put into separate lines, and they are coded as shown, using ^**'CR'** and ^**'LF'**. With all the lines of the program in place, the file can be recorded, and then run by typing:

SETLST LOGO

using the name, logo, in this example, that you decided on for the file when

If a number is greater than 255, then divide the number by 256, and find the *whole number* part of the result. For example, suppose that you need 460 pixels, then 460/256 gives 1 and a fraction. The 1 is called the most significant part. The remainder after this division is 460 —1×256=204, and this is called the least significant part. The number is then coded in least, most order, like ^**'204'**^**'1'**. This technique is needed mainly when dual-density printing is used, with pixel numbers up to 960 being available. Dual density is obtained by using ESC L in place of ESC K at the start of each line.

Figure 6.11 How to proceed when more than 255 pixels are used in one line.

you created it with ED. Unless you have forgotten, in your eagerness to get things going at last, to load paper into the printer, you should see the pattern appear.

There's one final step that you can take now. It's a nuisance to have to run SETLST SPACING and then SETLST LOGO in order to get the logo on a sheet of paper, and you will probably also want to run another SETLST file, NORMAL, to reset the printer afterwards, otherwise you will have rather close-spaced lines when you want to print anything else. To get around this, you can make a SUBMIT file. If, for the sake of example, you wanted to call this file COMLOG, then you would type **ED COMLOG.SUB** RETURN to get into the ED action. Using the **I** instruction, you would then type the following lines for the file:

1. **SETLST SPACING**
2. **SETLST LOGO**
3. **SETLST NORMAL**

and then return as usual by pressing ALT-Z, then E RETURN. This creates a file called in this instance COMLOG.SUB – and the tailpiece, SUB, is vitally important. You can now place the SUBMIT utility on to your disc, using **pip**, and from now on you will run all three actions simply by typing the single command:

SUBMIT COMLOG

Note that you don't have to specify COMLOG.SUB, because the SUBMIT utility will use only files that have the .SUB ending. By making use of SUBMIT in this way, you can change a set of tedious routine actions into one short and simple command.

Chapter Seven
Keyboard Antics

The action of SETLST is closely paralleled by that of another Amstrad utility, SETKEYS. As the name suggests, this allows you to set up the keyboard so that the action of each key can be specified. This is done by using a number code for each key (not to be confused with the ASCII codes that pressing the key will generate), a letter code to show whether the key is used alone or with SHIFT, ALT or EXTRA, and a set of characters that determines what the key will do. Rather than start to explain all of the options that exist, it's easier to start by looking at an existing file that SETKEYS can act on. This is the KEYS.WP file which is on Side 2 of your Master discs, and the listing is shown in Figure 7.1. The printer has been put into its US language set for this example so that the hashmark will be correctly displayed.

The first item that you need so as to be able to work with SETKEYS files is a list of the key numbers. These are illustrated in the manual, and a copy is shown in Figure 7.2. In the first part of the KEYS.WP file, which is the file that is used to set up the machine for word processing packages such as WordStar and New Word, you will see that each line of the first part of the file starts with a number, which is the key number. The first line, for example, concerns key 14, which is the up-arrow key for moving up the text listing in word-processing. Following this key number are the letters N and S. N means normal, meaning that the key is pressed by itself, and S means that the SHIFT key is pressed as well. By using both N and S in this line, the same action will be defined whether the key is pressed by itself or along with the SHIFT key. The other abbreviations that are used in this way are fairly obvious, A for ALT, E for EXTRA, and SA for SHIFT-ALT.

Following the key number and its state (normal, shift, etc.), we have the action. This is placed between inverted commas (*double* quotes), and the action for key 14 is shown as ^E, meaning ALT-E. The standard CP/M action of ALT-E (usually called CTRL-E) is to move the cursor down one line (or the text up, depending on your point of view), so this is the action of the up-arrow key in word processors. This is not the same as its use normally under CP/M, so that running SETKEYS KEYS.WP has changed the action of this and other keys. The second line defines what will happen if this same key is used with ALT or with SHIFT-ALT. This time the action will be to generate a code number that is written as #9E. The hash sign here means that

```
14 N S  "^E"       ^E
14 A SA "^'#9E'"   ^QE
 6 N S  "^D"       ^D
79 N S  "^X"       ^X
79 A SA "^'#98'"   ^QX
15 N S  "^S"       ^S
 5 N    "^D"       ^D
 5 A    "^S"       ^S
 5   S  "^F"       ^F
 5   SA "^A"       ^A
13 N    "^'#9C'"   ^QD
13   S  "^'#9C'"   ^QD
13   SA "^'#9D'"   ^QS
12 N    "^C"       ^C
12 A    "^R"       ^R
12   S  "^'#90'"   ^QC
12   SA "^'#91'"   ^QR
20 N    "^'#92'"   ^QF
20   S  "^'#93'"   ^QA
10 N    "^'#94'"   ^KB
10   S  "^'#95'"   ^KK
11 N S  "^'#96'"   ^KC
 3 N S  "^'#97'"   ^KV
 1 N S  "^B"       ^B
23 N S  "^V"       ^V
16 N S  "^G"       ^G
16 A SA "^'#9A'"   ^QY
72 A SA "^'#9B'"   ^Qdel
 8 N S  "^'#1B'"   esc
66 N S  "^U"       ^U

E #90 "^QC"
E #91 "^QR"
E #92 "^QF"
E #93 "^QA"
E #94 "^KB"
E #95 "^KK"
E #96 "^KC"
E #97 "^KV"
E #98 "^QX"
E #9A "^QY"
E #9B "^Q^'#7F'"
E #9C "^QD"
E #9D "^QS"
E #9E "^QE"
```

Figure 7.1 A listing of the KEYS.WP file is on the Master disc. This is for use by word-processors other than LOCO SCRIPT, such as New Word.

the number is written in a form called hexadecimal, used by programmers, and you will not be forced to use this type of number when you redefine keys for yourself. Where numbers in the range #80 to #9A appear, you can generally expect to find them defined separately elsewhere. The #9E number, for example, is another of this set, known as 'expansion characters' and its definition for word-processing use is shown in the later part of the list, starting with the E letter to distinguish this list from the key list.

The KEYS.WP file is a very specialised one, and it's likely that if you want to set up key definitions for yourself you will be looking for something comparatively simple, like getting one particular character when a program is

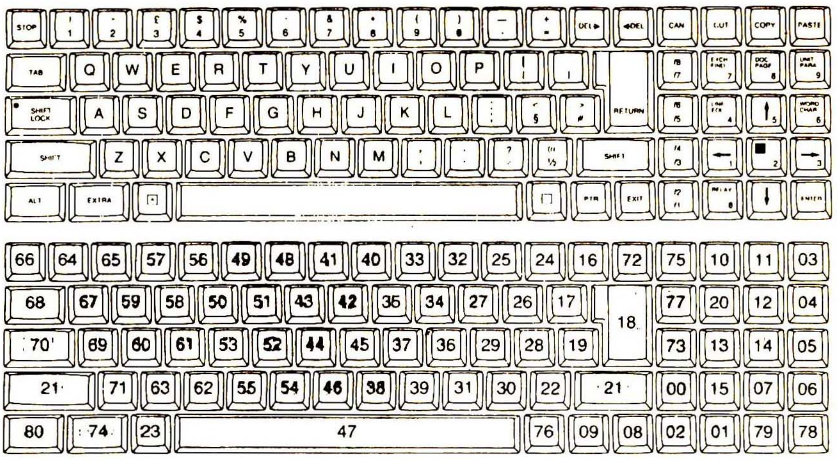


Figure 7.2 The keyboard, showing the key code numbers that have to be used in a SETKEYS file.

running under CP/M. A good example to use at this point is the solution to the problem of printing both pound signs and hashmarks in one piece of work. Suppose that the hashmark is the sign that is most wanted, and the pound sign is needed only from time to time. The printer would then be set to the US character set, producing the hashmark both for the hashmark code and for the pound code. We then need to redefine the pound character key, which is SHIFT 3. The problem is what to shift it to, and the answer comes from a look at the list of extended characters that can be printed following sending the commands ESC Il to the printer. In this extended range, the character whose ASCII code is 6, and which normally appears on the screen as ^**F**, will cause the printer (not the screen) to produce the pound sign. In the file for SETLST, then, we set the language to the US character set, and we use:

^'ESC'I'^'I'

to switch to this set of characters for codes that are not usually printed. That concludes the work on the printer. We must now prepare our keyboard.

The file for SETKEYS must include the line:

57 S "^'6'"

which will redefine the SHIFT 3 key, key number 57, as providing the ASCII code of 6. Note how this is stated, with the whole definition enclosed in double quotes, and the number preceded by the up-arrow sign, and in turn surrounded by single quotes. If this file is called POUNDF, then the command SETKEYS POUNDF will cause the keyboard to be redefined. You can then work to create a file that contains both hashmarks and pound signs. Each time you use the pound key, you will get the ^**F** symbols appearing on the screen rather than the pound shape. When the file is printed, however, the signs will appear correctly, as the sample in Figure 7.3 shows.

(a)

```
57 S "^'6'"
```

(b)

```
^'ESC'@
^'ESC'R^'0'
^'ESC'I^'1'
```

(c)

```
This is pound £££ this hash ###
```

Figure 7.3 The files for SETKEYS (a) and for SETLST (b) that will give both hash and pound signs on the printer, as shown in (c).

Using SETKEYS for altering the action of the normal keys of the machine is convenient when a few keys have to be altered. This can be needed surprisingly often, because programs that are sold as 'installed' for the PCW computers do not necessarily use the keys of the PCW machines in an efficient way. One that does is Caxton's Scratchpad Plus and, incidentally, the same firm's Smartkey is a utility that allows you to change key specifications without the need for mastering the use of SETKEY files. Many other popular programs, however, have been adapted for the PCW computers only in the sense that they have been put on to 3-inch discs and will run exactly as on any other CP/M machine. You will find, however, that manuals have very seldom been adapted – you may be told, for example that your computer has no SHIFT LOCK, but you will find that the SHIFT LOCK on the PCW machines will work, and you will find that manuals always refer to a CTRL key, meaning the key that is labelled as ALT on the PCW machines. The key that the manuals refer to as ESC is the EXIT key of the PCW machines. Remember also that when such programs are described as producing files that can be used by 'industry-standard' word-processors they do *not* mean LOCO SCRIPT!

The fact that many programs are of US origin can mean that you get dollar signs used in money sums (as in SuperCalcII), or that the keys which produce word-processing actions in LOCO SCRIPT are not used for these purposes in other word processors, and you need to use the same keys as on any other computer that did not have word-processing keys included. For very many of these programs, an hour spent in making a suitable SETKEYS file can be very rewarding. Programs of UK origin will generally provide for the £ sign in money sums, and are more likely to pay closer attention to the features of the PCW machines. Since it takes a fair time to become used to the action of a program, it's a good idea to spend some of this time preparing a list of keys whose action could be changed, and possibly also SETLST file actions such as printing a pound sign in place of the dollar. One good idea is to look at all of the files on the disc for any program. These are sometimes concealed, and you may have to use DIR[FULL] to get a complete listing. If there is no key-changing file, then you can add one of your own. You are not likely to find any SETKEYS or SETLST file on programs of US origin, because all of these programs were written before the advent of the PCW machines, and the SETKEYS and SETLST files are not part of standard CP/M, but special extensions for the Amstrad machines only. You can expect to find such files only on very recently written software of UK origin.

For changing to another language set, there are more convenient ways than the use of SETKEYS, as we'll see later. For the moment, though, it's useful to explore the capabilities of SETKEYS, because its actions extend well beyond the simple substitution of keys, no matter how useful that may be. The need to substitute keys is not so pressing as you might imagine, because the set of characters from the use of the ALT and EXTRA keys is just as available to you under CP/M as under LOCO SCRIPT. The example of the pound key redefinition, in fact, is one of the most common ones. For a lot of purposes, however, the use of 'expansion strings' can bring even more benefits. An expansion string is a set of characters that can be created by pressing a key, just as if you had laboriously typed them out from the individual keys. Suppose, for example, that you are experimenting with a file called TRIAL which you intend to use with SETLST. In the course of this work, you will have to type the phrase ED TRIAL RETURN many times, and it would be particularly useful to have this complete action carried out when a single key was pressed. As it happens, most of the keys on the right-hand side of the PCW keyboards can be used just for such purposes. The most useful are the function keys **F1** to **f8**, since the marking on the key does not conflict with what you intend to do with it. It's difficult to associate a key marked CUT with an action like loading a program, but if you assign the action to **f1** you can usually remember it better!

As usual, assigning a key action like this is done with a SETKEYS file. In this case, however, we use the 'expansion token' for the key. The list of standard allocations of these token numbers is shown in Figure 7.4 – using ordinary denary numbers rather than the hex numbers that are used in the manual. Suppose, for example, that you want the **f1** key to produce the effect of typing ED TRIAL at a time when you are working on a SETKEYS file called TRIAL. To achieve this, you use ED to make a file which we can call KEYF1, consisting of the single line:

```
E 129 "ED TRIAL ^M"
```

and save this to use with SETKEYS. In this line, the **E** is used to indicate that an expansion string will be used. The key number of 129 has already been allocated to function key f1, otherwise we could perform the allocation in the usual way, described earlier. The action that we want is enclosed in quotes, and consists of the names ED TRIAL, and the ending **^M**. This CTRL-M sign means the same as pressing the RETURN key, and is the way that is universally used in this type of file to get the RETURN key action. When you

Number	*Key*	*Standard*
128	STOP	^C
129	f1	^Z
130	f2	^Z
131	f3	^Q
132	f4	^Q
133	f5	^S
134	f6	^S
135	f7	^P
136	f8	^P
137	DEL→	^G
139	CAN	^H
140	CUT	^U
141	PASTE	^W
142	FIND	^]
143	EOL	^F^B^B
144	LINE	^F^B
145	↑	^
146	[+]	^ $\overline{V}$
147	Left	^A
148	CHAR or →	^F
149	RELAY	^R
150	↓	^^
151	ALT DEL→	^K
152	[—]	^\
153	ALT ↓	^E
154	ALT DEL	^X

Note: up = up-arrow, down = down-arrow etc., [+] = ⊞[−] = ⊟

Figure 7.4 The expansion string numbers and the keys to which they are assigned in the PCW keyboards, with the standard settings.

press the f1 key, then, it doesn't just produce ED TRIAL on the screen, it also starts the action, just as if you had pressed the RETURN key after typing ED TRIAL.

The use of expansion strings can be very helpful if you need to keep phrases in stock or if you need to call up a variety of programs. You are most likely to need them when you are developing programs of SETKEYS files for yourself, but several commercial programs would certainly benefit from the use of expansion strings, if only to avoid the need for a lot of typing. The main use of SETKEYS with commercial programs, however, will be to regain the use of keys that the program does not use, and for some purposes, the LANGUAGE utility will be useful. The LANGUAGE utility selects any one of eight language character sets *for use on the screen.* The reason for emphasising the point about the screen is that LANGUAGE does not affect the printer, so that the use of the LANGUAGE utility will nearly always cause discrepancies between what appears on the screen and what appears on the printer. The utility is brought into use, assuming that it's on the disc, by typing LANGUAGE, a space, then a number (range 0 to 7). The default character set is the US one, LANGUAGE 0. In this set, the symbols that appear on the screen correspond to the symbols on the keys – including the hashmark and the pound sign. The English set is LANGUAGE 3, and in this set, the key that is marked as the hash sign gives the pound sign, and the key that is marked with the pound sign gives the hashmark. Don't therefore assume that it's any advantage for you to go over to the English character set for any particular reason, because it's not fundamentally different, and it's hardly an advantage to have two keys whose actions are reversed. Where the LANGUAGE utility comes in useful is in preparing work for use with foreign scripts – but you need to approach this very carefully, because you will nearly always need to print the material, and you have to be certain that the printer will be using the same character set. This is difficult enough when you are using the Amstrad printer, and becomes very much harder if you are using a printer of another make. In particular, if you are using a daisywheel printer, you will need to be certain that the correct language daisywheel is in use, and that the correct codes are being sent to the printer for the different characters of that language set.

Chapter Eight
Assortment

When we deal with an operating system as old and as well developed as CP/M, there are bound to be many topics that do not fall conveniently under the headings we have used so far, or which deal with uses of utilities that are not exactly a first priority for most users. In this chapter, we shall take a look at some of these utilities and at some actions that we have passed by. You may want to pass over this chapter on a first reading of this book, or perhaps dip into one particular section of it. You are not likely to find that the utilities described here are all of equal interest to you unless you have been hooked by the charms of CP/M for its own sake. Some of the actions described here are carried out in commercial programs as part of their normal action, others might be useful additions to such programs, some are of interest only to the fairly dedicated computer devotee. What they have in common is that they are all on your Master disc sides.

Date/time stamping

A feature that is provided for in CP/M+ is the date and time stamping of files. This means that the date and time can be coded into a file so that you can check, without the need to read back an entire file, when the file was created, or when it was previously used or amended. This is very useful if you make use of a lot of files that are updated at frequent intervals. You might need, for example, to update a spreadsheet at irregular intervals rather than daily, and date stamping is very important to check how much information needs to be altered since the previous updating. Files for word-processing can often benefit from date stamping because you might want to know if a file was created before or after a change to the format of all work. Note that the whole process of date and time stamping does not apply to LOCO SCRIPT. It may seem odd that a program supplied with a machine should not be usable with so much business software and with so many CP/M features, but you need to remember that LOCO SCRIPT was designed to make the best use of the more limited memory of the PCW 8256, and so it controls the machine directly rather than through CP/M. If you need to use a word-processor along with business programs, then you should consider New Word, which is dealt with in more detail in Chapter 11.

To work with date and time stamping of files on a disc, the directory of the disc must be changed. You cannot, in other words, decide to work with stamping of just a few files on a disc. You must make sure that the disc that you are using contains no LOCO SCRIPT files, and then you must run the INITDIR utility. This is on Side 3 of the Master discs, and you can either copy it over to the disc that you want to use, or type something like:

INITDIR B:

to get the action carried out on a disc in the other drive, or another disc placed in the single drive of the PCW 8256. When you press RETURN on the command, you will get the messages:

INITDIR WILL ACTIVATE THE STAMPS FOR SPECIFIED DRIVE
Do you want to reformat the directory on Drive: B?

the latter message will specify whatever drive you have declared as being in current use for stamping. When INITDIR has run, the directory of the disc will have been changed to allow for date/time stamping, *but only if there is space on the disc.* If the disc was almost full, it may not be possible for the INITDIR program to allocate space, and if so, the program will not proceed and you will be informed with an error message. Using INITDIR, remember, only carries out the reorganisation of the directory, it does not, of itself, make any noticeable change to what the directory contains. This latter step is carried out by using SET.

SET allows you three date/time stamping commands, CREATE, ACCESS and UPDATE. You can use any one alone, or either CREATE or ACCESS with UPDATE, but you cannot use CREATE and ACCESS on the same file together. As the names suggest, CREATE will stamp the date and time of creating each new file, ACCESS will stamp each time a file is used, and UPDATE will stamp each time a file is amended. These stampings will be seen when a DIR[FULL] is requested, so that if you intend to make use of these features, you should have on the disc the utilities SET and DIR (so that you can use DIR[FULL]), and you might also want to use SHOW. If SET has not been used, then a DIR[FULL] at this stage will not reveal anything unusual. In other words, there is nothing to remind you whether you have used INITDIR or not, and it's a good idea, unless you treat all discs alike, to make some kind of recognisable mark on discs that have had their directories rearranged by INITDIR so that you know on which discs you can use SET for date and time stamping.

If your disc already contains a number of files, there is no point in using SET[CREATE] unless you are going to create a number of new files. This does, however, allow you to alter all the files you have that can be altered by ED. The reason is that using ED on a file is equivalent to a new file creation, even if you do nothing more than read the file and re-record it. If, however, the files that already exist were not created by ED, then you might want to use the ACCESS feature. This is done by typing **SET [ACCESS=ON]** – assuming that you have a copy of SET on the same disc. The visible result is the screen

```
Label for drive A:

Directory       Passwds  Stamp    Stamp    Stamp
Label           Reqd     Create   Access   Update
--------------  -------  -------  -------  -------
A:LABEL   .       off      off      on       off
```

Figure 8.1 The screen message produced by the SET[ACCESS=ON] command for a disc that has stamp access on.

message that is reproduced in Figure 8.1. This lists the actions that SET can provide, and shows that access date and time stamping has been turned on. If you now use DIR[FULL], you will see the time and date stamping for your access of SET and DIR! The only problem is that both time and date will be incorrect, because at no point so far have you entered these items. The dates are written in US style, that is in the order of month/day/year, and the dates that you see will be the dates that were originally set when the SET utility was last updated – this was 15th December 1982 on my copy. To put in the time and date for yourself you need yet another utility, DATE, preferably once again on the same disc.

As with so many of these utilities, DATE can be used in different ways, but mercifully only in two different ways. Typing DATE RETURN will give the date and time that has been entered into the code when the CP/M version was created – the age may surprise you as I have indicated above. To put in your own date and time, you have to type these items following DATE. There must be a space immediately following DATE, then the date typed as MM/DD/YY. In other words, each part of the date must consist of *two* digits, using 0 if necessary, with the month number first. This is very important, and very easy to forget if you are not accustomed to the US month-first style of writing dates. If this feature is likely to cause confusion then you may have to abandon the idea of date stamping. Following the date figures, which have to be separated by the forward slash sign, you must leave another space, then type the time. This is done as hh:mm:ss, once again using two digits for each, and separating the items with the colon sign this time. When you press RETURN, the date and time are not stamped instantly – you will get a message to the effect that the stamping will be carried out when you press a key again. This allows you to synchronise time fairly exactly if this is important. From that point on, for as long as the computer is switched on, the date and time of access of files will be recorded, and will be noted in the DIR[FULL] display.

The other options are CREATE and UPDATE. If you choose to use SET [CREATE=ON], then stamping is carried out only on files when they are created. You will, however, see CREATE stamping appear for the files SET, DATE and DIR. No ACCESS stamps can now appear on the same disc, because CREATE and ACCESS are mutually exclusive – you cannot choose to show both. You can, however, choose to combine UPDATE with either, using SET [UPDATE=ON], so that the date and time of updating any file can be recorded. Note that when you use DATE by itself, it provides the date and time as you would expect, but also shows the correct day of the week though this is something that you are not required to enter. This is because the DATE utility includes a calendar program that can find the correct day of the week for a given date.

A lot of applications programs will include date/time stamping independently of these utilities, and when you load up the utilities you will be asked to supply data and time information. If you use stamping independently for your own purposes, you will need to run the DATE utility each time you switch the machine on. This can, of course, be made automatic if you make DATE part of a PROFILE.SUB file on each disc that you use. If, on the other hand, you are using a program in which date/time stamping is included and you wish to stop using it, you *might* find that such stamping is being carried out with DATE and SET being used in a PROFILE.SUB file. If this is so, then you can easily alter it. Most programs, however, do not make use of such files because they are not intended to be used at the time when the CP/M system is loaded, and do not contain CP/M tracks. A careful study of the files on your discs, using DIR[FULL] can often be useful, however, particularly if you use TYPE to examine any .SUB or other SUBMIT files.

One of the main uses of SET with files of all types, including LOCO SCRIPT files, is to alter access to files, including directory access. The options that exist are called 'attributes', and you will often find that such attributes have been used in commercial programs. Some of these attributes can be set for the whole disc, others for specific files only. The attributes consist of DIR, SYS, RO and RW. The RO and RW attributes can be used either for the whole disc or for individual files. Using SET B:[RO] for example, will make all of the files on the disc in Drive B have read-only status – you can use the file by reading it, but you cannot write a file of that filename on the disc. This is a useful way of protecting a disc full of data files from alteration. Using the command such as SET A:[RW] will reverse the RO action, so that all of the files on the disc are available for reading or writing.

The RO and RW attributes can also be applied to named files, using commands such as SET filename[RO]. You can also make use of SET filename[SYS] and SET filename[DIR]. The SYS option makes a file into a 'system' file, meaning that it is not listed in the directory. You will find that this has been done for several files on the LOCO SCRIPT disc. Using DIR reverses this action, so that the file name can be read once again. You can combine either RO or RW with SYS or DIR, but you can't use SYS and DIR, or RO and RW together.

SET has another important use that is not possible when you use discs with LOCO SCRIPT. This is password protection, and its use is as convoluted as that of date and time stamping. Unlike date and time stamping, however, the use of password protection involves some risk of losing data. If you protect your files with a password and then forget the password, you will not be able to gain access to your files unless your knowledge of the CP/M system is rather better than you will gain from this book, or indeed from most others. In other words, the password system will not defeat a keen programmer, but it can definitely make life difficult for the user who has forgotten the magic name. This usually tempts people to use simple passwords such as their own surname or initials. Using password protection like this is quite useless, because it's the first thing that any would-be file hacker will try. If you have a need to use a password for your files, then make up an unlikely one, preferably formed from bits of other words that even a close friend would not know. For example, using ROBMITCH as a password because you admired some Robert Mitchum films is quite useful as long as no-one knows your tastes too well.

Passwording is rather pointless unless you protect everything, so in the following example, I'll run over the technique for putting a name label on to a disc, protecting it with a password, and then protecting individual files. You might want to protect only a single file, and if so the information is here, but you will have to skip over what is irrelevant, and you will find that the protection for a single file is not really adequate. The reason for going through the whole procedure is that you may not realise just how much or how little is protected at any given time.

To start with, whether you want to use protection or not, disc labelling is a useful feature of the SET utility. This allows you to put a name, not exceeding eight characters (plus an extension of three following a dot, if needed) to label the disc. This is done by a command of the form;

```
SET [NAME=discname] RETURN
```

but this will not show when you use DIR. The label shows when you use the SHOW utility in the form:

```
SHOW [LABEL]
```

when it will display the label name, the SET attributes (date/time stamping, read, write, etc), and the dates and times of creation of the label and its updating. The creation time is usually that of the CP/M version, and your date and time of labelling are shown as the update date and time.

This label name can now be protected by the use of a password. This protection is not what you probably imagine, because it does *not* prevent the label name from being displayed by SHOW. What it does is to prevent you from using SET again until the password is entered. This makes it impossible for anyone other than a skilled programmer to use SET to alter the password protection or to alter the label name. You can get this layer of protection by using:

```
SET [PASSWORD=yourone] RETURN
```

which will provide for a password of **yourone**, and you must make a separate note of whatever you have used. At first sight, the password does not appear to have much effect. You can still get a DIR listing, and SHOW still displays the label name. Even more unexpectedly, the SHOW display shows the 'Passwords required' column as OFF. This is because no file protection has as yet been used. All you have protected with the password so far is the use of SET. You will find this out the hard way if you try to remove the password protection. This is done by typing SET [PASSWORD= and then pressing RETURN – but when you do this, you will be asked for your password! Unless you can supply it, the SET action will not work. This makes it impossible to remove the password (which is why you need to note it) or to make any change to the label or to alter anything else that makes use of SET.

The next logical step is the protection of the files with this password. To prepare for this, you need to type:

SET [PROTECT=ON] RETURN

and to enter your password when requested. If you have not used a password so far, this will be unnecessary, so that you are not forced to enter a label name, nor to assign a password to it to get file protection. The SET [PROTECT=ON] option is simply an enabling command, and it does not assign a password automatically. To do this, you have to specify the files in yet another SET command. This time, the command does assign password protection to the files, and you can use a wildcard specification for the files. You can, for example, use:

SET *.* [PASSWORD=mypass]

to protect all of the files on a disc. For text files, such as .SUB files, this protection means that a command such as TYPE FILE.SUB will have no effect other than to return the prompt sign > after a short time. Using ED will cause a request for a password to appear, and without the password, the editing cannot be done. If you have protected all of your files, then you *cannot* use some of the .COM files, such as ED, and this makes the action rather risky. The sensible way to use passwording is with files that are not .COM types, and this is yet another good reason for supplying your files with extension names. If all of the files that you want to keep secret use the extension .SEC, for example, then it's easy to protect all of them with the same password. If you have to go over the files one by one, this can be tedious.

If you need to remove protection, this also can be done by using SET, but you will need to know the password, particularly for a completely protected disc. In general, using SET [PASSWORD= (then pressing RETURN) will remove the passwording from the whole disc label, providing you know the password, and SET [PROTECT=OFF] will remove protection from files. Note that the file protection can always be turned off unless you have protected the whole disc by a label password. The system is geared to avoiding loss of data unless you are really determined to do so! Figure 8.2 lists a summary of the various SET protection options. Note that unless you specify otherwise, you always get the highest level of file protection, the READ option, which protects all forms of file access.

File actions

SET option	Action
DIR	Set files to show in DIR listing.
SYS	Hides files from DIR listing.
RO	Makes file read-only.
RW	Makes file read/write.
ARCHIVE=ON/OFF	Affects .BAK file creation. For programmers only!
F1—4=ON/OFF	User-definable attributes. For programmers only!

Drive actions

d:[RO]	Set disc in named drive to read only, so that writing is impossible.
d:[RW]	Restore normal read/write to disc in drive.
NAME=filename	Label disc with this name.
[PASSWORD=word]]	Set password to word supplied.
[PROTECT=ON/OFF]	Turns password protection on or off.
filename[PASSWORD=word]	Assigns password to named file.
filename[PROTECT=READ]	Makes password necessary for all actions on file.
filename[PROTECT=WRITE]	Makes password necessary for writing, deleting, renaming, but not reading.
filename[PROTECT=DELETE]	Makes password necessary for deleting, or renaming, not reading or writing.
filename[PROTECT=NONE]	Deletes password protection for file.
[DEFAULT=word]	Assigns a default password for access to files.
[CREATE=ON]	Turns on time stamping for file creation.
[ACCESS=ON]	Turns on access time stamping for files.
[UPDATE=ON]	Turns on time stamping for update of files.

Figure 8.2 A summary of the SET protection commands. Don't be put off by the number of these, only a few are likely to be needed.

Other utilities

A considerable amount of space has been devoted to SET in this chapter, mainly because the facilities are useful and because some are not explained at length in the manual, mainly because they are not used with LOCO SCRIPT. The rest of this chapter is spent on some other utilities. A full description of all of these utilities would run into several volumes, and because they are of secondary interest to most users, as distinct from programmers, the descriptions here are limited to a few points of interest. Some well-known names, like PIP, also appear here, being used for actions that are not quite so common as those discussed earlier.

We'll start with SETDEF. The name of this utility makes it easy to confuse it with SET, but there is no resemblance. SETDEF is concerned with disc searching and with data display. One form is **SETDEF[TEMPORARY=M:]**, which will ensure that any temporary files that are created by a program such as ED will be put on to a specified drive, in this example the M drive. A more common use of SETDEF is to define the order of searching for files. For example, if you use **SETDEF M:,A:** then any file that you want to use will be searched for, first from the M drive, then from Drive A. If you use the form **SETDEF M:,***, then the M drive will be searched, and following that whatever current drive is in use. Note that when you use the wildcard * with SETDEF to indicate 'any drive', you must *not* use the colon following the asterisk. Using SETDEF can make for faster running of programs that have many separate sections located on different discs, or partly in the M drive. By placing the files in the correct parts of memory and using SETDEF, you eliminate the need to specify where files exist, and the program runs faster – particularly when a large number of files can be put into the M drive. We have already seen this type of action used in a Sagesoft program.

The other uses of SETDEF concern display on the screen. By using SETDEF[DISPLAY] the main information about a file will be displayed before it is put into use. The information includes the drive letter, the filename, any different user number, and so on. For a lot of .COM files, this information may not amount to very much, and you can suppress it again for that session by using SETDEF [NO DISPLAY]. A more useful pair of operations is SETDEF [PAGE] and SETDEF [NOPAGE]. The default action, when anything is displayed on the screen, is to page the screen, displaying one screenful of data at a time until a key is pressed. This can be a nuisance if you want to use ALT-P to print out the data, so paging can be suppressed by the SETDEF [NOPAGE] command. The paging can then be restored, which is preferable for most purposes, by using the other option.

The previous mentions of PIP have not squeezed everything possible from this remarkable utility. The main use of PIP will always be to transfer files, but some of its other actions can be useful at times. One point to note here, following the advice on passwords, is that PIP will copy a file that is password protected. You will have to specify the password following the filename, otherwise PIP will not work, and the copy that is made will also be protected by the same password. A lot of PIP actions are duplicated by other utilities, but one feature of PIP is the remarkable number of options that can be selected to be carried out during a file transfer. The options are specified by letters placed between square brackets, and separated by a single space from each other. The first square bracket must follow a drive letter or filename with no space. These option letters are noted in the manual for the PCW machines, but only briefly, and Figure 8.3 is an expanded version. Some of the options are of interest only to dedicated programmers, but others, particularly the S and Q options can be very useful for text files, allowing selective copying from a file. The transfer actions of PIP that involve the screen, printer and disc drive, however, are handled also by the use of ALT-P, and by the utilities PUT and GET.

In this table, n means a number, and s means a string of characters that ends with ^Z. The option letter command must follow the file/device name and be enclosed in square brackets. Any number must follow its command letter with no space. A set of options can be placed between one set of brackets, with the options separated by blanks.

Option	*Effect*
A	Copy only updated files.
C	Ask for confirmation.
Dn	Delete characters after number n.
E	Echo at screen.
F	Remove form-feed characters.
Gn	Work with User number n.
H	Transfer in hex-file format.
I	Ignore 00 in file.
L	Convert upper-case to lower-case.
N	Add line numbers to file
O	Transfer object-code file.
Pn	Set page length at n lines.
Qs	Quit copy action after string s.
R	Read SYS files.
Ss	Start copying at string s.
Tn	Expand tab code to n spaces.
U	Transform lower-case to upper-case.
V	Verify that data has been copied correctly.
W	Write over RO files.
Z	Zero parity bit – programmers only!

Figure 8.3 The PIP options – here again, you will probably need only a few, possibly none, of these.

GET allows a program to take any inputs it needs from a file rather than from the keyboard. If, for example, you have a program that requires you to answer a set of questions you might find that you are always typing the same answers. If these answers are put into a file, using ED, with each answer on a line of its own, then using GET can allow the answer to be fed in automatically. For example, if your answer file was ANSWIT, and you wanted to use this with a program called ALLQUEST.COM, then you could enter two lines to run the program:

```
GET FILE ANSWIT RETURN
ALLQUEST RETURN
```

and the result would be that the questions were answered automatically. You would not, of course, want every question to be answered automatically, so GET provides for this by reverting to keyboard input if either the file of answers or the program itself terminates. By an extension of this, you can specify the program itself in the file. For example, if your ANSW file contains:

ALLQUEST
Answer 1
Answer 2

and so on, then the command:

GET FILE ANSW[SYSTEM]

will get the program ALLQUEST, and then feed the answers to this program when it runs. You can specify that the input is not seen on the screen by adding NO ECHO in the square brackets, and you can revert to normal keyboard input by using the GET CONSOLE command.

PUT is a matching command which redirects any output that would normally be going to the screen or to the printer. The redirection in either case is to the disc, and a filename must be specified for the disc file that will be created in this way. The redirection will normally cease when a program stops, unless you add a [SYSTEM] command, in which case the redirection continues until you use a PUT PRINTER OUTPUT TO PRINTER or PUT CONSOLE TO CONSOLE to restore normality. This option is not one that you need very often unless you are using a program in which output to a printer or to the screen is not also recorded on disc. For such a program, you would probably want to specify [**ECHO**] in the command in order to get the normal output as well as the output to the disc. For example, to redirect screen output to the disc with ECHO, you might use a command of the form:

PUT CONSOLE OUTPUT TO FILE SAVIT[ECHO]

so that all the screen output was saved as a file called SAVIT. By adding the word FILTER between the square brackets (separated by a comma from any other word in the brackets), you can transform any ALT characters into printable characters. Remember that *everything* that appears on the screen is recorded on disc by PUT, including prompts. This can, however, be a useful way of transferring data, particularly if you have a program that produces only screen text, and for which you want to transfer the output to a word-processor file (not LOCO SCRIPT).

The HELP facility on CP/M+ is very useful, particularly if you print all the pages out and use them as a guide to CP/M+. The technique for this is to use the option LIST, which will eliminate paging, form feeds, and also unwanted line spaces. If, for example, you specify HELP LINK[LIST], you can get a printed listing of this particular HELP page by using ALT-P just before you press RETURN. For the examples that follow, you need to type **.EXAMPLES[LIST]** RETURN in the usual way. One feature of HELP that is less well known is the addition of new HELP topics. This is not exactly straightforward, but the feature is so useful that the difficulty is worthwhile. To start with, you need to have the two HELP files copied from the Master disc Side 4 on to a disc that can be written. This can be done using PIP with the command for the form **B:=A:HELP.*** (and remember that if you are using the PCW 8256 you will have to change discs three times). The files are called HELP.COM and HELP.HLP, and of the two, the HELP.HLP file contains the text. Unfortunately, you cannot edit the HELP.HLP file directly, because the HELP.COM file contains indexing information that would be upset by changes in the HELP.HLP file. You can, however, extract the data from the HELP.HLP file into another file, edit this, and then re-create the file. To start this process, you type **HELP[EXTRACT]** RETURN and wait for some time until you get the message:

HELP.DAT created.

The HELP.DAT file now contains all the data of the HELP file, but in a form that a suitable word processor (not LOCO SCRIPT) can use. You cannot use ED for writing new HELP text either, but you can use ED to view the HELP.DAT file. If you have Pocket WordStar or New Word, you can add to or edit the file as you wish, but remember that it's a very large file, and there may not be enough space on an ordinary single-density disc. The rules for editing a HELP extract are shown in Figure 8.4. When you have re-recorded the HELP.DAT file, it can be put back into correct form by using HELP[CREATE]. Your new entries will now be part of the HELP text.

The SAVE utility is one that is normally used only by programmers. Its function is to save the contents of a specified part of memory to a disc, and as such it is normally of interest only to programmers. Typing SAVE will bring the utility into action, and you are then prompted for a filename. You can change to a freshly formatted disc at this point, and then type a valid filename followed by RETURN. You are then asked for a start address in hexadecimal and, when this has been entered, and end address also in hexadecimal. The snag is that you don't normally know these address numbers, but if you can find them from user groups or from magazine articles, then you may be able to salvage work following a crash. For example, there are several documented methods of recovering work from a WordStar

1. Obtain HELP.DAT file as noted in the text. This is a very large file and probably won't fit entirely into the memory when a word processor is also present, so that it will have to be split into sections.
2. Start each topic with three backslashes ///.
3. Follow the backslashes with the topic level – 1 for a main topic, 2 for the next level down and so on. Limit of level number is 9.
4. Type the topic name, maximum 12 characters. end the line with a RETURN character.
5. Place topics in alphabetical order, and put subtopics within a topic also in alphabetical order.

Figure 8.4 The rules for editing a HELP file so that you can modify entries or add to them. A suitable word-processor is needed, and neither ED nor LOCO SCRIPT is suitable.

crash, because when WordStar crashes, text remains in the memory until it is replaced or until you switch off. If you can find the correct memory addresses to use, then, it is possible to save the text that remains, or a reasonable amount of it, and return to edit it later. The whole procedure is risky, but better than losing a lot of work. At present, however, there isn't much information available on the addresses that WordStar or New Word use in the PCW machines, so that SAVE is something to bear in mind rather than use at present.

Finally, SID is a utility that is strictly for programmers. It is for examining, modifying and working with CP/M programs, and a more detailed description of its action will be found in my book *Introducing Amstrad CP/M Assembly Language*. It can be used in connection with rescuing WordStar files, but only if you understand what is going on. One problem here is that the CP/M+ system is comparatively new, and a lot of published material refers to the older utility, DDT, which SID replaces. Older texts also show SAVE and other utilities in a different form, incidentally, referring mainly to CP/M Version 2.2 which preceded CP/M+.

Chapter Nine
Spreadsheets

The spreadsheet is one of the mainstays of business computing, a versatile form of program that is widely used, and yet not used to nearly the extent that it might be. The spreadsheet is a method of storing data that allows relationships to be worked out and displayed, and in particular, allows the effects of changes to be demonstrated. The illustrations of spreadsheet action are nearly always concerned with its use for financial planning, and to some extent this has masked the remarkable usefulness of the system for many other applications.

The comment that is most heard about spreadsheets is that their usefulness is limited only by your imagination, but this fails to grasp that the imagination needs some starting point, and unless you have worked for some time with a spreadsheet, your imagination has nothing to get hold of. In addition, spreadsheets that appear to be more or less identical when you see them rapidly demonstrated can turn out to be very different to use. In the days when a spreadsheet program could cost around £300, you made your choice and had to stick with it, but at the current price levels, with good software at £50 or less, you can afford the luxury of changing from one spreadsheet package to another once experience has shown you that you need certain features, or can't live with others. If you feel that you want to stay with your first choice, then it's important to know the differences, and this takes much more than a quick look through the manuals. Demonstrations are not always useful unless you know exactly what you are looking for, and if you knew that, you wouldn't need the demonstration! The aim of this, and following chapters is to summarise the features of the main packages so that you have at least some idea of what to look for. I must emphasise that these are summaries – it takes a long time to get to know a package really well, and some of the packages for the PCW machines are relatively new, though others are very well-established. There are choices here for a start. A new package may not have had such thorough testing and documentation as an older one. On the other hand, the older package probably hasn't been changed much for the PCW machines, and may not be able to make use of their best features. You can expect programs of US origin to print out money amounts in dollars, and make no provision for anything else, so that you will have to use SETLST and SETKEYS before running a money program. Programs written or modified in the UK will probably show the pound sign on screen and also on the printer. As always, much depends on what you want. One point to note, however, is that high price is no guide to the quality of software, though it may help in the provision of better manuals and more help at the end of a telephone line.

The spreadsheet principle

Most books and magazines assume that everyone has seen a spreadsheet in action, so that any further description is unnecessary. This is so obviously false that I can only assume that the real problem is lack of space to describe the program. If you have never needed a spreadsheet, you have probably never seen one, and if you have just started in business with a PCW computer, it's unlikely that you have spent years reading the computing magazines otherwise you wouldn't have had time to start a business. We'll start, then, with a description of what a spreadsheet is.

The spreadsheet allows you to enter data into a set of compartments, called 'cells'. The data can be text, or, more likely, numbers that represent some quantities like total stock, buy-in level, consumption per week, cost per item, and so on. This data might relate to accounts, like an accounts day-book, stock levels, class examination marks, dimensions of components, readings of radioactive isotopes near a coal mine, air pressure at different places, whatever you like. Anything that you can measure at different places, or at different times, or for different things, is a likely candidate for spreadsheet use. Point number one, then, is that the spreadsheet is a splendid display system. It is not limited by the size of your screen, because you can scroll sideways or up and down to see different parts, and some spreadsheets allow you to look at two different parts of the sheet at the same time ('windowing'). In addition, the spreadsheet can be printed, and usually in such a way that if the size of the sheet is very large, the paper output can be printed in sections that can be taped together to show the whole sheet.

The next point is that the spreadsheet can also act as a calculating system. Taking an example, if you have 2000 items of one stock item, and you know the buy-in price, you can calculate the value of the stock, and from that show how much you could have earned in interest on that capital as a measure of what it costs you to keep that stock, ignoring warehousing. Now if you have cells to enter buy-in price and cost of stock, and interest rates and cost of keeping stock, you don't need to make entries in each and every cell. This is because some items are clearly related. The capital tied up in one type of stock, for example, is equal to the number of stock items times the buy-in price for each, less any discounts. This can be expressed in a formula – how would you know it otherwise? You can enter this formula into your spreadsheet so that when you enter the number of items of some article and the cost-per-item after discounts, then the total-value cells are *filled* *automatically*. In the same way, if you enter the interest rate in another cell, then you can enter a formula that ensures that the cost of keeping stock appears automatically in another cell. Point number two, then, is that when you know relationships between items, you don't have to calculate quantities whose values depend on the sizes of existing quantities. The entry of formulae ensures that any dependent quantities, meaning quantities that can be calculated from others, are calculated automatically and displayed as fast as you can enter data.

This leads to the most significant feature of all, the 'what if?' use. Because the spreadsheet program is handling all of the quantities together, and can make calculations as it goes along, you can use it to show what would happen to a whole set of figures if something changed. Suppose, for example, that you use a spreadsheet to list all of your stock items in the way that has been illustrated, and there is a change in interest rates. Enter this new rate *once*, and the spreadsheet, if you know how to drive it, will recalculate all of your costs of keeping stock. Take another example. Suppose that your spreadsheet used for its data all the buy, borrow and sell actions that your business, I hope, makes a profit from. You would include such items as interest rates, amount borrowed, wages, overheads, taxation and so on. Now you can see the effects of changes. What happens if some enlightened government reduced taxation on imported electronic components, so that it became profitable to construct electronic equipment in the UK? What if taxes on employment were reduced? What if some government increased taxes so much that you had to lay off staff – how many would need to be sacked if the tax level rose by 10%? What if the higher rates of taxes were increased so that all of your programmers took off for Spain? The important point here is that predictions that formerly took a long time and a lot of effort can be made effortlessly and very quickly. You can know your business much better by knowing what the effect of a change would be, and you can react quicker to such changes. This is probably the most important and most-used feature of a spreadsheet program.

SuperCalc II

SuperCalc II is a well-established US spreadsheet that has now been adapted to Amstrad disc form, and which comes with a large manual and learning guide. The manual is a good one, but has not been adapted to Amstrad use, so that it refers to CTRL and ESC keys. This should cause you no difficulty if you have used the machine for a reasonable time. The problem that most beginners will have is in deciding which parts of the manual are relevant. You can, for example, skip all the items about adapting the program to your computer, because all of this has been done when the program was put on to 3-inch disc. Don't, however, skip directly over to the section headed 'Ten Minutes to SuperCalc II', until you have copied the correct form of the Master disc. The manual shows clearly that you need to use SUBMIT

MAKE8256 for this purpose if you are using a PCW machine. Until you do this, there will be no SC2 program on the disc, and you will be stuck right at the beginning of the ten-minute guide. If you *must* try it out at once, then type SC8256! Because SuperCalc II is so well-known, it has been the subject of independent books, and you will probably find *SuperCalc Prompt*, by Randall McMullan (published by Collins) very useful once you have your disc copy ready for use. One considerable advantage of this book is that it shows examples that you might not have considered, illustrating the versatility of this type of program. Even if you use another type of spreadsheet, in fact, this book can be of considerable assistance.

Figure 9.1 shows the files on the SuperCalc II disc, many of which do not appear unless DIR[FULL] is used. It is vitally important to follow the instructions relating to making a backup copy, and if you are wise, two backup copies. The important side to copy is Side 1, because the programs on Side 2 are of less use to the Amstrad user. Using SUBMIT MAKE8256, you end up with the files that are listed in Figure 9.2 on Side 1. The programs on Side 2 are much more specialised, and should not normally be needed – you might make one backup copy of these files and put it away with the original Master disc. Once you have Side 1 copied, you are ready to start experimenting with the ten-minute guide. You can also load in the existing demonstration programs. A good one to practice with is SAMPLE, and a printout from this example is shown in Figure 9.3. This shows the style of the

```
Directory For Drive B:  User  0

    Name      Bytes  Recs   Attributes       Name      Bytes  Recs   Attributes
------------ ------ ------ ------------  ------------ ------ ------ ------------
BRKEVN   CAL     5k     33 Dir RW        BUDGET   CAL     2k     11 Dir RW
CHECKS   CAL     4k     27 Dir RW        DATTIM   COM     3k     18 Dir RO
MAKE6128 SUB     1k      1 Sys RO        MAKE8256 SUB     1k      1 Sys RO
PIP      COM     9k     68 Sys RO        PR1      COM     1k      4 Sys RO
PR2      COM     1k      5 Sys RO        SAMPLE   CAL     5k     40 Dir RW
SC2      HLP    14k    111 Dir RO        SC2      OVL    20k    158 Dir RO
SC6128   COM    28k    223 Sys RO        SC8256   COM    28k    223 Sys RO
SUBMIT   COM     6k     42 Sys RO        SUPERC   SUB     1k      1 Sys RW
TENMIN   CAL     2k     16 Dir RW

Total Bytes     =   131k  Total Records =     982  Files Found =   17
Total 1k Blocks =   131   Used/Max Dir Entries For Drive B:    20/  64
```

Figure 9.1 The SuperCalc II directory, showing the files and their sizes.

```
Directory For Drive B:  User  0

    Name      Bytes  Recs   Attributes       Name      Bytes  Recs   Attributes
------------ ------ ------ ------------  ------------ ------ ------ ------------
BRKEVN   CAL     5k     33 Dir RW        BUDGET   CAL     2k     11 Dir RW
CHECKS   CAL     4k     27 Dir RW        DATTIM   COM     3k     18 Dir RO
PIP      COM     9k     68 Sys RO        SAMPLE   CAL     5k     40 Dir RW
SC2      COM    28k    223 Sys RO        SC2      HLP    14k    111 Dir RO
SC2      OVL    20k    158 Dir RO        SUBMIT   COM     6k     42 Sys RO
SUPERC   SUB     1k      1 Sys RW        TENMIN   CAL     2k     16 Dir RW

Total Bytes     =    99k  Total Records =     748  Files Found =   12
Total 1k Blocks =    99   Used/Max Dir Entries For Drive B:    14/  64
```

Figure 9.2 The files that are placed on a disc after using SUBMIT MAKE8256.

```
SUPERCALC WORKSHEET

                                   JAN         FEB         MAR         APR
NET SALES                    $1,000.00   $1,100.00   $1,210.00   $1,331.00

COST OF GOODS SOLD             $300.00     $330.00     $363.00     $399.30
--------------------------------------------------------------------------
GROSS PROFIT                   $700.00     $770.00     $847.00     $931.70

RESEARCH & DEVELOPMENT         $160.00     $176.00     $193.60     $212.96
MARKETING                      $200.00     $224.00     $250.88     $280.99
ADMINISTRATIVE                 $140.00     $151.20     $163.30     $176.36
--------------------------------------------------------------------------
TOTAL OPERATING EXPENSES       $500.00     $551.20     $607.78     $670.31

INCOME BEFORE TAXES            $200.00     $218.80     $239.22     $261.39

INCOME TAXES                    $80.00      $87.52      $95.69     $104.56
--------------------------------------------------------------------------
NET INCOME                     $120.00     $131.28     $143.53     $156.84


        MAY         JUN         JUL         AUG        SEPT         OCT
  $1,464.10   $1,610.51   $1,771.56   $1,948.72   $2,143.59   $2,357.95

    $439.23     $483.15     $531.47     $584.62     $643.08     $707.38
-----------------------------------------------------------------------
  $1,024.87   $1,127.36   $1,240.09   $1,364.10   $1,500.51   $1,650.56

    $234.26     $257.68     $283.45     $311.79     $342.97     $377.27
    $314.70     $352.47     $394.76     $442.14     $495.19     $554.62
    $190.47     $205.71     $222.16     $239.94     $259.13     $279.86
-----------------------------------------------------------------------
    $739.43     $815.86     $900.38     $993.87   $1,097.30   $1,211.75

    $285.44     $311.50     $339.72     $370.24     $403.22     $438.82

    $114.18     $124.60     $135.89     $148.09     $161.29     $175.53
-----------------------------------------------------------------------
    $171.27     $186.90     $203.83     $222.14     $241.93     $263.29


        NOV         DEC       TOTAL
  $2,593.74   $2,853.12  $21,384.28

    $778.12     $855.94   $6,415.29
-----------------------------------
  $1.815.62   $1,997.18  $14.969.00

    $415.00     $456.50   $3,421.49
    $621.17     $695.71   $4,826.63
    $302.25     $326.43   $2,656.80
-----------------------------------
  $1,338.42   $1,478.64  $10.904.91

    $477.20     $518.54   $4,064.09

    $190.88     $207.42   $1,625.64
-----------------------------------
    $286.32     $311.13   $2,438.45
```

Figure 9.3 A typical printing from SuperCalc II, in this example of the SAMPLE file. The printout is in strips that can be cut and pasted to form a wide strip.

spreadsheet and also how the program copes with printing a sheet that is much too wide for the printer set up. As you can see, the report has been printed in three strips, with the headings and months JAN to APR on the top strip, MAY to OCT on the next, and NOV, DEC and the TOTAL on the last strip. This allows you to cut the paper and rejoin it into one wide strip to reproduce exactly the appearance of the original spreadsheet on the screen. There are, of course, many other options, such as printing selected parts of the sheet, or altering the printer settings so that smaller type can be used.

The ten-minute guide is what it claims to be – a simple introduction that allows you to get the feel of working with a spreadsheet without having to immerse yourself in too much detail. Once you feel more confident with SuperCalc II, you can start to create your own spreadsheets and to record them. Don't underestimate the amount of disc memory that these take up. The SAMPLE.CAL on the Master disc needs about 5K storage on the disc, and this is a small spreadsheet, though it contains a lot of useful information such as would be needed in a summary. If you were using the spreadsheet for an accounts day-book, as might be useful for a business in which the number of transactions was too low for a full-fledged accounts system like the Sagesoft Popular Accounts, you would certainly need a larger number of cells and the amount of disc space would be correspondingly greater. It's as well to remember that most of the 'classic' programs such as SuperCalc and dBASE were originally written for CP/M systems with at least two disc drives (usually 8-inch drives in those days) and that it would not seem excessive to keep one disc for each spreadsheet data. If you are using the PCW 8512 with its second disc drive using double-density recording, then the amount of space on such a disc is sufficient to allow you to work with large spreadsheets, even if some of the SuperCalc II programs are on the same disc. For PCW 8256 users, a very useful option that is now available is an add-on 5¼-inch disc drive. This allows you to use the very low-price 5¼-inch discs (at about 75p each in boxes of 50, and easily obtained as well), so that you no longer have to hesitate about using another disc after having waited six weeks for the previous delivery. In addition, don't forget that memory space is also needed. The quick-reference card for SuperCalc II illustrates the remaining memory display with 451K of remaining memory – don't expect anything like that with the PCW machines! With the sample program in the memory, you will have about 27K free, and this amount of memory can be very quickly gobbled up if you want to work with a larger spreadsheet.

Once you have worked with the sample programs of SuperCalc II and gained some experience, you will find the answer-card that is enclosed with the manual very helpful. The answer-card is an excellent summmary of the program, with a diagram that illustrates the features of the display, and a summary of the commands. These commands are very well explained and illustrated in the manual, which is excellent. Too many manuals give a concise account of commands, but with no examples that illustrate the meaning of the commands really well. The SuperCalc II manual scores extremely well in this respect, and once you know where to find information, you can gain access to what you need very quickly. The index is not as detailed as it might be, but the organisation of the manual is so good that this is not a great handicap. You must bear in mind at all times, however, that this is a program of US origin that has not been modified in any way for use in the UK, so that you will have the problem of dollar signs in money amounts. The program is fully installed for the Amstrad machines, but no changes have been made to accommodate the use of specific keys on the PCW machines, nor for the use of the RAM-disc memory. In addition, if you want to make use of the graphics of SupercalcII, then the book *SuperCalc Prompt* has several very useful examples of how to go about this.

Scratch Pad Plus

Scratch Pad Plus is a Caxton product and, although it was originally devised by Supersoft Inc. in the USA, it has been thoroughly 'anglicised' both for such features as using the pound sign in money sums, and the use of specific Amstrad features such as the PCW key set and the use of RAM disc. The manual assumes rather more basic knowledge than the manual of SuperCalc II – for example, you need to know how to use PIP, and a beginner looking at the manual might not realise that there was no copy of PIP on the Scratch Pad Plus disc. There is no independent book dealing with the program, though much of the information in *Supercalc Prompt* is relevant. The disc includes an installation program for the CPC6128, but the instructions refer mainly to the PCW machines. If you are using a PCW machine, then you can delete the AMS6128.DEF file from your working copy to leave more space on the disc. The data disc that you use determines the size of the spreadsheet that you can work with, because Scratch Pad Plus makes use of 'virtual memory', meaning that if you need to work with a spreadsheet that is too large for the memory of the computer, the program will automatically keep unused portions on the disc. Since the part you are working with is always the part on the screen, all that you notice when you are working with a very large sheet is that commands take longer to execute, and the disc is used to a greater extent. Once again, this is all made easier if you have twin drives.

Scratch Pad Plus loads rapidly (type **SP** RETURN) and goes at once into a spreadsheet display, with no HELP option shown, unlike SuperCalcII. To get help, you need to type **/H**, but this and other commands are listed on a very convenient reminder strip that exactly fits the space above the keys on the PCW keyboard. The screen messages are also tailored to the machine, so that if you are using the PCW machines you will see references to ALT–C, ALT–Z and so on rather than to CTRL–C, CTRL–Z. One disappointment is that the arrowed keys have not already been redefined, and the manual suggests that you might like to do so with SETKEYS. You need to put SETKEYS on to your disc copy, using PIP, and you will find that a suitable file is already

```
A: SP        COM : TERM      DEF : SAMPLE    SP  : HELP      SP  : AMS6128  DEF
A: SPKEYS        : CONS      COM : CONVERT   COM : SETKEYS   COM
```

Figure 9.4 The Scratch Pad Plus disc directory, with SETKEYS added so as to make use of SPKEYS.

present in the form of SPKEYS. By running SETKEYS SPKEYS, you transform the use of the keyboard to make very effective use of the PCW keys, and eliminate many of the ALT-key combinations. The reverse side of the prompt card lists the use of the PCW machines' special keys after the use of SPKEYS. The files that are present on the disc, with SETKEYS added, are illustrated in Figure 9.4. It's rather a pity that this was not done within the program, but the reasoning is sound – the keys that are used for shifting the cursor are arranged in the way that many word processors use, and they are more likely to be familiar to the users of word processors other than LOCO SCRIPT. It is simple enough, since the SPKEYS file is provided, to make a SUBMIT file that will use SETKEYS and then load in SP.

Though there is no 'ten-minute guide' to Scratch Pad Plus, and the manual is considerably slimmer than that of SuperCalc II, the instructions for the relative beginner will allow the creation of a sheet in a reasonable time, though some experience would definitely help. The use of the keys prompt card is very helpful, and makes it unnecessary to spend much time looking for the form of commands. The disc includes a sample sheet, SAMPLE, and this can be loaded to get an impression of what working with this spreadsheet involves. Loading this sample takes quite a surprising time, and it turns out to contain a lot of detail. The sample has been reproduced here in Figure 9.5, and this was done using the paged option that allows for single sheets. The normal output would be to continuous stationery, and this is the advised option, but the use of single sheets is not great problem, with one section of the sheet being printed on each A4 sheet. As before, the sections can be joined later to form one large sheet. One point that you need to be particularly careful about when you have used the SPKEYS file to take advantage of the PCW keyboard, is the use of the EXIT key. You use this, unavoidably, when you have loaded a single sheet of paper and you want printing to resume. Striking this key at any other time, however, is an instruction to exit from the program, and though you get a chance to confirm or cancel with a Y/N option, this can be disconcerting. The main problem is that using EXIT does not save the current spreadsheet, so it would be possible to lose work unless you were aware of the problem. The SPKEYS file is shown in Figure 9.6 – to remove the effect of the EXIT key you need to kill off line 5, the line that reads:

8 N "^C"

and you can revert to the use of ALT–C to exit the program from then on. This will be effective from the next time that you start up the program from switch-on, running SETKEYS SPKEYS, and then starting SP. You can still get the EXIT effect by using the STOP key, but at least this isn't likely to be struck

(a)

```
                          YEAR-TO-DATE EXPENSES
                          =====================


                 JAN          FEB          MAR          APR          MAY          JUNE
             ----------   ----------   ----------   ----------   ----------   ----------
Salaries:    £10000.00    £10000.00    £10000.00    £10000.00    £10000.00    £10000.00
Power:          £150.00      £164.00      £145.00       £90.00       £75.00       £63.00
Insurance       £500.00      £500.00      £500.00      £500.00      £500.00      £500.00
Phone:          £100.00      £123.00      £120.00       £98.00      £132.00      £156.00
Mail:            £50.00       £57.00       £68.00       £54.00       £67.00       £52.00
Off supp:        £15.00       £40.00       £75.00       £36.00       £42.00       £25.00
Water:          £150.00      £164.00      £145.00       £90.00       £75.00       £63.00
Security:       £500.00      £500.00      £500.00      £500.00      £500.00      £500.00
Telex:          £100.00      £123.00      £120.00       £98.00      £132.00      £156.00
Emery            £50.00       £57.00       £68.00       £54.00       £67.00       £52.00


             ----------   ----------   ----------   ----------   ----------   ----------
Total/mo:     £11615.00    £11728.00    £11741.00    £11520.00    £11590.00    £11567.00
Total/ytd:    £11615.00    £23343.00    £35084.00    £46604.00    £58194.00    £69761.00
             ==========   ==========   ==========   ==========   ==========   ==========
```

(b)

```
              YEAR-TO-DATE EXPENSES
              =====================


    JULY         AUG          SEP          OCT          NOV          DEC
 ----------   ----------   ----------   ----------   ----------   ----------
 £10000.00    £10000.00    £10000.00    £11000.00    £11000.00    £11000.00
    £100.00      £120.00       £85.00       £97.00      £143.00      £155.00
    £500.00      £500.00      £500.00      £550.00      £550.00      £550.00
    £134.00      £120.00      £143.00      £145.00      £120.00      £144.00
     £54.00       £46.00       £55.00       £68.00       £60.00       £75.00
     £65.00       £45.00       £66.00       £24.00       £44.00     £1000.00
    £100.00      £120.00       £85.00       £97.00      £143.00      £155.00
    £500.00      £500.00      £500.00      £550.00      £550.00      £550.00
    £134.00      £120.00      £143.00      £145.00      £120.00      £144.00
     £54.00       £46.00       £55.00       £68.00       £60.00       £75.00


 ----------   ----------   ----------   ----------   ----------   ----------
  £11641.00    £11617.00    £11632.00    £12744.00    £12790.00    £13848.00
  £81402.00    £93019.00   £104651.00   £117395.00   £130185.00   £144033.00
 ==========   ==========   ==========   ==========   ==========   ==========
```

(c)

```
Year Totals
----------
£123000,00
  £1387.00
  £6150.00
  £1535.00
   £706.00
  £1477.00
  £1387.00
  £6150.00
  £1535.00
   £706.00


----------
£144033.00
£144033.00
==========
```

Figure 9.5 The three sections of the printout from the Scratch Pad Plus SAMPLE file. Note that the sections are printed on separate pages, labelled as (a), (b) and (c) here.

accidentally when you are using the printer.

The output options of Scratch Pad Plus include the usual output to disc, and this can be in the form that a word processor working under CP/M+ (not LOCO SCRIPT) can use. There are also options to pass files to and from other Caxton programs. The less-detailed manual Scratch Pad Plus means that you are likely to want to spend more time in experimenting than you might with SuperCalc II, but on the other hand, you might find that the closer adaptation to the PCW keyboard was of particular use to you. It's not the function of this book to express preferences, and this brief guide is not likely to give you all of the information that you might need, but from what you have read you can see some of the more obvious differences between two products that on the surface look very similar.

```
 1:    14 N "^W"
 2:    15 N "^A"
 3:    6  N "^S"
 4:    79 N "^Z"
 5:    8  N "^C"
 6:    75 N "^G"
 7:    16 N "^D"
 8:    10 N "^'£8C'"
 9:    79 S "^O"
10:    15 S "^B"
11:    6  S "^F"
12:    03 N "^'£8D'"
13:    14 S "^O"
14:    11 N "^P"
15:    23 N "+"
16:    76 N "-"
17:    7  N ">"
18:    2  N "^'£81'"
19:    77 N "^'£87'"
20:    73 N "^'£85'"
21:    00 N "^'£83'"
22:    4  N "^'£9B'"
23:    20 N "^'£9C'"
24:    12 N "^'£9D'"
25:    1  N "^'£95'"
26:    13 N "^'£8F'"
27:    5  N "^'£9E'"
28:
29:
30:    E £8D "/R"
31:    E £8C "^E^E"
32:    E £81 "!"
33:    E £87 "/L"
34:    E £85 "/U"
35:    E £83 "/C"
36:    E £9B "/J"
37:    E £9C "/W"
38:    E £9D "/1"
39:    E £95 "/H"
40:    E £8F "/P"
41:    E £9E "/S"
```

Figure 9.6 The SPKEYS file – this shows that considerable thought has gone into adapting this program for the PCW machines.

Chapter Ten
Databases

A database program is another fundamental business program that is concerned with the filing of data. At this point, it's important to be aware of the implications of using such a program in the UK. If you make use of a database to keep and process data on individuals, as distinct from stock records or other non-personal data, you may be required to register under the Data Protection Act of 1984. At the time of writing, few users of databases had penetrated the legal jargon of the Act sufficiently to discover whether they needed to register or not, and there was even a suggestion that hobby owners of computers might have to register under some circumstances. My own interpretation is that you must certainly register if your database will contain any private and personal details of individuals, particularly such items as salaries of employees and confidential information on their background and abilities. Whether you would have to register for a simple mailing list is open to question, but it's safer always to assume that the Act applies to you if you keep any sort of personal details – even a Christmas Card list! In any event, you may very well decide that all personal details on employees should be kept in card-index form, because the Act appears to apply only to such data when held as a computer file.

Having mentioned that point, on with the work. A database is the computer equivalent of an office filing system, with the advantages that a lot of hard work can be eliminated. Once data has been entered, for example, it's very easy to arrange it the way you want. You can print out the contents of the database, for example, in alphabetical order of words that are used. You might, for example, use the database to keep stock records, and you can then print out in alphabetical order of stock item names. You might, on the other hand, like to print out in increasing or decreasing order of stock value for each item, or in order of stock level so that the list became a re-ordering reminder. As is usual for computer programs of the business type, once you get to grips with a program, you tend to find more and more uses for it. Unlike spreadsheets, which were a product that burst on an astonished world in the early days of microcomputers (late 70s), databases have been around as long as we have used computers of any size, and computers were originally built for the purpose of running database programs. It's not surprising, then, that database programs have reached a level of sophistication that can be rather baffling for the beginner who does not have the advantage of having programmed large computers in the 60s. Once again, you have to select the database program that suits your needs. It's always tempting to select the simplest program that you can find, but this is not usually good advice. The reason is that your needs and expectations can expand very quickly, and the simple database that is so useful for the home-computer user will soon turn out to be quite inadequate for the business user. The restrictions are, typically, on the total amount of information, the amount of information on each record item, and the manipulations like putting into order, adding numbers and so on that can be done. By spending the same learning time on a more elaborate and general-purpose program, you could find that your immediate needs would be well catered for, and you would be able to make increasing use of the more advanced features of the program as you progressed. Programs are like computers – if you are likely to outgrow them in a year or so, they aren't good bargains.

In this chapter, then, we'll follow the scheme of Chapter 9 and look at a 'classic' database program, dBASE II that has been in office use for many years and which is now available on the Amstrad machine. dBASE II is well-known, very successful, and has been a brand leader for many years. It is an extremely powerful program, which needs to be thoroughly learned if you are to get the best from it, and several books have been devoted to the ways in which commands can be passed to dBASE II. The control that the user can exert, in fact virtually amounts to a programming language in its own right. In addition to dBASE II, we shall look at some rather different styles of databases, both of much more recent origin, from Caxton and from Sagesoft.

dBASE II

dBASE II is more than just another database program, it contains a computer command language in its own right. It allows a complete office or business management system to be set up in such a way that information is always on tap. The price that has to be paid for this is that the package is *very* large and takes some time to learn, and in this respect, you will find the book *Working with dBASE II*, by Mario de Pace (Collins) very useful to you, particularly in the early stages. When you get your Amstrad copy of dBASE II, you will find that two discs are supplied, with both sides of each disc full of program files. This is a massive piece of programming, and even allowing for the fact that it caters for a lot of actions that you might not use, and contains a lot of examples, it really needs a second drive, and is better suited to the PCW 8512 than to a single-drive PCW 8256. The first disc consists of System files, the second of 'Toolkits' and other utilities to use along with dBASE II. There's also a note in the manual to the effect that the product is not sold to you, only licenced, which makes you wonder how much you would have paid to buy it!

Figure 10.1 shows the directory readouts of the four disc sides. The number

```
A: DBASE    COM : DBASEOVR COM : DBASEMSG TXT : MENUINIT CMD : MENU     HEX
A: MENU     CMD : SCCH1001 CMD : SCCH1002 CMD : SCC25000 CMD : SCCH1003 CMD
A: SCCH0002 CMD : SCC11000 CMD : SCCH0001 CMD : SCCH2300 CMD : SCC22000 CMD
A: SCCH2200 CMD : SCCH2400 CMD : SCCH2500 CMD
```

```
A: SCC27100 CMD : SCC20000 CMD : SCC20001 CMD : SCC90000 CMD : SCC10001 CMD
A: SCC27200 CMD : SCC23000 CMD : SCCH2702 CMD : SCCH2600 CMD : SCC26000 CMD
A: SCC10000 CMD : SCCH2703 CMD : SCC24000 CMD : SCCH2704 CMD : SCCH1004 CMD
A: SCC20100 CMD : SCC00001 CMD : SCC27000 CMD : SCDEXT2  DBF : SCDEXT1  DBF
A: SCDPROF  DBF : SCDREP   DBF : SCMMEM   MEM : SCIPFINT NDX : SCCH2705 CMD
A: SCCH2700 CMD
```

```
A: ZIPIN    COM : ZIP      COM : ZSCRN    OVL : DGEN     OVL : INSTALL  COM
A: DSORT    COM
```

```
A: WELCOME  CMD : FRONB    END : FRONA    DFL : FRONIX   DBV : FRONI    END
A: FRONA    END : FRONC    END : FRONF    END : FROND    END : FRONH    END
A: FRONI2   END : FRONG    END : FRONT    END : FRONE    END : BOOKS    CMD
A: BOOKS1   CMD : BOOKS3   CMD : BOOKS4   CMD : BOOKS5   CMD : BOOKS6   CMD
A: BOOKS7   CMD : BOOKS8   CMD : BOOKS    DBF : BOOKS    FRM : BOOKS2   CMD
A: BOOK2    FRM : TITLES   NDX : AUTHORS  NDX : SUBJECT  NDX : ROOMS    NDX
A: ROOMS    FRM
```

Figure 10.1 The directory listings of the four disc sides of dBASEII.

of files is rather intimidating, but the manual shows which are the most relevant to you. The manual, incidentally, makes absolutely no concessions to the Amstrad machine, and does not mention it. You will therefore have to ignore a lot of detail about setting up the system, because this has been aimed at the user of the other machines, particularly the PC type of machine, and the larger CP/M machines. The essential files on Side 1 of the System disc are DBASE.COM, DBASEOVR.COM and DBASEMSG.TXT. The third file consists of the HELP messages, and can be omitted from a copy if you do not use the HELP facility. The MENU files that are on this disc side are also there to allow you to use dBASE II without having to learn the commands. Once again, you can dispense with these files once you have become accustomed to the system. The remaining files on this side, all starting with the letters SCC are written in the dBASE II command language, of which more later. They are for use with the MENU files, and you will probably not need to copy them unless you want to make a complete backup of all the disc sides. This is a useful system to adopt, because it allows you to place these very precious Master discs somewhere safe.

A good scheme to follow is to copy the three dBASE files into a working disc, and also the CP/M operating system (the EMS file) along with SUBMIT.COM and a PROFILE.SUB to start things off. This gives you a useful working disc, on which you can also put PIP if you want to copy other files. The amount of space that is left on the disc, however, is not really enough to hold PIP as well as anything else that you might need. The procedure is as follows. Format a new disc, and then use PIP to place the three main dBASE II files on it, using:

B:=A:DBASE*.*

this will add up to about 116K. Adding your EMS file, SUBMIT.COM, and a PROFILE.SUB file will bring you up to 163K. This gives a useful working disc that you can insert into the drive after switch on. The PROFILE.SUB file needs only to contain the line:

DBASE

unless you want to set keys or other similar actions (remember that such actions will need the presence of SETKEYS, and there isn't much room left on the disc). Once you have this copy made, it's a good idea to make yet another copy straight from this disk, using DISCKIT, so that you don't have to select files from different discs again.

With the program loaded, you will be asked to enter the date in the US form, as MM/DD/YY, so that files can be date stamped. If you press RETURN without entering this data, the date will be ignored, and the next screen message concerns the version number and date of issue, which was 2.43 and 30th April 1985 in my copy. Following this is a copyright notice, unchanged from the US version. This is followed by a period (full-stop) sign at the start of the next line. This period is the prompt sign that dBASE II uses, and you have to be careful to look out for it, because it is easy to miss. With this prompt in sight, you can then enter the commands of dBASE II, which amount to more than 80 keywords that can be used singly or in combination. Fortunately, you do not have to master all of these right away, and ten or so will serve you well at the start and for a great deal of your everyday use of the program. One particularly useful feature of dBASE II is that it will detect errors, issue messages, and allow you to make corrections without necessarily having to type a whole command all over again. You can also, at any time, get help by typing **help** following the dot, followed by the name of the command with which you need assistance. This operates only if you have the DBASEMSG.TXT file on your disc, but this is one that you will have on your working copy if you have followed the procedure above. By typing only **help**, you get a description of the HELP action, along with the list of command words about which HELP can be given. This list is lengthy, and you might like to print it out as a guide.

Using files

dBASE II is a classic type of database, which means that you have to specify in advance how you are going to keep the information. Data is grouped into records, and in each record there will be fields of data (Figure 10.2). For example, if your records concerned customers, then there would be one file – friends

Each record concerns one friend. Within the record, each item is a field such as:
1. Surname
2. Forename
3. Address1
4. Address2
5. Address3
6. Age
7. Birthday
8. Hobbies

and so on. Note that the use of three fields for address allows three lines for the address, since you can't rely on being able to use commas in the entry.

Figure 10.2 The grouping of data into records and fields.

record for each customer, and the fields would be items such as name, address type of business, value of orders to date, and so on. To set up for dBASE II, you need to specify how many characters will be used for each field. If you are stingy with this allocation, you may find that some names have to be shortened because they would otherwise overrun the field size. On the other hand, if you are too generous with your allocation, your records will take up more space than they need, and access will be slower. If you add up the number of characters for each field in a record, and add one extra for information that the program itself will put in, you have the number of characters per record.Each character needs a byte of disc memory, so if you want to keep, say, 400 records, you can multiply the character-per-record count by 400 to get the total amount of disc space needed for the records. Dividing this number by 1024 gives an answer in K that corresponds to the storage you will need on a disc. For example, if you find that all your fields add up to 67 characters, then adding 1 gives 68, and for 300 records this will need 68 × 300 bytes, 20400 bytes. This is 19.92K and since we deal with complete quantities of K on the disc, this means 20K of space, a very modest amount, but more than you could place on a disc that contained all of the dBASE II program files also, though you could easily fit it onto a disc that did not hold the EMS file, or which did not use the .TXT file for HELP. Since there may be other files on the disc that are needed to keep other data about the records, you must allow adequate space, and if you need to keep extensive records you will certainly need to use a second drive, which allows much more disc space than the Drive A on the PCW machines.

Having decided what you need to reserve for each field, you can then create a file. This requires the **create** command to be typed (lower-case or upper-case) vollowing the prompt dot, and you will then be asked to enter a filename (which can include a drive letter), and then the record structure. When the file is recorded, the extension DBF will be used for the filename. The fields are numbered, and for each field number you have to enter a field name, its type (numeric or character), its width (number of characters) and decimal places if the field is a numeric one. The name of the field must not contain characters such as the full stop or comma, and you will get an error message if you include such characters. Entering a number of decimal places mean the maximum number of digits that will follow the decimal point. For a number, the figure for field size consists of the number of figures before the decimal *and* the decimal point itself if fractions are likely to be used. For example, if you need to accommodate numbers like 23167.658, then you need to put in **n,6,3** to specify that this field is numeric, will use 5 digits and a decimal point, and will use up to three digits following the decimal point. The entry of field sizes is ended by pressing RETURN instead of typing data.

Once a file specification has been created, you can make use of the file. At any point when the prompt dot shows, however, you can leave the system by typing **quit** RETURN, which will make sure that all files are correctly recorded and closed ready for use again next time. You also get a message reminding you to back up your data file. In any case, when you want to use a file that has been named, you type **use filename** following the dot prompt. If you have forgotten what numbers you specified when you entered the field sizes, you can get this by typing **display structure**, (Figure 10.3) and if you now want to put data into a new file, or to add data to an existing one, you type **append**. This will give a display of the field names for a record, with space for each answer marked by a colon at the end of each field. My master copy of dBASE II appeared to be corrupted, because the display at this point was gibberish, and nothing could be entered into it. From experience of a different version, however, these colons are important, because if you type past them, data will be put into the next field and you will have to do some editing to

```
Structure for file:  A:FRIENDS .DBF
Number of records:  00000
Date of last update:  05/13/86
Primary use database
Fld       Name            Type  Width   Dec
001       SURNAME         C      020
002       FORENAME        C      020
003       ADDR1           C      020
004       ADDR2           C      020
005       ADDR3           C      020
006       ADDR4           C      020
007       AGE             N      002
008       TEL             C      012
** Total **                    00135
```

Figure 10.3 The result of a **display structure** command after a record has been specified.

correct this. Almost all business-orientated programs use the same pattern of keystrokes to move the cursor for editing; the pattern makes use of the ALT key with the E, S, D and X keys, as used by WordStar. You might like to reconfigure these actions to the arrowed cursor keys of the PCW keyboard.

Once a database has been created, you will want to make use of it. It's at this point that you start to appreciate how very useful and powerful dBASE II can be. Just to take one minor example, you can use **.list** to list every record of a file that you have loaded. You can also select fields – for example, you can type **.list name,age** for a list of names and ages only from a database file that contained considerably more information. You can also specify more complicated selections, such as listing names that correspond to certain age limits, like 28 to 40. Obviously, you can also change the requirements for display, and you can change the records themselves, as you want. So far, this is what you would expect of any database program.

Programming

Programming means the issuing of a set of instructions that are not obeyed at once. As applied to dBASE II, it means that a set of instructions can be stored and acted on in order when a suitable command is given. This makes it possible to leave dBASE II working on a file by itself, printing out only the items that correspond to what you have programmed it to come up with. These dBASE II programs can be stored separately on a disc, if you need to use them over and over again, or you may store them and change them to suit different purposes each time. The method that is used will look familiar if you have worked through the chapters on SETKEYS and SETLST in this book. You create a file of commands, and then order dBASE II to work on them. If, for example, you created a file of commands called **getname**, then the command **.do getname** would cause this file to be read and acted upon. To write a command file, you must use the file editor of dBASE II (obtained by typing MODIFY COMMAND), ED or any standard word processor (not LOCO SCRIPT). It's at this point that some experience helps. If you have written even very simple programs in the BASIC programming language for any computer, then you will find it quite easy to write programs for dBASE II. If you have used the command programming language of a program like the WordwisePlus word processor for the BBC machine, then again you will be at ease with the dBASE II programming language. If you have programmed in the Pascal language, you will need practically no help at all. If any sort of programming is new to you, however, you will need to proceed slowly, and it's at this stage that you will find a book such as *Working with dBASE II* particularly useful. You are not, I must emphasise, forced to work with this programming language, but when the need arises, it can save an enormous amount of time. This is particularly true if you need to use the same kind of data output many times. The advantage of programming such

```
A: CARDBOX  OVR : TERMINAL DEF : CUSTOMER FIL : CUSTOMER FMT : MAILLIST FMT
A: CARDBOX  COM
```

Figure 10.4 The disc directory for CardBox.

outputs is that it makes it possible for dBASE II to be used by anyone in the office, who need not be familiar with the programming methods and who might not be happy with all the commands that would have to be entered one by one to achieve the same result. This programmability is the feature that has made dBASE II the leading database program for users who are concerned with a lot of data which they might want to be searched and sorted in various different ways. Remember also that you have a large selection of ready-made programs on your main System disc, on each side.

CardBox

CardBox is a database program written by Business Simulations Ltd, originally in 1982, and marketed by Caxton. The disc directory is illustrated in Figure 10.4, and the files occupy about 95K of the disc, leaving room for some CP/M files if you need them on your copy. The opposite side of the disc is unused and unformatted. As usual, you should start work by making a working copy, using PIP from your CP/M Master disc. The manual starts with advice on installation of the program into CP/M machines. This does not apply to the Amstrad machines for which all the changes have already been made. You can therefore delete the file TERMINAL.DEF from your working disc – the manual refers to this file as TERMDEF, and also refers to a file TERMGEN which is not supplied on the disc for the Amstrad machines. The program is started by typing CARDBOX RETURN in the normal way. You are then presented with a pair of menus, labelled as PRIMARY FUNCTIONS and SECONDARY FUNCTIONS. You can make an arrow-head point in turn to the different options by pressing the **P** key for the primary functions, and **S** for the secondary functions. The default is the primary function of Database and the secondary function of Use. In this state, you can load in the example file that is supplied.

The file is loaded by using the F option, then typing the file name of CUSTOMER. This action does not load in the main customer file, only the subsidiary CUSTOMER.FMT file. Following this, you need to press the EXIT key to bring up a new menu, and press **G** to make the program load the CUSTOMER.FIL and start the database action. It's important to make yourself a reminder sheet on how to leave the program, because simply switching off, even if you have removed your disc, can cause problems later. With the CUSTOMER file loaded, you see on screen the first record in the file of (imaginary) customers of a software supply firm. Under the record are the menu choices that are on offer. Most of these require only one a– or two-letter entry, and the general style is that when you have to select options, as for

```
******************************************************************************
*COMPANY Jesabelle Ltd                                    DATE 14.07.82 PGE 1 *
******************************************************************************
*ADDRESS Cavendish Court                      *NME Mrs Mary Smith            *
*        10 Bensham Manor Rd                  *SAL Mary                      *
*        Tottenham Heath                      *POS Managing Director         *
*    TOWN Middx                               ********************************
* COUNTY                                      *LOB Lingerie                  *
* PSTCDE E13 5TB                              ********************************
*TELPHNE 01 356 3428                          *MGR  ACW         CAT 2 5      *
******************************************************************************
*HARDWRE Osborne 1 Zenith Monitor Epson MX80                                 *
*SOFTWRE SuperCalc WordStar T'N'G               OPS CPM                      *
*   OTHER Disks                                                              *
******************************************************************************
* Exclusion category. Account unpaid as of 12.01.83                          *
*                                                                            *
******************************************************************************


******************************************************************************
*COMPANY Etiquette and Law Plc                            DATE 02.02.83 PGE 1 *
******************************************************************************
*ADDRESS Etiquette and Law House              *NME Mr Alan Wood              *
*        29 Bardoly Place                     *SAL Mr Wood                   *
*                                             *POS Sales Director            *
*    TOWN Milton Keynes                       ********************************
* COUNTY Bucks                                *LOB Insurance                 *
* PSTCDE MK10 3TU                             ********************************
*TELPHNE 0908 618276                          *MGR ABE          CAT 1        *
******************************************************************************
*HARDWRE Osborne 1 Diablo 630 Sirius (2.4, 512K) Epson MX100                 *
*SOFTWRE WordStar SuperCalc Cardbox             OPS CPM MSDOS                *
*   OTHER                                                                    *
******************************************************************************
* Hang on in there! They want to buy 200!                                    *
*                                                                            *
******************************************************************************


******************************************************************************
*COMPANY Etiquette and Law Plc                            DATE 02.02.83 PGE 2 *
******************************************************************************
*ADDRESS Etiquette and Law House              *NME Mr Robin Davies           *
*        29 Bardoly Place                     *SAL Robin                     *
*                                             *POS Accountant                *
*    TOWN Milton Keynes                       ********************************
* COUNTY Bucks                                *LOB Insurance                 *
* PSTCDE MK10 3TU                             ********************************
*TELPHNE 0908 618276                          *MGR              CAT 1        *
******************************************************************************
*HARDWRE Osborne 1 Diablo 630 Sirius (2.4, 512K) Epson MX100                 *
*SOFTWRE WordStar SuperCalc Cardbox             OPS CPM MSDOS                *
*   OTHER                                                                    *
******************************************************************************
*                                                                            *
*                                                                            *
```

Figure 10.5 A typical CardBox printout from the CUSTOMER file.

printing, you can take as long as you like in selecting these options. Only when the EXIT key (referred to as ESC) is pressed will the program move on, and normally you are given yet another chance to escape. A sample page from the CUSTOMER file is illustrated in Figure 10.5, showing a typical use of CardBox. The format that is shown here is completely determined by the user, and the amount of typing is not so much as you might expect.

Several of the commands relating to moving a set of records up and down make use of ALT-key combinations, and you might prefer to use a

SETKEYS file in order to place these functions on to more familiar PCW keys such as the arrow keys, with or without SHIFT. Figure 10.6 shows a short file that can be used with SETKEYS in order to make use of the PCW keys to replace some of these ALT–key actions. This file allows the up and down arrowed keys to be used to select the previous (up) or next (down) record. The same keys along with SHIFT will select the first record (SHIFT-up-arrow) or the last record (SHIFT-down-arrow), and the CAN key can be used to erase an entry. The left-arrow key will cause a backspace and delete action. The use of this file with SETKEYS is not overruled when CardBox is loaded, and to my mind at least it made the program easier to use. You can, as always, make this into a SUBMIT file so that the keys will be automatically configured and CardBox run when you use a command such as SUBMIT CARD.

The various actions that are specified at the bottom of each record include just about all the needs that this type of database can supply, and are illustrated in Figure 10.7. The options are specified by using the first two letters only, and these will appear as upper-case no matter how you type them. It's possible to become rather confused by the number of these two-letter commands, and in particular when you are selecting items. Taking the CUSTOMER file as an example, you can type SE to select a number of items with common characteristics. Now at this point you are provided with a list

```
1:   14 N "^A"
2:   14 S "^R"
3:   79 N "^F"
4:   79 S "^C"
5:   75 N S "^x"
6:   15 N S "^H"
```

Figure 10.6 A SETKEYS file to allow you to make use of the cursor with CardBox.

CardBox commands – these will usually be followed by a backslash and a word to be acted on.

SElect	Find the records that contain a word or phrase.
EXclude	Remove records containing word.
INclude	Include records with chosen word.
MAsk	Find records containing word, whether indexed or not.
HIstory	Show list of selections made so far.
BAck	Go to the previous selection level.
CLear	Get back to level zero.

Figure 10.7 The commands of CardBox – the first two letters are printed in upper-case to emphasise that only these letters need to be typed.

of abbreviations, and you need to be careful about how you select them. Looking at the manual which illustrates the use of TO/LONDON to select the records that contain London as the TOWN entry, it is tempting to try CO/ LANCS to get the county of Lancs entries. You will find none, despite the fact that the first record of all is for this county. The reason is that you have taken the wrong abbreviation, and you must use CY to select county. The abbreviations appear in the same order, read left to right, as the item types in the record, read top to bottom on the left-hand side and then top to bottom on the right-hand side. Until you get used to this scheme, you are likely to make mistakes in selection. Another point is that selection is from a shrinking list unless you use the CLear command. In other words, if you have selected all the LONDON entries, you can't then select the PRESTON entries from the LONDON ones – there can't be any. The successive selections to narrow down a choice will operate a Level number that is indicated on the screen, and which will return to zero when you use CLear.

Data entry into CardBox has several parallels with the use of LOCO SCRIPT inasmuch as a file carries a template with it, and new entries will use the same template. The creation of the templates is the most difficult of the actions unless you are accustomed to on-screen drawing. One consolation is that you won't have to design too many templates, and you are not really forced to have lines of dashes and asterisks unless you really want them. The pattern of a 'card', however, can often make a considerable difference to the ease of reading, and particularly when the data is printed. If you are in a hurry, then, you can use lettering alone on your first efforts, but you will probably want to use a more graphical design later. This aspect of design would be easier if a joystick or light-pen could be used. As you enter the key words of a template, you can choose to make various words into index words. These will be shown in inverse video (black on green) on screen, and will be used in searches. You also specify for yourself what two-letter abbreviations shall be used for each item type (field) in the record. You need to specify also a letter to be used for filling space in each line of the card shape, and the start and end positions. It helps very considerably if you do your design work on paper before you start on the screen, because it's much easier to see the whole pattern when you work on paper. If you use squared paper with suitable sized squares, the design of a several cards can be worked out and the later formatting for CardBox will be much easier. If you are transferring the contents of an existing card index, you may want to preserve your existing format, and CardBox makes this relatively easy.

Magic Filer

The Sagesoft database, Magic Filer, once again performs all of the classic database operations, but in a style that is quite different once again. The directory of the disc is illustrated in Figure 10.8. About 115K of one side of the disc is used by the Magic Filer files, and the reverse side is unformatted and blank. When you add the CP/M files that are needed (as detailed in the manual), you end up with a disc almost filled (170K used) that can be used, unlike most of the previous discs that we have considered, as a 'boot' disc. In other words, you can switch the machine on, and then insert the Magic Filer copy disc into the drive in place of the CP/M Master disc. The machine will read the CP/M system tracks, perform initialising routines, and then run Magic Filer. This makes Magic Filer more convenient to run than many other programs, as befits a very new (1986) program redesigned closely around the PCW machine, but in fact, this type of action is always possible if you have enough space on a disc for the CP/M EMS program and the files of the program itself, plus a PROFILE file. The switch-on actions are determined by the PROFILE.SUB file, which is reproduced in Figure 10.9. This starts by erasing all files in the M drive, with the <Y on the following line acting as a YES reply to the question that would normally appear on the screen – about removing all matching files. The SETFONT program then adjusts the screen and printer character lists so as to match up correctly with the requirements of the program. PIP is then loaded and used to transfer five files into the RAM-disc memory, following which the MAGICFIL program is loaded and run.

When Magic Filer runs in this way, the copyright notice that appears contains your personal reference number and name. This is an anti-pirating precaution, since the program is sold with an agreement that it is a one-user copy. You should have at hand a formatted blank disc, because there is no room on the copy of the Master disc for any files. When you start using Magic Filer for the first time, you will be asked to insert this data disc in a specified drive, and allow the Magic Filer program to make a special catalogue for your files. You are then asked to provide some information for the catalogue. This starts with a general title, such as the name of your business. When you are

```
A: CRTEXTND DEF : PROFILE  SUB : MAGICFIL COM : CRT      DEF : INSTALL  COM
A: SQUIRREL COM : CRVT2880 DEF : SETFONT  COM
```

Figure 10.8 The directory for Sagesoft Magic Filer.

```
ERA M:*.*
<Y
SETFONT
PIP
<M:=CRT.DEF
<M:=MAGICFIL.COM
<M:=SQUIRREL.COM
<M:=CRVT2880.DEF
<M:=CRTEXTND.DEF
<
M:
MAGICFIL
```

Figure 10.9 The PROFILE.SUB file of Magic Filer, illustrating the sensible use of the M drive.

entering this, the delete key will move back over a character, but it is not deleted until you type another character over it. The size of the name can be practically as long as you need – there is no restriction like the eight-character name of a CP/M file. The next part of the screen is then used for a 'Page Name'. Here you might want to put Employee File, Stock Items, Parts List or whatever you want to call the file. After this part has been entered you are asked to create a menu, with numbered items. This menu will be used later to catalogue items and enter them in any order that you like. Unlike the conventional database in which you enter the items very much as you would in a card-index, and for which you would need to specify field sizes, Magic Filer uses a 'tree-and-branch' system in which each item is associated with a group, and each group with a title. It's still records and fields, but in a very different form, free-range rather than battery. Each 'menu' page that you see can be split up, with each item on the page appearing in another page for the addition of more details. You can then add 'keywords' of up to eight characters that can be used to access data as and when you need it, and these keywords can be altered as and when you need to modify them. Each page of display is separate and is numbered so that you can move from one to another very quickly if need be. The main one to remember is the first page, which is 01001, because this is your main menu to which you will nearly always want to return.

Magic Filer takes longer to become used to if you have previously used other databases, but it has a lot to offer whether you have used others or not. It can be delightfully simple to work with, particularly if you aren't really sure how you would like your information organised, and its structure is such that the time to locate items doesn't change very much as you add items to the file. The freedom that it allows can sometimes seem rather too much, making you uncertain where to start a file, but the long and extended tutorial section in the manual is very helpful. You should work through this exactly as it is printed, rather than try to short-circuit anything to create a file of your own right at the beginning.

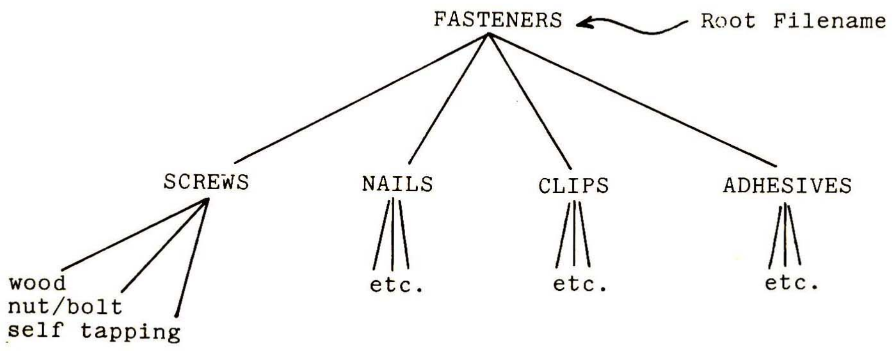


Figure 10.10 A simple illustration of the 'inverted-tree' structure of Magic Filer.

The one thing that Magic Filer lacks is a demonstration file, and this is probably because its structure means that you learn a lot more by creating a file than from simply using one. The structure also means that printing a page does not really do justice to the capabilities of Magic Filer, because several pages need to be printed to show how the information is organised. Figure 10.10 gives some idea of the principle, and is adapted from a diagram in the Magic Filer manual. The connecting links are keywords, because these allow data to be fetched and displayed from any section. Because you are not confined to any specific structure, you can add new pages as you like. For example, if your file contains a page headed Fasteners, and contains a list such as:

1. Nuts
2. Bolts
3. Washers
4. Wood screws
5. Self-tapping

then you can select item 4 and get a page headed Wood screws. You can alter this heading if you like, otherwise you can start a new list like:

1. No. 4
2. No. 6
3. No. 8
4. No. 10

ans so on. You can then pick item 2, leave the new page heading of No. 6 as it is and put in:

1. Brass finish
2. Steel
3. Chromed
4. Mirror head

and so on. The important point is that you can always create another 'branch' of data as you want, but it will be just as accessible as the rest of the data, and the use of keywords ensures that all of the data can be obtained as needed. As I said, it takes some getting used to, but it can be very rewarding, and the program is thoroughly adapted to the PCW computers, with simple commands, and remarkably few of them.

Chapter Eleven
Some Other Programs

New Word

One of the problems that you keep running into with business programs on the PCW machines is that none of the files created by these programs is compatible with the only word-processor program you have – LOCO SCRIPT. This is because LOCO SCRIPT does not make use of the CP/M operating system, and therefore its files are of a different pattern. To make use of the files of the spreadsheets and databases that we have been looking at in the two previous chapters, you need a word processor that makes use of similar files, and New Word is one such. New Word is a well-established program of US origin, available from New Star Software in the UK, in Amstrad form. The actions of New Word are very similar to those of the 'industry standard' word processor, WordStar, so that the books which act as a guide to WordStar, such as *WordStar in Action*, and *Wordstar Prompt*, both by Randall McMullan (Collins) are also good guides to New Word, though a few commands differ. The manual for New Word is very extensive, and not in any way adapted for use with the Amstrad machines. If you have any experience with your computer using LOCO SCRIPT you will find parts of the manual irritatingly long-winded, yet other parts offering insufficient explanation. It does, however, provide a lot of material, but so much that it looks very intimidating.

Because the manual does not provide any specific advice to the Amstrad owner, a few hints that are provided in note form by New Star Software may be useful. To start with, a working copy of the master New Word disc must be made. This can be done by using the Copy utility in DISCKIT, rather than by the irritatingly slow method of using PIP, particularly if you have a single drive. It's a great advantage to make your working disc one that can be used at switch-on, as you could for Magic Filer, and the procedure is as follows. The Master disc is copied, then the files checked. The files on my copy of Side A were as listed in Figure 11.1. These files take up practically all of the space on the disc, and to make a self-starter copy, you have to start by deleting some files that you are not likely to use in the working copy. These are the .DOC files, READ-ME, the .MRG files and NWNAMES.DTA. They can be erased in four commands:

ERA *.DOC
ERA *.MRG
ERA READ-ME.*
ERA NWNAMES.DTA

The next step, if you are using the built-in Amstrad printer, is to delete the printer installation program NWINSTAL.COM. If you are using a non-Amstrad printer then you need to run this program first, but it can then be erased. The next step is to remove most of the file that deals with printer driving. This is done by typing:

NWPRMAKE NWPRINT.OVR RETURN

followed by typing the numbers for the printer types. In all probability you will need only number 7, though if you intend to use a daisy-wheel you may need number 2 as well. If you are using only the Amstrad printer, thus saving a lot of money and hassle, then type 7 and press RETURN. You now need to erase the old file by typing ERA NWPRINT.BAK and then erase the editing program by using ERA NWPRMAKE.COM.

```
A: SAMPLE1  $A$ : SAMPLE1  $B$ : NW       COM : NW       OVR : NWFORM   MRG
A: NWINSTAL COM : NWLABEL  MRG : NWMSGS   OVR : NWNAMES  DTA : NWPRINT  OVR
A: NWPRMAKE COM : PRACTICE DOC : PRINTERS DOC : READ-ME      : SAMPLE1  DOC
A: READ-ME  $A$ : READ-ME  $B$ : PROFILE  SUB
```

Figure 11.1 The files on Side A of the New Word disc.

```
Directory For Drive B:  User  0

    Name       Bytes   Recs   Attributes      Name       Bytes   Recs   Attributes
------------ ------ ------ ------------ ------------ ------ ------ ------------
NW       COM     7k     52 Dir RW       NW       OVR    43k    344 Dir RW
NWMSGS   OVR    31k    247 Dir RW       NWPRINT  OVR     8k     60 Dir RW
PROFILE  SUB     1k      1 Dir RW       SAMPLE1  $A$     0k      0 Dir RW
SAMPLE1  $B$     0k      0 Dir RW

Total Bytes     =     90k  Total Records =     704  Files Found =     7
Total 1k Blocks =     90   Used/Max Dir Entries For Drive B:    10/  64
```

Figure 11.2 The directory of a working disc, ready to have CP/M tracks added.

Your working disc should now appear as in the directory illustrated in Figure 11.2, leaving quite a lot of extra space on the disc. The next steps are to add the CP/M system program, and a PROFILE.SUB file that will make the best use of the M drive. To do this, you need to copy from your System Disc, using PIP, the SUBMIT.COM, KEYS.WP, SETKEYS.COM and PIP.COM files, then the operating system, using PIP B:=A:*.EMS for the operating system. Once again, a further step is needed if you are using a non-Amstrad type of printer, which will require the presence of DEVICE.COM. You can then take the disc(s) out and reset the machine with the SHIFT, EXTRA and EXIT keys in the usual way. Do not reset the machine with a disc in place, because I for one have lost files that way. You can then type NW to enter the New Word program, and when the opening menu shows on screen, select the N option for creating a non-document file – meaning a file for use with SUBMIT in the style of an ED file. Enter as the filename PROFILE.SUB, and answer Y to the question about creating a new file.

Type the following lines as the file:

```
SETKEYS KEYS.WP
PIP M:=A:NW*.*
M:
NW
```

and then save the file by holding down the ALT key, then pressing K, then X. This is an action of a type that is often used in New Word, and is described as CTRL-KX. The important points are that the ALT key is used in place of CTRL, and the two other keys are pressed in succession while the ALT key is down, you don't try to hold all three keys down together. Now take your working disc out, reset the machine again, and insert the working disc. If all is well, you will see the initialisation carried out, and the New Word program will start itself. As a check, your directory should now be as shown in Figure 11.3. Once again, this is a well-filled disc, and you will probably want to use a separate data disc if you are going to create a document of any size.

New Word starts with an opening menu, and the important thing is to know that you return to this menu only when there is no file in the memory – it's used only when you start up and when you have recorded the whole of a document file. The screen shows also that the directory is for the M drive, and to read the PRACTICE document from the Master disc, you should first of all change the 'logged drive' by typing L, then answering the question with A , assuming you have the disc in this drive. You can then press D and you need only consult the manual for checking details. The one thing lacking is the thing that I regard as essential – a word count. I would gladly sacrifice most of the features of WordStar type programs just to have this feature which is more important to me (and to other authors and journalists) than so many other items. It appears that word-processing programs are not written by authors or journalists, however. The display of equality signs on the top status line indicates how much space a file takes up in memory, but there is nothing

```
Directory For Drive B:   User  0

    Name      Bytes   Recs    Attributes        Name      Bytes   Recs    Attributes
------------ ------ ------ ------------    ------------ ------ ------ ------------
J11CPM3  EMS    40k    320 Dir RW          KEYS     WP     1k      7 Dir RW
NW       COM     7k     52 Dir RW          NW       OVR   43k    344 Dir RW
NWMSGS   OVR    31k    247 Dir RW          NWPRINT  OVR    8k     60 Dir RW
PIP      COM     9k     68 Dir RW          PROFILE  SUB    1k      1 Dir RW
SAMPLE1  $A$     0k      0 Dir RW          SAMPLE1  $B$    0k      0 Dir RW
SETKEYS  COM     2k     16 Dir RW          SUBMIT   COM    6k     42 Dir RW

Total Bytes     =    148k  Total Records =    1157  Files Found =   12
Total 1k Blocks =    148   Used/Max Dir Entries For Drive B:   17/  64
```

Figure 11.3 The directory of a disc that can be used at start of work.

to show how much this actually is. After using ^ QP you can also get from the status line the number of complete pages, the number of lines on the last page, and the number of characters in the last line. This is quite helpful if you know from previous experience how many words you get per page and how many words per line. What you can't do is to use the WordStar dodge of calling up an alternate status line with the number of characters in the file, and dividing this by six, because New Word does not possess this alternate status line.

Communications programs

The categories of Spreadsheet, Database, and Word Processor cover the three main areas of business use, apart from communication. The use of communications programs, such as the excellent Chit-Chat from Sagesoft, has been omitted here for two reasons. One is that suitable hardware is needed in the form of a modem for transforming the computer signals into a form suitable for transmission along telephone lines, and vice versa. This hardware is available but at present the telephone system on which it can work most smoothly is not. Though many computer users make use of the telephone system in the UK, they cannot use the features that are commonly employed on modems of US origin, and this is a severe handicap. The other point is that the use of the telephone system for communicating data is expensive, and particularly slow because of the time that is needed. In addition, any system that is kept connected to the telephone lines with the modem switched on is vulnerable to hacking from other users, so that the use of telephone lines is out of the question for data that you want to remain confidential. My experience of communication in this way has not been good, and it may be that I am painting a gloomy picture which might not apply in other regions. Certainly the sales of modems and communications programs suggest that this method of communication is widespread. My own view is that lengthy text can be communicated more easily by making a disc and posting it by Royal Mail Special Delivery. At least that way I don't have the frustration of having to dial a number every minute for half-an-hour until I can get a clear line, and then tying up the computer for an hour or more while the text is transmitted!

The Ideas Processor

A class of program that is now beginning to gain acceptance as part of a business suite of programs is the type generally named the Ideas Processor. The principles are not completely new, and the description in the previous chapter of Magic Filer is a good introduction to the ideas behind the technique. The principle is to be able to input words and phrases which the program will then assemble into some sort of coherent order. Obviously, this would be of little interest if you were preparing a single-page report on something trivial, but for complex and lengthy works, the system can be very valuable. As always, it's up to the user to be imaginative, and the program does the donkey work. One of the best-known of these programs is Caxton's BrainStorm, written by BrainStorm Software Ltd and now available in Amstrad PCW form.

The BrainStorm disc contains only three filenames, one of which is a sample, with only 22K of disc space used. As usual, the files can be copied on to a blank formatted disc using PIP. The copying instructions with my BrainStorm disc referred to the copying of Scratch Pad Plus, but the technique is the same in any case. The manual suggests that you make a bootable copy, meaning one that you can use by inserting the disc after switching the machine on. The method is not shown in the manual, but is the same as we have seen for other programs. Having copied the BRAINSTORM files, you place the Master CP/M disc in the drive (or Drive A for a twin-drive machine) and load PIP. You then use:

```
B:=A:*.EMS
```

to copy over the CP/M operating system. Follow this with:

```
B:=A:ED.COM
```

to make a copy of ED, which you need to create a PROFILE.SUB file, and then use:

```
B:=A:SUBMIT.COM
```

to copy the SUBMIT program. You can now use ED to create a file called PROFILE.SUB. This file consists of one line only, BRAIN, which will call up the BrainStorm program. You can then erase ED, unless you feel that you might later want to create SETKEYS or SETLST files. The disc will now operate as an Early Morning Start (EMS) or boot disc – in other words when you put it into the drive (Drive A) after switching on or after a restart using SHIFT-EXTRA-EXIT, it will load in the CP/M system and then BrainStorm automatically.

Your copy of BrainStorm will be serial numbered on screen, and my copy carried a 1983 copyright notice. Pressing RETURN on this opening announcement brings up the main menu, which is simple and straightforward. One particularly useful point is that you have easy access to choice of disc drive and to the directory. You can choose to have all directory items, or to restrict the directory display to selected items, such as the .BRN files. To see the sample in action, you can load in the SAMPLE.BRN file. This requires you to press the L key to start the loading operation, and then specify the filename. The method is slightly unusual – you are asked first to type the main name, such as SAMPLE, press RETURN, and then type the extension, BRN. You then have to press Y to confirm, and using any other key will return you to the menu. The bottom line under the menu reports that a file has been loaded, and the menu returns. To make use of the file, you now press U, which brings up a new menu. This allows you to enter text (not yet) or to see the range of commands by pressing !. The use of the ! key then lists the command keys (and a misspelling!) so that you can look through the sample file. The manual at this point refers to a pull-out reference card for these commands which was not supplied with my copy of BrainStorm. The command keys use ALT and letter combinations, and you might want to make your copy conform to PCW keyboard use by allocating some of these ALT key commands to other keys such as the arrow keys. In particular, you'll find that the DEL key produces a zero, and this can be very irritating, because ALT-A never sounds like an obvious key combination for deleting.

The difficulty now is to see what the demonstration file consists of! To do this, you need to return to the first menu by pressing the EXIT key. When the first menu shows, use the P key to print the demonstration file. This consists of reminders about the use of BrainStorm, and it's best printed on continuous stationery. The first response is to ask how much to print, and pressing RETURN causes all of the file to be printed. You are then asked for a printing format, which means how different lines of the text are displayed. Pressing the RETURN key once again gives the default setting of a sawtooth format, with successive lines indented. The smallest print size of the Amstrad printer is used, and you will find it easier to read if you switch to high-quality when you load the paper at first. Though the screen display shows the message 'Hit VDU Spacebar to Pause', don't depend on this to be able to stop the printer, because the text in the memory of the printer will continue to be printed. If you want to use A4 sheets, then select the option P when you are asked about formats. The text will then be arranged in little paragraphs with no indentation and into A4 pages, but the printing will start with a form-feed so that your first sheet will be fed out unprinted. The demonstration file includes a personal database diary section which can serve as a very useful combination of demonstration and practical program.

Having printed out the file, it's then easier to see how it can be used. Pressing the U key again returns to the Use menu, and this time, the messages on the top of the main window make more sense – they refer to the two sections of USE commands summary and the Personal Database. By using the ALT-E and ALT-X keys, you can move the pointer to the listed files. The default is to create a new entry, which is why you can't see anything listed when you try to use the U key at first. Try moving the cursor to the Personal Database line, and then pressing ALT-R. This action is described as Promote, which doesn't exactly do a good job of informing you what will happen when you use it. The ALT-R action, in fact, displays one level of the database structure, so that you can see the subdivisions. At any point, you can add data as you choose, delete, amend and so on. Demoting a heading with ALT-C provides the opposite action, moving back to the more important headings. Try this to get a flavour of the system. Move to Personal Database and press ALT-H. When the prompt appears, type your birthday in the form 11-oct, 06-Jun, or whatever the date happens to be. You can use upper-case or lower-case letters as you wish. The date will then appear as a heading, and you are asked to confirm that this is what you want by pressing the Y key. You can now type the words My birthday, press RETURN, and use ALT-Q to get back to the first page. Now select Personal Database again, and try hunting for the entry. You can press ALT-H for this and use either the date to find the entry or the entry to find the date. To use the date following ATL-H is straightforward – you just type 11-oct or whatever you chose originally (not oct-11, which won't work), and the phrase 'My birthday' will appear with the heading 11-oct above it on the title line. Hunting for the entry is not quite so straightforward. This time following ALT-H, you need to enter something like ***my birthday***, or ***my*** if there are no other entries that start with my. The asterisk, which is used as a wildcard, is necessary at the start of a request like this because it is usually impossible to match correctly otherwise. Using an asterisk at the end of the sequence of letters allows you to type only part of the phrase that you are looking for. Having located the phrase, My birthday, using *my* or *My*, then you can get the date as a title by using ALT-C.

As usual, this can be only a very brief introduction to a fascinating program that can have uses that considerably transcend this trivial demonstration. The manual is not really detailed enough to open up the possibilities of the program to anyone who has not used databases before, and this makes a weekend of familiarisation essential before you really decide on whether this might be a necessary addition to your collection of programs. The ease of use and the speed of response are two very useful factors to consider, and with such a slim manual, someone is almost certain to write a book about it soon!

Chapter Twelve
Other Printers

The built-in printer of the Amstrad PCW computer is the one that you will normally use for most purposes, but for some purposes you may wish to use another type of printer. This cannot be done by using any form of direct plug-in, because the printer of the PCW machines is very closely integrated with the operation of the computer. To make use of another printer, then, you need to buy a special interface which will allow you to use either serial or parallel printers. I think it's worth stressing that unless this is really important, it's not likely to be worth the time and effort that you will spend, because the Amstrad printer is adequate for most business purposes. Certainly it would be ill-advised to go to all the trouble of installing another printer of the same general type (dot matrix) unless you had damaged the Amstrad printer and happened to have, say, an Epson RX-80 lying about unused. With that in mind, we can examine what is available in the way of printers for business applications.

Printer types

The PCW printer interface includes the almost-universal Centronics connection for printers, including the Epson, Oki, Juki and most others. It's difficult to imagine any 'serious' computer without a Centronics interface, for this is the connection method that is used by all the famous-name printers which are available. This means that you can attach almost any good-quality printer that you like to the PCW machines if you have the interface connected. Printers that are used with small computers will use one of the mechanisms that are listed in Figure 12.1. Of these, the impact dot-matrix type is the most common, and this is the mechanism of the Amstrad printer. A dot matrix printer creates each character out of a set of dots, and when you look at the print closely, you can see the dot structure. The printhead of the dot matrix printer consists of a set of tiny electromagnets, each of which acts on a set of needles that are arranged in a vertical line (Figure 12.2). By firing these needles at an inked ribbon which is placed between the head and the paper, dots can be marked on the paper. Each character is printed by firing some needles, moving the head slightly, then firing another set of needles, and so on until the character shape is completely drawn (Figure 12.3). Using 9×9 (nine

Dot matrix
- impact
- thermal
- electrostatic

Type impact
- type stalk
- daisywheel
- thimble
- type band

Plotters
- graphics printers
- X-Y plotters

Ink jet
- single colour
- multicolour

Laser
- laser fast printer

Figure 12.1 The various types of printer mechanisms that are available.

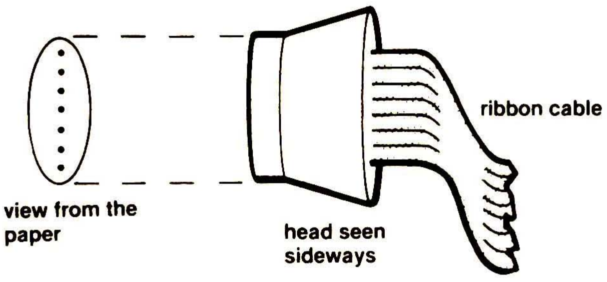


Figure 12.2 The principle of a dot matrix printing head. This consists of a vertical line of needles, each separately controlled. This is why vertical columns have to be specified in creating graphics patterns.

needles, nine steps across) or 15×9 heads can create good-looking characters, lower-case or upper-case. Another advantage of these dot matrix printheads is that the characters are not limited to the ordinary letters of the alphabet and the numbers. Foreign characters can usually be printed, and it is possible to print Arabic script, or to make up your own character set for example. We have seen earlier in this book how graphics patterns can be produced by making use of the dot patterns.

The ultimate in low-cost print quality at the moment of writing is provided by the daisywheel printer. This uses a typewriter approach, with the letters and other characters placed on stalks round a wheel. The principle is that the wheel spins to get the letter that you want at the top, and then a small hammer hits the back of the letter, pressing it against the ribbon and on to the paper. Because this is exactly the same way as a typewriter produces text, the quality of print is very high. It's also possible now to buy a combination of typewriter and daisywheel printer. This looks like a typewriter, with a normal typewriter keyboard, but has an interface connection for a computer. You can use it as a typewriter, and then connect it to the computer and use it as a printer. Machines of this sort are made by leading typewriter manufacturers such as Silver Reed, Brother, Triumph-Adler, Smith-Corona, and others. If you need a typewriter as well as a printer, then this type of machine is an obvious choice. A point to watch out for, however, is ribbon costs. Just to give you an example, I have an electronic typewriter which uses ribbons that cost twice as much as the ribbons for my Juki daisywheel, and last for one fifth of the number of characters! One of the great advantages of using machines in the Epson/Juki class is that ribbon costs are as low as you are likely to find. Looking further ahead, laser printers are now available at very high prices which provide quite superb quality of printing at very high speeds. Their prices at present rule them out for ordinary home use, but for even a comparatively small office they have a lot to offer when connected to the PCW machines.

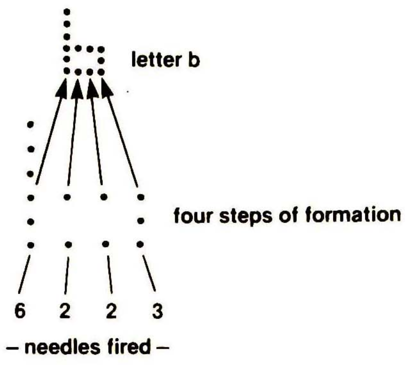


Figure 12.3 How a character shape is drawn by a dot matrix head. the head is shifted by a small amount for each part of the character, and the appropriate needles are fired against the ribbon. By making the shift distance smaller, denser images can be obtained for bold type.

Interfaces

The printer has to be connected by a cable to the computer, so that signals can be passed in each direction. The computer will pass to the printer the signals that make the printer produce characters on the paper, but the printer must also be able to pass signals to the computer. This is because the printer operates much more slowly than the computer. Unless the printer contains a large memory 'buffer', so that it can store all the signals from the computer and then get to work on them at its own pace, some sort of 'handshaking' is needed. This means that the printer will accept as many signals as its memory will take, and then will send out a signal to the computer which makes the computer hang up. When the printer has completed a number of characters, (one line, one thousand, or possibly just one character), it changes the 'handshake' signal, and the computer sends another batch. This continues until all of the text has been printed. This can mean that you don't have the use of the computer until the printer has finished. Printers can be very slow, particularly daisywheel and plotter types. Even the fastest dot matrix printers can make you wait for a minute or more for a listing. For this reason, it's possible to buy additional buffer memory for many printers that allows them to get to work on a piece of text and leave the computer free. This usually entails using continuous stationery, however, because very few buffer systems allow for the printing to be interrupted when you want to change paper.

Two types of interface are used by practically all printers. These are classed as serial or parallel. A parallel printer, such as the Amstrad printer, is connected to the computer by a cable which uses a large number of separate strands. Since each character in ASCII code uses seven signals, the parallel printer sendss these along seven separate strands – many printers can use an eighth signal and this is often sent as well. In addition, there are cable strands for the 'handshake' signals. The best-known, and most-used variety of parallel connection is called 'Centronics' after the printer manufacturer which first used it.

The serial interface sends the signals out one at a time. This means that at least seven signals have to be sent for each character, and in practice the total must be ten or eleven, to allow for 'start and stop' signals which are used to mark where the signals for each character start and stop. This system uses less cabling, because only two strands need to be used for signals, and the cables can be longer, because there's no risk of one signal interfering with another. The standard system is called 'RS232'. Printers can be obtained with RS232, but seldom as standard, and often only as an extra, costing up to £50 extra.

A problem that you are bound to run up against when you use any non-Amstrad printer is that of line feed and carriage return. A lot of computers send out only one code number, the carriage return code (13) at the end of a line. Other machines send both the line feed (code 10) and carriage return codes. Printers are arranged, therefore, so that either possibility can be catered for by a switch. If you connect your printer and find that everything is printed on one line, then don't return the printer. Just look in the manual, and find out the details of the switch that alters the line-feed setting. If, on the other hand you find that each line is double-spaced, then this switch will have to be set to the opposite position. Note that the printer must be switched off when you make this adjustment. Apart from any other considerations, the changes do not take effect until the printer has been switched off and on again. This is a safety precaution to prevent a piece of printing from being ruined by a thoughtless piece of switching! Since the Amstrad printer is of the dot matrix type, however, there is little point in considering using another type of dot matrix printer with the computer, and we'll concentrate here on the use of a daisywheel printer, typified by the excellent Juki model which has been available for many years.

The Juki 6100 daisywheel

The Juki was one of the first low-cost daisywheel printers to become available. Like most printers, it comes with a Centronics parallel interface, though an RS323 serial interface is available at extra cost. The Juki is a large and very heavy machine which can accept paper up to 13-inches wide. The daisywheel is of the same type as is used on Triumph-Adler printers, and in my experience its life is very long. The ribbon cartridge is an IBM Selectric 82/C type. The ribbon that was supplied with my Juki was of the 'single-strike' variety, and this had a very short life (about three chapters of this book!). A 'multistrike' type of ribbon is much better. With this type of ribbon, the whole width of the ribbon is used by moving the cartridge up and down as well as by moving the ribbon itself. These ribbons are very easy to obtain from a lot of suppliers, but the best prices I have seen have been in the INMAC catalogue. The ribbons are carbon film rather than inked nylon, and are thrown away after use.

The printhead of the Juki will print in either direction, and there is a 2K buffer. This means that short pieces of text can be transferred to the printer buffer almost instantly, and the computer can be used for other purposes while the printer gets on with the printing actions. Printing is much slower than the draft rate of the Amstrad, but not so much slower than the high-quality mode of the Amstrad as to make the daisywheel seem irritatingly slow. Its enormous advantage is the quality of the type. This is exceptionally clear on the top copy, and even three carbons later it is still very legible. For any letter work, or for the manuscript of a book, the Juki is ideal.

As you would expect of any modern design of printer, the Juki permits a lot of character sets, but you need to have the appropriate daisywheels fitted for each language. You cannot, for example, have words in alternate character sets without changing wheels in between. Changing wheels is particularly simple, but this is something that you don't have to worry about with dot matrix printers, because the same dot matrix head can produce any

Each of these codes will be preceded by CHR$(27).

Code	*Effect*
1	Set horizontal tab (HT) at present position.
2	Clear all tabs.
3	Graphics mode on (C/R clears).
4	Graphics mode off.
5	Forward print on (C/R clears).
6	Backward print on (C/R clears).
7	Print suppress on (C/R clears).
8	Clear present HT stop.
9	Set left margin at present position.
0	Set right margin at present position.
CHR$(9)	Set HT (tab number follows).
CHR$(10)	Set lines per page (number follows).
CHR$(11)	Vertical tab (VT) set (number follows).
CHR$(12)	Set lines per page (number follows).
_	Sets VT at present position.
CHR$(13)P	Remote reset.
CHR$(30)	Sets line spacing (number follows).
CHR$(31)	Sets character spacing.
C	Clears top/bottom margins.
D	Reverse half-line feed.
U	Normal half-line feed.
L	Sets bottom margin at present position.
T	Sets top margin at present position.
Y	Special character.
Z	Special character.
H	Special character (new paragraph symbol).
I	English pound sign.
J	Diaeresis mark.
K	Spanish c with cedilla.
/	Automatic backward print.
\	Disable backward print.
S	Set character spacing.
CHR$(26)A	Remote error reset.
CHR$(26)I	Initialise printer.
CHR$(26)1	Status (serial interface only).
P	Proportional spacing on.
Q	Proportional spacing off.
CHR$(17)	Offset selection.
E	Underline on.
R	Underline off.
O	Bold print on (C/R clears).
W	Shadow print on (C/R clears).

&	Bold or shadow print off.
%	Carriage settling time.
N	Clear carriage settling time.
CHR$(8)	1/120 inch back space.
X	Cancels all word processing modes except proportional spacing.

Figure 12.4 The codes that are used to control the actions of the Juki daisywheel printer.

character under software control. The Juki allows underlining, bold type, and shadow type in addition to the normal printing style, and you can select your print style from a range of at least fourteen daisywheels. The daisywheels are expensive in comparison to others on the market, but ribbons are cheap. By removing the top cover, you can gain access to a set of miniature switches. Switch No. 1 controls auto-line feed, and for use with the PCW machines, this must be set to the OFF position. This will give correct line-spacing – the ON position causes each line to be double-spaced. The switch-change must be done with the machine switched off. This is not so much because of risk but because these switch settings have *no* effect until the machine is switched off and then on again.

Like the Amstrad printer, the Juki permits a number of changes to be made simply by sending control codes to the printer. These use the ESC character, followed by one more character, so that whatever immediately follows ESC is never printed. The options include graphics mode, left and right margins, lines per page, half-line feeds in either direction (for printing subscripts and superscripts), top and bottom page margins, and some special characters, including the English pound sign. Even more usefully, the print can be changed to bold or shadow by sending such codes, and text can be underlined. Figure 12.4 lists these actions. If you are using New Word, then the printer driver for the Juki is the same as for the Diabolo.

The same quality of print can now be obtained from a large number of daisywheel typewriters, and many of these can now be obtained with a Centronics parallel interface. This type of machine offers a lot of advantages, because it can be used as a typewriter for small items that do not justify the use of the computer, yet is available for word-processing use along with the computer and such programs as New Word. These machines can now be bought in the 'high-street stores' as well as from office supply shops. The only thing to watch is that replacement ribbons and daisywheels are obtainable from several different sources. There's nothing worse than being stuck with a machine for which you can get spares from only one supplier.

Appendix A
ALT—Key Codes

The ALT key of the PCW machines is used to a very large extent in programs for the machine. In the manuals, it is generally referred to as the CTRL key, because this is the labelling that is on practically all other computers – including other Amstrad machines. The ALT key also affects the CP/M system itself, so that if a program has not altered the key definitions, the following effects can be achieved. In each case, the use of the ALT key is denoted by the carat (^) symbol. This appears on the screen as an up-arrow, but is shown as the carat sign when printed on paper. When an ALT-key combination is being used, neither the symbol nor the key letter will appear on the screen. These key combinations, singly or in groups, can be used in key redefinitions, making use of the Extra-U key combination to give the ALT symbol in files for use with SETKETS.

^A	Move cursor one character left.
^B	Move cursor from one end of the line to the other.
^C	Stop program when prompt shows, or following ^S.
^E	Take a new line on screen, but not in text.
^F	Move cursor one character right.
^G	Delete character under cursor.
^H	Backspace and delete.
^I	Tab along eight spaces.
^J	Line feed.
^K	Delete from cursor position to end of line.
^M	RETURN action.
^P	Toggle printer on or off.
^Q	Restore scrolling (after ^S).
^R	Put characters to the left of cursor on a new line.
^S	Stop screen scrolling.
^U	Show characters to left of cursor.
^W	Recall previous command line.
^X	Delete from cursor to start of line.

Appendix B

Languages Under CP/M

When you use CP/M on your Amstrad PCW machine, you can choose to write your own programs in the very good Mallard BASIC that is provided with the Master disc set, and this language is completely compatible with CP/M. You can also load in other languages providing that you can get hold of a copy on the Amstrad 3-inch disc. There is little point in buying the type of BASIC that is described as interpreted, because this type of BASIC must be present in memory in order to run any program that it has created, and you already have this in the form of Mallard BASIC. By contrast, a BASIC that is described as 'compiled', and which is specifically described as creating CP/M .COM files can create programs which you can later run without the language file being present, just like any other COM file. At the time of writing, no BASIC of this type was available on the Amstrad disc size, though many versions are available if you are one of the fortunate users who have 5¼-inch disc drives connected to your machine.

There is one very good higher level language, called C, which is available in a form which will generate CP/M .COM files for the Amstrad machines. This C Compiler is obtainable from Hisoft of Dunstable (look for their advertisements), and if you have already bought their earlier C Compiler for AMSDOS, an upgraded version is available. The use of this C Compiler is a much easier path to generate really large CP/M programs than programming directly in Assembly language. Hisoft also market a version of Pascal which is, for business purposes, a better programming language than C.

Appendix C

ASCII Codes in Hex

No.	Hex.	Char.	No.	Hex.	Char.
32	20		80	50	P
33	21	!	81	51	Q
34	22	"	82	52	R
35	23	#	83	53	S
36	24	$	84	54	T
37	25	%	85	55	U
38	26	&	86	56	V
39	27	'	87	57	W
40	28	(	88	58	X
41	29	)	89	59	Y
42	2A	*	90	5A	Z
43	2B	+	91	5B	[
44	2C	,	92	5C	\
45	2D	-	93	5D	]
46	2E	.	94	5E	^
47	2F	/	95	5F	_
48	30	0	96	60	`
49	31	1	97	61	a
50	32	2	98	62	b
51	33	3	99	63	c
52	34	4	100	64	d
53	35	5	101	65	e
54	36	6	102	66	f
55	37	7	103	67	g
56	38	8	104	68	h
57	39	9	105	69	i
58	3A	:	106	6A	j
59	3B	;	107	6B	k
60	3C	<	108	6C	l
61	3D	=	109	6D	m
62	3E	>	110	6E	n
63	3F	?	111	6F	o

64	40	@	112	70	p
65	41	A	113	71	q
66	42	B	114	72	r
67	43	C	115	73	s
68	44	D	116	74	t
69	45	E	117	75	u
70	46	F	118	76	v
71	47	G	119	77	w
72	48	H	120	78	x
73	49	I	121	79	y
74	4A	J	122	7A	z
75	4B	K	123	7B	{
76	4C	L	124	7C	|
77	4D	M	125	7D	}
78	4E	N	126	7E	~
79	4F	O	127	7F	

Appendix D

Additional Reading Matter

Amstrad Word Processing on the PCW 8256
Ian Sinclair
0 00 383328 3
This book enables beginners to learn professional word processing techniques really quickly without having to wade through long manuals. A special feature of the book is the summary pages section which details the commands and procedures most used, providing a useful prompt to both beginners and more experience users alike.

Introducing Amstrad CP/M Assembly Language
Ian Sinclair
0 00 383309 7
Written especially for newcomers to CP/M, this book deals with the CP/M structure and how simple CP/M machine code programs can be written with the assembler editor package supplied with the machine. Numerous useful routines and listings are also provided.

SuperCalc Prompt
Randall McMullan
0 00 383004 7
This is a rapid guide to using the popular spreadsheet packages – SuperCalc, SuperCalc 2 and SuperCalc 3. Beginners can use the unique prompt pages to start using a spreadsheet straight away. More experienced users can use these prompt pages to find commands and sequences rapidly. The second part of the book contains practical model spreadsheets useful in business, home and college.

WordStar Prompt
Randall McMullan
0 246 12446 6
WordStar Prompt helps you master the world's most popular word processing package without the tedium of wading through long instructions. The prompt pages in the first part of the book enable beginners to start word processing straight away. If you have used WordStar before then the prompts

Index

The very successful Amstrad PCW 8256 and PCW 8512 are well-known as word processors, but are also the most powerful computers ever offered at such low prices. As a result, owners can use programs – such as SuperCalc and dBase II – that have previously needed large and expensive machines to run. The key to this computing power is the use of the CP/M Plus operating system, a new version of a system that was originally written for professional programmers and users.

This book shows which parts of the manual are relevant to your needs, and how the command words of CP/M apply to the kind of work that you are likely to do. Even more important, it shows in clear detail how you go about taking control of the printer and the keyboard when you are using the machine as a computer, as distinct from using it as a word processor. It does not aim to make you a programmer, but shows how much more profitable use you can get from the machine when you understand how the system operates.

Whatever your computing requirements, this book will help you, and show how classic office software can be used to your advantage. Many useful examples and illustrations are provided, and the book is written in a straightforward style, free of computing jargon.

THE AUTHOR

Ian Sinclair has been a microcomputer user for many years, and has taught computing and programming to both beginners and more advanced users. He has written many books, most of which have been concerned with the small computers that have revolutionised home and office alike.

Cover photograph courtesy of Amstrad Consumer Electronics plc

ISBN 0-00-383359-3

9 780003 833591

£8.95 net

COLLINS

Printed in Great Britain