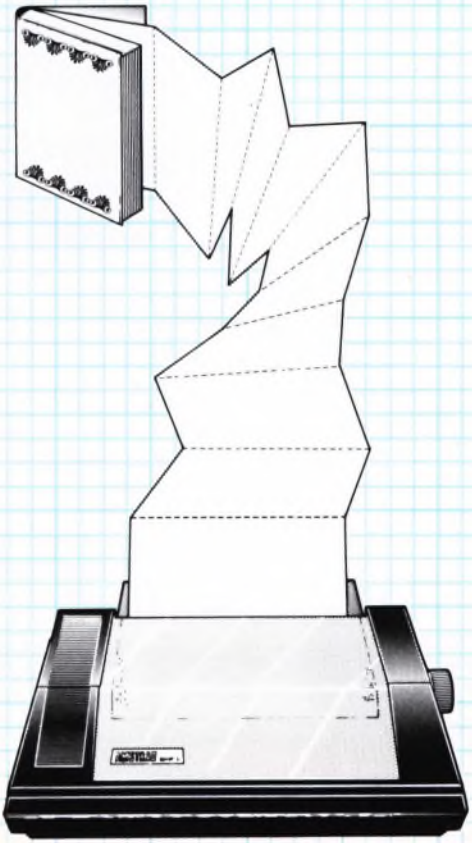
Simple Applications of the Amstrad CPCs for Writers

W. SIMISTER



SIMPLE APPLICATIONS OF THE AMSTRAD CPCs FOR WRITERS

ALSO BY THE SAME AUTHOR

BP157 How to Write ZX Spectrum and Spectrum+ Games Programs
BP159 How to Write Amstrad CPC 464 Games Programs
BP175 How to Write Word Game Programs for the Amstrad CPC 464, 664 and 6128

ALSO OF INTEREST

BP153 An Introduction to Programming the Amstrad CPC 464 and 664
BP189 Using Your Amstrad CPC Disc Drives
BP152 An Introduction to Z80 Machine Code

SIMPLE APPLICATIONS OF THE AMSTRAD CPCs FOR WRITERS

by

W. SIMISTER

BERNARD BABANI (publishing) LTD
THE GRAMPIANS
SHEPHERDS BUSH ROAD
LONDON W6 7NF
ENGLAND

PLEASE NOTE

Although every care has been taken with the production of this book to ensure that any projects, designs, modifications and/or programs etc. contained herein, operate in a correct and safe manner and also that any components specified are normally available in Great Britain, the Publishers do not accept responsibility in any way for the failure, including fault in design, of any project, design, modification or program to work correctly or to cause damage to any other equipment that it may be connected to or used in conjunction with, or in respect of any other damage or injury that may be so caused, nor do the Publishers accept responsibility in any way for the failure to obtain specified components.

Notice is also given that if equipment that is still under warranty is modified in any way or used or connected with home-built equipment then that warranty may be void.

All the programs in this book were written and tested by the author using a model of the Amstrad CPC464 and DMP1 or DMP2000 printer that were available at the time of writing in Great Britain.

First Published – April 1987

British Library Cataloguing in Publication Data
Simister, W.
Simple applications of the Amstrad CPCs for writers.
1. Word processing 2. Amstrad CPC 464 (Computer) 3. Amstrad CPC 664 (Computer)
I. Title
652'.5'02854165 Z52.5.A4

ISBN 0 85934 165 8

Printed and Bound in Great Britain by Cox & Wyman Ltd, Reading

CONTENTS

Introduction

The writer, whether writing novels, or articles for magazines and newspapers, needs to present his copy on sheets that are approximately A4 size (11½ ins. x 8¼ ins), with double spacing between the lines, and with a reasonable margin at the sides and bottom. In addition they should be numbered on each page.

Usually the writer uses a pen or pencil, writes out and alters his copy, types it, alters it again, and retypes it for presentation, making a copy as he does so. In some cases the alterations require typing more than twice to ensure they are correct, and in each retyping there is the danger (one could almost say the certainty) of more mistakes in spelling. It is extremely tedious – and 'extremely' is not the first word I wrote there. There are times when one can hardly express it strongly enough.

Imagine the trouble caused when it is essential to insert a full paragraph into a chapter early in the book, and the hours of retyping needed to make the copy look good with consequent renumbering of succeeding pages. Even a couple of extra words on a page can demand a line or two being carried on to the next page. The difficulties are only lessened in degree during the writing of articles. Many of these run to four or more pages, and alterations to them can be even more frustrating because of the greater number of articles written.

The advent of personal computers, together with special programs offered for 'word processing' seemed to be the answer to these problems, and in some cases they were; but in many more cases the difficulty of learning to handle the program of word processing involved too much thought about the program, and not on the actual writing. They are excellent programs in most cases, and when used by a secretary typing out manuscripts for a writer who can afford such a luxury, they fill the bill.

However, word processors are not the only way a writer can obtain help in his work from a personal computer. Most of them come complete with an in-built language, usually a variant of 'BASIC', which has commands built into it that will satisfy most writers.

The original manuscript for this book was written using an Amstrad CPC 464, with disc drives, and with a DMP1 printer; together with the BASIC program that is described in its pages.

The program, of only fifteen lines, is so simple that in many cases it will translate almost directly from the Amstrad to almost any other kind of personal computer. Each of the lines will be dealt with in turn, so that even someone who has no knowledge of BASIC programming should be able to use it on any Amstrad 464, 664, and 6128. To use it on any other make of computer will require the careful alteration of a few lines. The 'WINDOW' facility on the Amstrad is not available on some machines, but that will be dealt with when we come to that chapter.

It is suggested that the book is read through most carefully to the end, and then gone over chapter by chapter as you proceed. One writer I tried it on (one who had never used a computer before, but always wrote directly onto a typewriter) was able to master it in less than two days, and immediately went out to buy the same set-up. He tells me it has saved him many hours of work each week, and that correcting errors, and producing extra copies, is now simple.

Since the above was written a DMP 2000 printer has replaced the DMP-1. This adds two chapters to the book, and is easy to work with. It is advisable, though, for those writers who have a DMP 2000, that all the chapters dealing with the DMP-1 are read first, for in them are most of the details of the many program features.

Chapter One

THE HARDWARE

The number of words used by any writer in his article or novel can range from 500 to 80,000 or more, and to place that quantity of words safely in the computer, so that it can be regained at any time, is the purpose of this book. However, a computer considers all characters separately (whether letters, figures, commas, spaces, or anything else); and when a writer counts his words, one does not normally think of the spaces, dots, and other interjections as characters. Therefore we must get that matter straight first.

The sentence: 'The cow jumped over the moon, and the little dog laughed.' contains eleven words, but to a computer it contains fifty-eight characters (including the two inverted commas at the beginning and end). So we have the equation: as 11 is to 58, so 500 (or 80,000) is to X. X being the number of characters your writing is composed of. That is the quantity that has to be considered when working out what a computer and its storage system (discs or tape) can deal with.

This is not a hint to start calculating. It has been worked out (approximately) that a page of 27 lines, each line having an average of 11 words (287 words per page), is roughly equal to 1,500 characters; or in computer language: 1.5k. So the contents of a chapter of about 6000 words takes up 33k (that is 33,000 characters).

To this number of characters used up by the writing of the 6000 words has already been added the various characters used in writing the program that puts the chapter into the computer; so it can be stated (roughly) that each page of writing requires 1.5k.

The capacity of the Amstrad CPC464 is given as 64k. Yet, when FRE is used to ask for capacity remaining in the empty computer, the result is 42,249, so where has the extra 21,751 gone? It has been used up by placing in the memory a program called 'BASIC'; a program that allows you to command the computer in various ways.

But that is not all. You cannot use all that 42,249 for your entries; about 10k is needed by the computer to print on the

screen the words you want to enter. So you are left with approximately 30k to work with. That means that if your chapter is over 5500 words long you will have to divide it into two sections for storing.

Whichever sort of computer you are using must therefore be of a big enough capacity to hold a chapter. Anything of 64k or over will be excellent. All the Amstrad personal computer range are suitable, and so are many other computers. However, they must have provision for use with a disc drive: that is a means of storing the program; other than a tape recorder, which has the capacity for storing long programs, but takes too long to do so, and is somewhat uncertain in its results at times.

The disc drives supplied by Amstrad for use with either of their personal computer have a capacity of 169k on each side of the disc; but not quite all of this is available for storage. The explanation of this is a little complex, but worth studying, for on it depends how much you can store.

Let us suppose that you have three chapters to store. When the first one goes in, and is corrected even once, a use of the CAT command will produce a list, together with the number of ks it has used, but it will be seen that the correction has produced two versions of that program: one will be (in the instance of this program) WRITER1 .BAS and the other one is WRITER1 .BAK. The BAS version is the latest one, and the BAK version is the one before you corrected it. The BAK version is left there in case you have made a mistake, so that it is always available. It can be recalled by clearing the computer, and entering SAVE followed by the name of your program plus .BAK, enclosed between double inverted commas. This can be a real save at times, but is not usually needed very often.

It does, however, use up more of the storage space, for if you had three programs on that disc, each of 30k, they would amount to 90k between them, and with the BAK version of one as well while you worked on it, there would be 120k in use – plus another 30k for the use of the disc's memory while it was transferring a program from the BAS to the BAK memory, making five times 30k in effect: 150k. Now the disc drive has only 169k in all. Therefore, if there were 34k in each of the three programs, the total would be 170k, and there would be a message to tell you the disc is full when you try to save the full program.

That is not easy to understand, so we will express it in a different way.

As soon as a program is entered, and seen to be correct and complete, then erase the BAK version, thus releasing (say) 30k. To erase it: see the instructions in Chapter 2.9 of the Amstrad DDI-1 User Instructions.

This is done for each program in turn as it is completed, so that as the third program is being entered and SAVEd, the CAT command will produce: WRITER1 .BAS 30k WRITER2 .BAS 30k WRITER3 .BAK 30k WRITER3 .BAS 30k. This is 120k in all, and to it must be added (mentally) another 30k with which the disc operates, making an effective use of 150k.

From this it can be seen that there are just 19k to spare, so that if each chapter were 32k long, the remainder becomes just 9k, which is as small a margin as you should allow.

After all, there are two sides to each disc, and each side can hold (safely) three chapters of 5,000 words each, so a complete book of (say) 24 chapters (120,000 words) can be held on four discs. At the time of writing such discs can be obtained for as little as £23 for ten of them, and there is every sign that prices will be lower in future.

To sum up what has gone before: a computer of not less than 64k capacity, and a disc drive that will hold 169k on each side is needed to enter and hold almost any size of book, and will, of course, deal with smaller pieces so much easier.

The Amstrad CPC464, with an attendant disc drive, and a printer attached, will allow any writer to enter and save all his writing. So will the CPC664, and the CPC6128. If you already have a different computer, and so long as it has the capacities mentioned, there should be no difficulty in using the program that is described in this book.

Before leaving the disc drive it should be mentioned that a double disc drive is a great advantage in the matter of safely saving what has been written. The way I use it is as follows:

With a clean, formatted disc in each of the two machines: A is the top one, and B is the bottom, I make sure the computer is in the A mode. Just typing and entering |A will do that. The '|' is the shifted sign '@' to the right of the letter P on the keyboard.

I then type and enter the name of that chapter, enclosed by double inverted commas, and preceeded by SAVE. The disc drive

whirs and passes back the ready sign, after which I use |B to put it in the B mode, and repeat the SAVE. Then I go back to |A again. In this way what I have written has been saved on two separate discs.

This is not essential at all, and it is perfectly correct to use only one disc drive, and merely change the discs when you want a copy. However, in the interest of simplicity of working when I am thinking out some text, I use the two.

There is another matter related to the quantity of text entered: it is the number of characters that can be entered in one numbered line. This is another instance of thinking in terms of characters rather than words.

When writing a program in 'BASIC', a line number is typed first, followed by the instructions. Have a look at the program in this book (page 8) and you will see what is meant.

In entering a line of instructions (or anything else that is to be put into that numbered line), only 256 characters can be entered; and these have to include the number and whatever else is used in the way of letters, spaces, commas, or other interjection.

For instance, in the case of the text for a book of any kind, there must first be: 1000 PRINT #S, and this, together with the double inverted commas at the beginning and end of the text, amounts to 16 characters, so there are then only 240 left. This means that every three or four lines (depending on how you are using line length in the finished print-out) you have to start a new line.

This check to the stream of thought when producing the copy has stopped some writers from using a computer, but it is a minor irritation compared with the tedium of continually typing out fresh copies, with the necessity of always having to watch for typing errors. When the text on a computer is checked properly, it never has to be done again. In any case, with the aid of the KEY re-allocation facility on the Amstrad (and some other computers) there need be only one key pressed after the number is entered. This will be explained when we come to detailing the program item by item.

Printers of other makes than Amstrad can be used with the Amstrad, for it accepts the standard 'C E N T R O N I C S' style interface printer. The one I use at present is the Amstrad DMP-1 printer, and any instructions given for the printer apply specifically

to that machine. I am given to understand, however, that the intructions are easily adaptable to other printers.

Printers are normally passive partners to a computer, accepting all orders passed to them by the computer, so far as they are able. The DMP-1 is set into single spacing between lines, or double spacing, by the flick of a dip switch. This is situated at the back of the machine, on the right hand side when facing the back of the machine. There are four small switches in a recess; the one on the right (nearest the end of the machine) is No.4. If this is set down (off) the machine prints single spacing between lines. If it is set up (on) the printer uses double spacing between the lines. This switch is so small, and set so deeply in its recess that it is difficult to operate unless one is facing the back of the machine. Leaning over from the front is not recommended – mainly because it is then so very difficult to see and operate the tiny switch.

The DMP-1 cannot print both the pound sign and # at the same printing. It prints one or the other, according to the setting of No.3 dip switch (second from the right when facing the back of the machine). When dip switch 3 is down (off), it prints #. If the switch is set up (on), it prints the pound sign. Therefore, in all printing one must decide before setting the machine which of these two signs is the more important in the text. In this program, for instance, the # sign is more important than the pound sign, and so switch 3 is set down.

Most other normal commands are given in the computer program. The relevant ones will be described when we are explaining the program, but one that you may wish to use is given here. It is possible on the DMP-1 to print words or lines at an extended length. That is: each letter is extended to double its former length. This is done by using two codes: CHR$(14) and CHR$(15) in the form:

'PRINT #8,CHR$(14)+"SIMPLE APPLICATIONS OF THE AMSTRAD CPCs FOR THE WRITER",CHR$(15):'

It may have been noticed that in the above command two single inverted commas were used in place of a double inverted comma. This is necessary because when using the present kind of program the double comma is reserved for use at the beginning and end of the text only. If used in any other place it will cause trouble. Therefore, all conversations in the text must have only single

inverted commas to enclose them. For example: John came into, the room, and said, 'Collect the mail, please.' and the servant did so.

This chapter has dealt only with the 'Hardware'; that is with the computer, the printer and the disc drive. In future chapters the program will be explained thoroughly, and therefore some of the previous remarks may be repeated.

```
10 REM ************************************
20 REM **** SIMPLE APPLICATIONS OF THE ****
30 REM *** AMSTRAD CPCs  FOR THE WRITER ***
40 REM ************* WRITER1  *************
50 REM ************************************
60 KEY 138,"CLS:LIST 1000-"+CHR$(13):KEY 128,"RUN"+CHR$(13):
KEY 129,"PRINT #S,"
70 INK 0,23:INK 1,0:BORDER 23
80 MODE 2:WINDOW 1,65,1,25
90 S=0:A=3:REM to 8 (incl)
100 WIDTH 65:GOTO 1000
260 PRINT #S:PRINT #S:PRINT #S:PRINT #S,"_":PRINT #S:PRINT #
S,SPC(30);A:PRINT #S:A=A+1:RETURN
270 PRINT #S:PRINT #S:PRINT #S,"_":PRINT #S:PRINT #S,SPC(30)
;A:PRINT #S:A=A+1:RETURN
280 PRINT #S:PRINT #S,"_":PRINT #S:PRINT #S,SPC(30);A:PRINT
#S:A=A+1:RETURN
900 REM from 1010 only:- 19 lines(260):20 lines(270):21 line
s(280):  The rest:  26, 27, or 28.
1000 PRINT #S,"_":PRINT #S:PRINT #S,SPC(29);A:A=A+1:PRINT #S
:PRINT #S:PRINT #S,SPC(5);"SIMPLE APPLICATIONS OF THE AMSTRA
D CPCs FOR THE WRITER":PRINT #S:PRINT #S:PRINT #S,"Chapter O
ne              The Hardware":PRINT #S:PRINT #S
```

Chapter Two

SOME DETAILS OF THE PROGRAM

Because a large number of writers have never used a computer before, the explanation of these few lines of program is going to be extremely detailed, so I ask more experienced programmers to be tolerant. They may find some uses of 'BASIC' they have not met previously, but much of what is explained will be simple. It is based on the type of programming needed for the Amstrad, and should require very little adaptation to be able to work well on many other computers.

It is usual to number each line in steps of 10 normally, because that leaves room for extra lines to be inserted later, when a use of the program indicates such a need. There are instances, however, when a greater jump than 10 seems to be an advantage. The explanation of that will be given when we arrive at lines 260 of the program on the preceeding page.

A space is left after the line number, and, starting at 10, the word REM is found. This is short for REMark, and is a word that makes the computer ignore everything in the line that comes after it. It is most useful for adding in remarks to help yourself understand what you intended, or what you should do, if you come to the program after an interval. The computer reads each number in turn, and what comes after it. In the case of REM it immediately goes on to the next line number.

Lines 10 to 50 have been used to produce a decorative title for the program. In line 20 I have used the title of this book. You should put the title of your own book there.

Line 40 is a reminder of the name used for storing the program on disc. WRITER1 is the word used for the first chapter. Succeeding chapters will have succeeding numbers at the end of WRITER, so that as the need to SAVE a section is reached, a glance at that title will help. It may seem to be a waste of labour and space, but there is no doubt that when a number of programs have been written, and one comes back to them after a period, a glance at the program itself will not always reveal what it is about. So, use REM lines in the heading to help yourself.

```
60 KEY 138,"CLS:LIST 1000-"+CHR$(13):
   KEY 128,"RUN"+CHR$(13):
   KEY 129,"PRINT #S,"
```

Line 60 is the first important one. The word 'KEY' is what is called a KEYWORD. It performs a function when used: in other words the computer performs certain actions when that word is used. In this case the computer will accept your instruction to change the use of that key: it can be programmed to put some other character (or characters) onto the screen.

All the keys used in re-allocating are shown in a drawing on page 15 of appendix 111 of the Amstrad manual: the square section of 12 keys numbered 0 to 9, with . and ENTER as well. The ones I use are Nos. 138 (the full stop), 128 (the nought), and 129 (the figure 1) of that block. They are handy to get at, and there are other keys with those characters elsewhere.

The formula is fixed: line number/space/KEY/space/138/comma/double inverted comma/CLS/colon/LIST/space/1000-/double inverted comma/+/CHR$(13)/colon.

Compare that with the program at line 60, and you will see that the oblique lines are merely a device to separate off the various elements of the command during the explanation. They are not used in the program. After that colon comes another KEY re-allocation, for number 128, and after that another, for 129. When you have got used to them they become easy to find.

Starting with the first one: key 138 (the full stop) is now used to enter into the computer the order to LIST from 1000 onward. The dash after the 1000 means 'onward' in that context. The CLS is a direct order to the computer to clear the screen. The +CHR$(13) is equivalent to ENTER when used in this fashion, and the command is momentarily flashed onto the screen prior to the command being carried out. It will be realised from this that the key is now printing 14 characters, instead of the single full stop it printed before re-allocation.

Key 128 (the 0) is slightly different in that the CLS is not now needed. This is because the command RUN carries within itself the order to clear the screen. Otherwise, it is a similar exchange of use: it prints RUN, and is followed by ENTER.

Key 129 is different again, for it prints the characters we put into

the key, but does not clear the screen, nor does it use ENTER. This is because we have not this time used either CLS or +CHR$(13). It just enters the 9 characters: PRINT #S, , and leaves them on the screen. This is a very convenient way of saving typing time.

At the beginning of each piece of text, during the typing of the book or article, it is necessary to type in the line number, a space, then PRINT #S, (9 characters), and then a double inverted comma – beyond which comes the text. Now, all that is needed is the line number (we can deal with that later), a space, the characters on the new key (129) with one dab of a finger, and the shifted 2 (to produce the double inverted comma).

The use of these pre-programmed keys does not come into action until after RUN has been used, for the computer has to read through the program before it can act on the command. If the command had been entered directly, that is without a line number, the computer would accept it directly, but when entered after a line number it must be read in sequence with the program to be acted on correctly.

Some of you will have become a little restive by now at the repeated use of '#S', because you will know that the command to print on the printer is actually #8. This will be made clear when we reach line 90. But before that we will deal with the numbering.

There is a facility of the Amstrad called AUTO. This will automatically produce a new line number, with an increase of 10, each time the ENTER key is pressed. It is cancelled by using the ESC (break) key, and has to be re-introduced after that. It is started off by entering your next line number preceeded by AUTO and a space.

For instance, to start this paragraph I had previously reached line 1350, so I typed in AUTO 1360 (the next one I needed) pressed ENTER, and the new line number (1360) was produced, complete with a following space. Then I had only to dab the figure 1 (previously pre-programmed), use shift 2 (for the double inverted comma), and could at once continue with the text. At the end of that line (about 200 characters) I pressed ENTER, and a new number was produced, ready for the single finger dab and inverted comma that made a new line ready for new text.

This may not be quite clear, but will be explained in more detail

when we reach the stage of using the program in later chapters for entering your own text.

```
70 INK 0,23:INK 1,0:BORDER 23
```

Line 70 of the program deals with the colours to be used in the computer while writing. There are many arguments about which colours are the most restful on the eyes while continually staring at the screen. I have now settled on a very pale blue background (nearly white), with black lettering. For a time I used a pale yellow background (25) with dark green letters (9), but found it not so restful as the light blue and black, so that is the one I use. Do your own experiments with these colours, and try each one for a time.

It is simply done, for the line at 70 is easily changed. Notice the way it is done: INK 0,23:. INK is a key word, and tells the computer, when picking the colour for 0 (its background), to use No. 23, which is pale blue. The second command: INK 1,0:, tells the computer to choose No. 0 (black) for the lettering. In MODE 2, which is used throughout this program, there can be only two colours in use.

In experimenting for your own choice, then remember that it is the second number after INK that is the colour. The list of colours and their numbers is given on page F3.2 of the manual for the Amstrad. The second number after the first INK is the background colour, and the second number after the second INK is the lettering colour.

The BORDER can be a different colour to the background and may be any colour you choose. However, because the lettering is always tight up against the border on the left hand side, it does not help in the matter of clarity. I prefer to make the BORDER the same colour as the background. Try some changes to that command, and you will see what I mean. It operates in a different way to the INK command: The only number after the BORDER is the colour number.

```
80 MODE 2:WINDOW 1,65,1,25
```

There are two commands in line 80: MODE and WINDOW. Dealing with MODE first, it should be understood that there are

three MODEs on the Amstrad: 0, 1, and 2. In MODE 0 there are twenty characters to the width of screen; in 1 there are forty; and in 2 there are eighty. For the purposes of writing text for books or articles, we need to use about 65 characters to the page, so this is the most useful for our purpose. It is most difficult to make the beginnings of each line level when using MODE 1, as will be explained at greater length when we come to the section dealing with typing your own text into the computer. For the present please accept that MODE 2 best suits our purpose.

The second command in line 80 is WINDOW. This is a most convenient facility of the Amstrad (and of some other computers), for it allows us to put on the screen, before we transfer it to the computer, the text written to the width of characters that we require. The width of the screen (in MODE 2) is eighty characters. We want to use (for instance) 65 characters. With this facility we can make the screen size anything we like. The width of (normally) 80, can be reduced to 65, and the depth, which is 25 (lines) can also be altered. In fact quite a lot of WINDOWS can be used on the screen at the same time – but that is of no present concern. If you want to learn how to use this facility for other purposes, then see page 53 of chapter 8 in the Amstrad manual.

In fact we use WINDOW 1,65,1,25 for our purposes. After the word WINDOW come four figures, separated by commas. The first represents the left hand margin of the window, the second is the right hand margin, the third is the top margin, and the fourth is the bottom margin. Since the normal screen would be 1,80,1,25 it can be seen that 15 character spaces at the end of each line are not being used. If you put a different coloured margin around the screen for a test, you will see that this is so. You can do that with the BORDER command in line 70.

This simple command ensures that when we have entered a piece of text, and wish to see what it looks like on the screen when we use RUN, it will be the size we want it to be. There will be a much fuller explanation of it in use in a later chapter.

Now, for some purpose of your own, you may wish to use a different number of characters for your text. I first of all started out by using 60, but after a time I considered that the lines were a little too short, so adopted 65 as my standard. This is almost sure

to vary according to the type of printer you use. I use the DMP-1, which prints 10 characters to the inch. That gives me 6½ inches of text width, and on an 8¼ ins. width of paper that gives sufficient margins at the sides.

On the DMP-2000, which I hope to get shortly, I am told that the use of 12 characters to the inch is possible. I could then increase the width on the computer with WINDOW 1,72,1,25, which will give me the same printer width in inches as at present, provided I also alter the value given in line 100 (of which more later).

```
90 S=0:A=3:REM to 8 (incl)
```

In this line comes the explanation of the #S that has been used so frequently. When working on entering the text, whatever it may be, it is often necessary to throw it onto the screen to see what it looks like. If each line was prefaced with #8, which is the command to direct the text to the printer, it would be an elaborate procedure, and waste much paper. Instead, with this use of #S, the text is directed to the screen; and is displayed on the screen exactly as it would be on the paper. So we use a variable to change the 0 to an 8.

A single letter variable, in the Amstrad and most other computers, is a letter used to represent a number. If the letter has the string sign ($) after it, it then represents a string of characters, but that is not our present concern. We use a single letter to represent a number, and do this at the beginning of the program so that right at the start the letter S is made to equal the number 0. Thereafter, whenever the computer encounters S it knows that S means 0, and proceeds to PRINT to the screen. When we are ready to use the printer it is only necessary to enter LIST, and then alter the 0 to an 8 in line 90: S=8 instead of S=0. At once the text is diverted to the printer at every place where it is commanded to by PRINT #S,.

This is easy to do, but can be forgotten after printing out some sheets of text. Always, after finishing the printing, (for instance when an alteration is needed) remember to alter that variable back to the 0, so that it is ready for screen printing to check the alteration.

The second command in line 90 is A=3. This is another much used variable. The program list printed on page 8 is of the first chapter, and the first page number of that chapter is 3. It may be surmised from this that A is a variable used to represent the page number, and the experienced programmer seeing the commands to make A=A+1 in lines 260, 270, and 280 will also surmise their function.

However, this book sets out to help the beginner, and must therefore explain in more detail. This will be done when we come to the explanations of lines 260, 270, 280, and 1000, in chapter three. For the present we will say only that A is made to equal the first page of the each chapter as you come to it, and that the REM remark that comes after it should be made to contain the last page number in the chapter. Thus, in the heading, there is a record of the page numbers in that chapter. It will be seen that in chapter 1 there were pages 3 to 8. Therefore A in the next chapter (2) will have A=9, and the number after REM will be left until the chapter is written, and then entered before it is permanently saved.

```
100 WIDTH 65:GOTO 1000
```

In line 100 are two commands, and the first one, WIDTH, is a direct command to the printer to print each line 65 characters long. This is a most convenient command, for with it the printer can be made to print within the limits of the number given. It therefore makes it easy to type in lines of the correct length that will print at the same length.

As will be seen it is used in conjunction with WINDOW in this program. WINDOW gives a width of 65, and so does WIDTH. So, if all the lines are typed to conform to that width, within the window of 65, they will all be printed to justify on the right. In this case 'to justify' means that all the letters at the extreme right hand side of the text will end directly under each other, and there will be as straight an edge to that margin as to the beginnings of each line.

All this justification is carried out by the writer in this program, except that he is helped by the program form, and it must be done manually. The way it is done will be explained when we deal with entering one's own text.

Goto 1000 means exactly what it says: the computer is directed to jump any intervening lines between 100 and 1000, and go directly to line 1000. The explanation of these intervening lines will be given in the next chapter.

Chapter Three

MORE PROGRAM DETAILS

The rest of the BASIC program, lines 260 to 1000, is there to place the pages of text in a correct position relative to the sheets of paper between the perforations that mark the separate sheets of a continuous length, usually folded.

On the single sheet of paper, 11ins. long, there is room for sixty-six lines of printing by the DMP–1. Because double spacing is in use (the fourth tag has been raised) we can talk about each page containing 33 lines of printing. To provide for the margins at the top and bottom, some of these 33 lines must be blank; and these blank lines must be provided for by a similar number of 'PRINT #S:' entries.

Also, it is a help to provide a small dash to print onto the perforation, so that we can check that each page is filled with 33 lines. The first command in line 1000 provides a dash. Then there is a blank line, followed by the page number. This can be seen in the first three commands:

```
PRINT #S,"–":PRINT #S:PRINT #S,SPC(28);A:
```

The first command, for a dash, is typed by pressing the '1' key (already programmed to provide for this in line 60), and a double inverted comma (shifted 2), which is always necessary to enclose anything to be printed. In the case of the first command it is for a dash, which is made by the shifted zero key (on the main keyboard). A colon (:) always separates each command, and in effect causes the computer to start a new line. For a real beginner it should be explained that a shifted key is one that is pressed at the same time as the SHIFT key (at either end of the bottom line). This particular dash, although it looks high on the key, is printed at the base of the letter space. So in effect it does not contribute a line of space to the page below, when it lands exactly on the perforated line. Conversely, the dash at the end of the page (which we shall come to later) adds its space (double space in this program) to the end of the page.

The second command should be clear by this time: it merely adds a double line of space. The third command has two features that may confuse. SPC(28);A: tells the computer to start the line, print nothing (spaces) for 28 characters, and then print A. (which has no inverted commas). Now we know that A equals 3 in line 90 of the program shown, so the computer prints what A stands for. To repeat that: the computer does not print the letter A, but what that A stands for; because there were no inverted commas.

Next comes A=A+1. This is 'computereese' for adding 1 to the number that A stands for. Put in another way it means that A(3) has now had 1 added to it, and A now equals 4, so that the next time the computer is told to print A it will print 4. Remember, a computer, (and by order of a computer – the printer) prints any character enclosed by double inverted commas as it appears in a program; but without those commas it prints what that character stands for. If, for instance, you typed PRINT 4 * 4, it would print 16. The '*' stands for multiply on a computer.

'SPC(28)' needs little explanation. SPC is an abbreviation for space; in effect the command here is to print 28 spaces. In the brackets is put the number of spaces required. Because it was not enclosed by inverted commas the computer printed what it meant, and not the actual letters and numbers. Of course that number (28) can be altered to put the 'A' in any place along the row, so that it can be at the right hand edge if wanted there.

```
PRINT #S:PRINT #S:PRINT # S,SPC(5):"SIMPLE
APPLICATIONS OF THE AMSTRAD CPCs FOR
THE WRITER"
```

The 5th. 6th. and 7th. commands in line 1000 should be quite clear now because of the previous commands. There are two blank lines, and then a command to leave 5 spaces, and print the name of the program. This is printed on the third line below the number, and is positioned by the number of spaces used.

It is a controversial point whether or not the book title should be used at the head of each chapter. I prefer to do it because my chapters are stapled, and the book title at the head of each chapter prevents it getting lost, or placed in the wrong file. Line 1000 is used only once, at the chapter beginning, so it is written in this form. You must decide for yourself whether or not you wish to do this. If you do not, then the 5th. 6th. and 7th commands should be deleted from line 1000.

This, as you may realise, would mean there are three lines less in this first page, and therefore line 800 must be looked at. This is a line of REMinders only, put there to remind you that in order to position each page of text on its sheet, there must be 33 lines on each page between the dash marks '_'. The lines of text must be counted carefully, to ensure that this is achieved. When the correct number of lines has been put in, a command is typed in telling the computer to GOSUB to line 260, 270, or 280. This is done because the text is entered in batches of three or less lines. More details of why this is necessary will be given in the chapter dealing with the entry of your own text.

The 33 lines for each page are made up from blank lines; the No. line; the text; and the space at top and bottom that is left to give a margin. In line 1000 (for the first page only) there are ten lines incorporated, excluding the line making the dash. That allows for either 19, 20, or 21 lines of your text before the command to GOSUB 260 (or 270, or 280) is entered.

If however, you decide to delete the title from this program during your writing (its spaces and words occupy 3 lines), then you must add those three deleted lines from line 1000 to the number of lines of text counted. That is: line 800 must now have the 19 altered to 22, the 20 to 23, and the 21 to 24. This is only applicable, of course, to the text of the first page.

To put this another way: the part of line 800 dealing with the counting of lines has to be altered by the same amount as an alteration of the lines in 1000.

In our analysis of line 1000 we had reached the 7th command, and will deal with the 7th to 12th commands together, for they are similar to those that have gone before: two blank lines, and a command to print the chapter number after 24 spaces, followed by 2 more blank lines.

This brings us to the next line, which is line 1010. This, however, is the beginning of your own text so will be left until later. For the moment we have to deal with the lines numbered 260, 270, and 280.

It will be as well to study the beginnings of each of these three lines, for they have only one difference, and that is the one caused by the number of lines of text.

```
260 PRINT #S:PRINT #S: PRINT #S:PRINT #S, "_"
270 PRINT #S:PRINT #S:PRINT #S," "
280 PRINT #S:PRINT #S,"_"
```

From the way these have been printed in this book: directly in line, it will be seen that line 260 orders 3 spaces to be printed before it prints the dash (which must be printed on the perforated line). Line 270 orders 2 spaces, and 280 orders 1 space - in each case. This is plus the space caused by printing the dash. This makes room for either 26, 27, or 28 lines of your text before you have to enter the GOSUB line.

The program lines have been deliberately numbered with the number of text lines counted (with an 0 at the end). These are entered into the GOSUB lines. This makes them easy to remember, and is a practice that can make life a lot easier at times.

If you wish to have a larger space than I have allowed at the bottom of the page, then make a new line 250 with one extra PRINT #S: in it. That is: like 260, but with an added PRINT #S:. This makes 4 in front of the dash 'PRINT #S'. Then remove line 280. From then on you can use only 25, or 26, or 27 lines of your own text. This kind of alteration is easy with this program, provided you remember to think in terms of a definite count of these new numbers of your own lines of text you have decided to use.

There has been mention of the fact that the dash must be set on the perforated line between separate pages. This is not an automatic act. It has to be arranged, and is done in this way:

On the DMP-1 printer there is a transparent plastic cover over the paper as it emerges. The paper moving knob on the right causes the paper to emerge. Count the side holes as they emerge. When the 10th hole following after a perforation is exactly half covered by the trailing edge of the plastic cover, then it is positioned correctly for the dash to print on the perforation. But only if the printer has already been used once or more since being switched on. I make sure of this by using PRINT #8," " on its own, directly into the printer, before I start getting the 10th hole into place to put the perforation line in position to be printed on.

This is far simpler than it sounds when first read. The act of preparing to print a chapter is almost automatic after one or two

mistakes. Enter LIST, and press ESC when the top half of the program appears: see that lines 60 to 100 are centred on the screen. Then, by editing, see that in line 60 the KEY 138 command is set to LIST 1000; that in line 90 the S is made to equal 8, then in line 100 set the GOTO to 1000. After this enter the PRINT #S line, and then set the number of holes from the trailing edge of the plastic cover as directed above, and and you are ready to press the 0 key that has been set to the command: RUN. This description has taken it for granted that you are able to 'EDIT' the screen. If not then study the Amstrad manual, for the process is well described there.

The alteration of those lines (60 & 100) to 1000 can have been occasioned by earlier alterations as you worked. They need not be set to 1000 all the time. As you work, and the program gets longer it is useful to alter those numbers to ones that are much nearer the line you are presently working on. For instance, when working on line 1500 and onward, I set them both to 1480, so that there is not a long wait for the program to scroll on the screen after an alteration or addition. If they were always set at 1000 they would take a long time to scroll up on the screen. I always work with those two keys set a few lines back from the current line, and increase their number by 10 lines or so as I work. This is the real reason they have been pre-programmed.

For the beginner at programming the RETURN on the end of lines 260, 270, and 280, should be explained. The command GOSUB always has to have coupled with it a RETURN on the end of that line, or section of program, to which GOSUB has been directed. The effect of RETURN is to send the computer back to the line immediately after the line containing the GOSUB. It never makes a mistake.

The position of the line to which the GOSUB is directed does not matter. The three lines 260, 270, and 280, have been placed at that position because it keeps them within the program we are writing at the moment. They could just as easily have been put at lines after 1000. Because they are in the compact 'heading' program, they must be passed over by the computer when reading its way through the program; hence the GOTO 1000 in line 100. The computer passes over those lines until commanded by a GOSUB line to scan them. Then it acts on the commands, and RETURNs to the line after the one it came from; having read, and acted on, only the line it was directed to.

In these two chapters, two and three, the main program has been explained thoroughly, and should prove a valuable source of reference when first learning to use it. There is one point, however, that might be referred to, and that is the lay-out, or positioning, of the computer, the disc drives, and the printers in relation to each other. It is usually a matter of personal preference, but the resulting tangle of the attachment leads can sometimes be horrific, so my own way of doing this might be of interest to some.

My own arrangement is with the printer on the left, computer and monitor centrally placed, and the disc drives to the right of the monitor: 8ins. away, as specified in the manual.

I have placed the monitor on a 4 ins. high plinth, so that it is more on eye-level, about 3 ins. back from the keyboard. It has the lead for the printer (coming out of the left back of the keyboard) going left to the printer, which is also placed on a plinth, making it 3½ ins. higher. Behind the printer is a box to contain the folded continuous paper. The cover of this box is about 2 ins. lower than the issuing paper, so it folds nicely into position. There is no front to this box, so the paper unfolds out of it in a most satisfactory manner.

The lead for the two disc drives comes from the right back of the keyboard, and so is kept well separate. The whole set-up occupies a table space of 4 feet by 2 feet (backward).

In addition, because I frequently type from a sheet of paper on the left of the keyboard, I have made a very light table top from hardboard, which sits on top of the printer when that is not in use, and is supported by three legs. This fits closely against the top of the printer, and on it I stand a small stand, which holds the paper I am reading from. In this way I can easily correct anything I wish, for both copy and monitor are on about the same level.

A couple of shelves, the lower one being 11 ins. above the table, straddles the rear of the paper holding box, to hold all my spare discs. This lay-out makes it easy to work, and is never disturbed, except for the occasional spring-clean.

Chapter Four

GETTING USED TO THE PROGRAM

If you have read, and thoroughly understand the two previous chapter (2 & 3), it is time to try your hand at entering some text. I have chosen a chapter from a book of mine that is sufficiently varied in text to demonstrate the method fairly clearly. It is the beginning of Chapter Eight of 'James of Little Heaton'.

The first move is to copy the complete main program, lines 10 to 1000 inclusive, from page 8 of this book. When this is entered accurately, with all the punctuation marks and spaces correctly in position, alter the title and chapter No. to 'JAMES OF LITTLE HEATON', and Chapter Eight, and also change A to equal 129 (A=129 in line 90). Then RUN it, and you should get the page number, the title, and Chapter Eight under each other, well spaced out, but with no other wording, except the READY sign.

An easy way to enter those 'PRINT #S:' items in lines 260, 270, 280, and 1000, is to enter only up to line 100, and then at this point RUN the program. The screen colour will change, and there will be a ready sign, under the program you have entered so far, but now it will all be in MODE 2: narrow black letters on a pale blue screen. From now on those KEYS you have pre-programmed in line 60 will be ready for use, but you can check them properly later. For the present continue to the end of line 1000.

Then is the time to check you have entered correctly the KEYS line (60). With the heading on the screen, and the ready sign waiting, press the full-stop sign that is pre-programmed, and immediately the program should list the 1000 line only; nothing more – except for the ready sign. If that is correct, then press the 0 sign, and that should return you to the display of the page number, and title, etc. Note that you cannot press those keys with any result until after an ordinary RUN has been typed and entered. You are then ready to start, so can now press the full-stop again to produce the line 1000.

Make a start by typing 'AUTO 1010', and then entering it. A number will be produced in the proper place, ready for the next

command. Ignore that for the moment, and press ENTER again. It will produce 1020, and if you continue to press ENTER the numbers will follow in tens. The way to stop it, and revert to manual numbering, is to press ESC.

I do not often use AUTO now, for I prefer to correct lines as I proceed, and that entails the use of ESC in most cases, so AUTO is of little use to me. However, when using a written copy to type from, I like to get on with the typing, and go back over the previous lines to correct them after a paragraph or two have been typed in. In such cases AUTO is invaluable.

Press the full-stop key again, and the listing of 1000 will appear again. After producing a number (complete with a space after it) press KEY 1, and 'PRINT #S,' will appear (I used it to print that). Now use the SHIFT key and 2 to produce the double inverted commas, and you are ready for text. It it is the start of a paragraph, so you must type in some spaces after the double inverted commas.

This is another matter of personal preference: Some prefer a small indentation for starting a paragraph; others like a big one. I normally use four spaces in this position. When using double spacing on the printer they usually show up well enough.

One matter has to be stressed here before you proceed: if a double inverted comma is used in the text (between the one at the beginning and the one at the end) it will cause a mistake to stop the computer when you RUN it. The double inverted commas in this type of program can be used only in those two places: the beginning (after the PRINT etc.) and the end of a numbered line. This is a simple statement of fact. You have to use a single inverted comma for all places in the text where you need double ones: for instance when typing in a conversation.

Another restriction is on the number of characters that can be put into the computer with one line number (including the line number). It is 256. When the little black cursor reaches that number it makes a small squawk, and stops there. Each time you press a letter (at that point) it protests, giving a warning that it will not take any more. The reason is that the buffer, (the mechanism inside that stores the characters before putting them in the correct place), can only hold that number.

Therefore, since we are using 65 characters to the line of text, and 3 times 65 is 195 (leaving only 61 out of the 256) we must

restrict the number of lines of text we use to three. This together with the fifteen taken up by the number and its command PRINT #S," makes 210 characters in the line when we stop at the end of the 3 lines (of 65 each). To attempt to enter more would cause a broken line, for we would have only 46 characters left. This is about two thirds of a line, and we could not work out a suitable way of matching that with a new line.

I have digressed from the main instruction, which is about how to enter part of a chapter. This is inevitable, and will occur frequently.

The next move, after entering 1010, a space , and the PRINT #S,", is to enter some text:

```
                V
1010 PRINT #S,'   Phoebe gave the slab of butter an extra hard s
mack, shaped it quickly, and then stamped it with the beechwood m
ould to put the Polefield mark on it.  That was the last one toda
y.  Since'
                                  ^
```

Please note that at the beginning and end of that three line statement I have put a single inverted comma. You will have to use double apostrophes. This is because, as I have explained earlier, the computer will not accept double apostrophes inside a statement, and to enter this in my program for this book, I had to put the double apostrophes before the 1010, and after the last inverted double commas.

When this is RUN it will be seen that 'quickly' and 'Polefield' start a letter in from the leading edge; and 'Since' is not at the end of the line; they need justifying.

Justification consists of making the end letter of each line sit at the beginning and end of a line exactly: so that the entire piece of typing (or printing) has straight edges at each left and right hand margin. It is simple to do, and extremely hard to explain in words.

LIST the program again, and study the position of the words forming those three lines of text. The three lines actually start directly after the double inverted comma (for we used four spaces, and those are counted in the three lines). Now, under that double apostrophe, in a column, must come a space in each line, terminated by the double apostrophe at the end, (see arrows).

To explain that differently: a column must start with a double

apostrophe, and below that must be a space, another space, and a double apostrophe. And each of those two spaces must have a character close to each side of it. This is achieved by adding an extra space here and there in the text; as shown in the following example.

```
                V
1020 PRINT #S,'    Phoebe gave the slab of butter an extra hard
smack, shaped it quickly, and then stamped it with the beechwoo
d mould to put the Polefield mark on it.  That was the last one
today.  Since'
                Λ
```

A space has been placed before 'Phoebe', before 'shaped', after 'quickly,', before 'That', and before 'Since'. This second example has been numbered 1020, so that you can enter and RUN both of them together for comparison.

A careful count of the characters will show that the second example has now got 64 characters in each line. The 65th one is being used as a spacer while we enter the text. I will give one more example, and this time will put a full stop in each place where a space has been added.

```
                V
1030 PRINT #S,'    This is an example .to show .that spaces can b
e added easily here and there in a sentence, .without spoiling .i
ts .appearance when it is printed.  .This sentence will finish in
 the middle of'
                Λ
```

When you have studied this, replace all those extra full stops with a space, and then study it again after it has been RUN. It may seem obvious to you that spaces have been added to various places, but justification by a word processing program can be just as obvious when searched for. Whoever receives your manuscript will notice the straight edges, and will be pleased at the presentation, having been used to receiving less tidy and presentable copy.

When you are quite sure you understand the principle behind this means of justification, then remove lines 1020, and 1030, so that you can proceed with typing in the examples given in the correct way. We will now move to line 1020, which, you may recall, will have to start without an indent for a paragraph beginning.

```
1020 PRINT #S,'she had come to work for James Somerton  six years
 ago  she must have done that hundreds of time.  There was more b
utter produced here than at any farm near by; and in the winter,
too;  for this'

1030 PRINT #S,'was still the only farm that consistently kept the
ir stock alive right through the cold season.'
```

These three lines (1010 to 1030) form a single paragraph, and paragraphs do not need to finish at the same place each time. In some cases they may have only one word in a line, and in others may complete a line. The only rule to remember is that in all cases the column under the double apostrophe must contain a space until the final double apostrophe (if this is not a para. end), and that the only way to achieve this is to use spaces discreetly. Sentence ends, and where commas occur, are good places to choose, and the next best is where long words come close together. Word processors cannot think things out in this way, and are inclined to bung them in just anywhere.

```
1040 PRINT #S,'    There was a gentle tug at her skirts, and she
looked down at little Anne, who was beaming up at her hopefully.
 She smiled in response,  and lifted up  the three-year-old.  He
r mother would'

1050 PRINT #S,'chide her for spoiling the child,  but a scolding
 from Mistress Elizabeth was never very serious.  She had a way
of letting you know which offences were really serious,  and whi
ch  were token'

1060 PRINT #S,'offences, like the one she was about to commit.'

1070 PRINT #S,'    She sat the girl on her arm, while Anne hugged
 her neck, and buried her face in Phoebe's yellow hair. Then, ove
r at the board table where the cream was separating in four large
 pans of milk,'

1080 PRINT #S,'she sat the girl on the table, reached for a large
 wooden spoon, scooped up a generous  portion of cream,  and gave
```

the spoon to the child. 'Don't spill any, else I'll have trouble from your'

1090 PRINT #S,'mam.'

1100 PRINT #S,' There was a rapturous silence as the child licked off the cream, and then she carefully handed the spoon back to Phoebe, wiped her lips with the back of a small hand, and then held up'

1110 PRINT #S,'her arms to be lifted down. On the floor she turned, looked up at Phoebe with a smile, said 'Ta.' and hurried off out.'

1120 PRINT #S,' As she disappeared Hannah entered through another door. With a glance at Phoebe's position near the cream pans, and a glimpse of Anne's back, she knew what had happened.'

1130 PRINT #S,' 'You'll have Mistress Elizabeth after you. She's told you before not to do that.''

1140 PRINT #S,' Phoebe smiled. 'I know. But when that child smiles it's hard to resist her. I only let her have a little bit, and she never clamours for more, as some children do.' She grinned at'

1150 PRINT #S,'Hannah. 'She's getting wise, is that one. She always picks a time when her mother can't get here before she's away again.'

1160 PRINT #S,' Hannah smiled sympathetically, but as her gaze swept around the small room the smile left her face. 'You'll have to hurry, Phoebe. Mark Rawson's ready to leave, and that butter has to go'

1170 PRINT #S,'with him. Come on. I'll help you pack it.''

When this is all typed into your own computer, and you look at the listing, it will be easy to see (if you have entered the spaces

shown here), that right down the page, under each of the apostrophes, is a column of blank spaces. Examine that closely, and make sure there is a character close to each side of each space in that column. In other words, look on that space as if it were the margins of the ends and of the beginnings of all the lines; for in effect that is what it truly is.

To draw together all that has been described in this chapter is not easy, but put briefly it is this: type the line number, a space, press KEY '1', type a double apostrophe, type in four spaces (if it's a paragraph start), and type in your text. You can ignore spacing at first, but remember that only three lines of text can go into any numbered line. Then add spaces where you think they will produce a space under that first double apostrophe. If you want both your margins to be ruler straight, then don't put another space next to that column of spaces.

If you don't wish to justify the right hand margin of your work, that is to leave it with your right hand margin as ragged as when you are typing on a normal typewriter, then you will still have to put some spaces into your text at times, in order to prevent words being cut in two. In this case all you have to do is to see that no word crosses that space under the double apostrophe. It will not be of importance if a letter from the word on the left of that space obtrudes into the space, but it must not extend beyond it.

I have found that it is very little (if any) more trouble to justify the right hand margin while you are making sure no words are cut in two.

While you are working, adding lines as you go, the use of the pre-programmed keys is most helpful. After entering a line, a single press of '0' (for RUN), will produce what you have typed. After checking for mistakes, a single press of the 'full-stop' will produce the listing once more. However, as you were warned earlier, if you leave lines 60 and 100 to refer to 1000 all the time, you will eventually have a long wait for the lines you are working on at that moment to appear. That is why it is best (after entering an additional 10 lines or so) to change the LIST number, and the GOTO number in lines 60 and 100 respectively, to the number of a line or two before the next one you wish to work on.

The next task confronting the writer using this program is to separate off the pages, so that they will each print on the same

place on the pages formed by the perforation marks of a continuous stack of folded paper.

Some writers may prefer to count their text lines as they go on with their writing, adding in the GOSUB commands as they go, but it is normally advisable to wait until the entire chapter is finished, so that all the corrections and additions to the text are completed – otherwise, after a correction that adds lines, an erasion of all subsequent GOSUB lines is called for, with the additional task of counting the lines again.

As I have said previously, the DMP-1 printer has 66 lines of print between one perforation mark and the next. Using double line spacing gives 33 lines. Ignoring the chapter headed first page, each page has at its head: a space, a number, and another space. Then comes the text, of 26, 27, or 28 lines. Then, and this is the point, if there are 26 lines the program puts three lines of spaces before producing the dash that prints onto the perforated line as a marker. If there are 27 lines of text, then two lines of spaces are given, and for 28 lines of text just one line of spaces is given; all three quantities of spaces are plus the line of spaces where the dash is printed of course. So this amounts to 33 lines in each instance.

If you are using a printer other than the DMP-1, it is easy to find out how many lines are allowed by that printer. With the printer set to double spacing, type in a line as follows:

```
50000 FOR X=1 to 40:print #8,X:NEXT
```

Using a line number well beyond any in your own program. When this line is RUN, using 'RUN 50000' the printer will print on the paper a column of numbers. See that a perforation line is about level with the printing head on the printer, and you will then have a permanent record of just how many lines that printer uses between perforation marks.

After that digression we will go back to the DMP-1.

With the first page, because extra room is taken up by the title and the chapter number (one space before the page number, two before the title, two between title and chapter number, and two before the text) there are seven less lines free for the text. Therefore, the transfer to normal pages from first page is effected

simply by using 19, 20, or 21 for the count of text lines. If there are (on the first page only) 19 lines, then the GOSUB for 26 lines of text is used, for 20 use the one for 27, and for 21 the one for 28. There is a REM line at 800, that keeps this reminder in a convenient place.

The manner in which these spacing lines are used is simple. At the point reached when either 26, 27, or 28 lines matches the end of a program line, an extra line is inserted between that program line and the next one. Use a number finishing with 5 in each case: if the 26 (etc.) line is at the end of 1170, type in 1175 GOSUB 260. It is easy to remember, for one just adds a 0 to the number of text lines in the GOSUB command. Of course, the RETURN at the end of lines 260, 270, and 280 will send the computer back to the line number after the GOSUB line number.

This is a suitable time to remind you that the use of jumps of 10 in line numbers is normally preferred. For this there is a very sound reason. I will give an example. When I had moved well past line 1900 in the program for this chapter, I found a good reason to add in an extra paragraph at that point. I was able to use five extra line numbers (1902 to 1906), and they slipped in easily. Then I used the RENUM facility of the Amstrad to make the lines revert to their jumps of 10, by entering RENUM 1900,1900,10, so that every line from 1900 onward was renumbered in jumps of 10; including my new paragraph.

There are nine extra program lines to be used between the jumps of ten, and at three text lines to a program line, that gives about a page of text. If your addition is liable to be more than that, then renumber the lines immediately after the gap before you start. Assuming the gap comes before line 1900, then first use: 'RENUM 2900,1900,10,' and there will be a hundred lines waiting to be used. For beginners I should explain that the RENUM facility starts with the new number wanted, the old number you wished to discard, and the amount of the jump wanted.

One further point needs to be covered: the way the lines are counted. It would seem to be too obvious to need explanation, but I have found errors creep in.

Examine the lines you have typed in: lines 1010 to 1170. The 21st. text line finishes at the end of line 1080, so at 1085 you should

put 'GOSUB 280'. This leaves just one line of a last sentence (the word 'mam.') to be counted with the next page. It is frequently possible for this to happen, but there is no way to avoid it – apart from re-wording the sentence, and I prefer never to do this. Once a word has been selected it must be changed only in order to improve the sense of the sentence.

The remaining lines you have entered amount only to 20, too few to make a normal page, so more lines would have to be put in to fill that page. However, remember that on the second page of a chapter (and on all succeeding pages) the number of lines to count are 26, 27, or 28.

Everyone will count the number of lines in their own way, so my explanation of the way I do it is given only as a guide. I count the first double apostrophe (after the PRINT etc. command) and the spaces (if any) that come directly below it: making sure not to count the double apostrophe at the program-line end. The existence of no double apostrophe at the end will only indicate that the line finished elsewhere.

To print out the chapter I make sure that lines 60 and 100 are set at 1000 for LIST and GOTO respectively, and that A=8 and not 0. Then, setting the printer so that the perforated line is level with the printing head (as explained previously, with the aid of the tenth hole being level with the plastic cover end), I start it off with RUN, carefully watching all the time to see that the dash sits on the perforated line on every change of the page No. If one is wrong then a press of ESC will stop both the printer and the computer. A second dab at ESC will take it out of the computer's control, and the alteration can be made. It will only be a miscount of lines that can allow it to go wrong. If it is correct make sure it is put on disc at once.

The method I use to prepare for a new chapter may be of use to some. This chapter has used 'writer4' as its title in the discs, and that is first altered to 'writer5'. I then make sure that lines 60 and 100 have had their LIST and GOTO set to 1000. Next line 90 is altered: if S=8 I change that to S=0. A is next altered to A=34 (1 after the no. of pages in the last chapter). Then I go to line 1000, and alter the Chapter number to Five. This completes the alteration of the heading program, and it is necessary then to remove the main text. This is done easily by entering: 'DELETE 1010-10000'. This clears everything after line 1000.

If you are in any doubt about how to work with this program, then read chapter 2, 3, and 4 again, using some examples on a computer to check your progress. An entirely different program to enter text is given in the next chapter. It may suit writers who must write in a hurry when the mood attacks them.

```
10 REM ****************************************
20 REM ********* SIMPLE TEXT WRITING *********
30 REM ****************************************
40 REM ************  RAPAWORD  ***************
50 REM ****************************************
60 KEY 138,"CLS:LIST 1000-"+CHR$(13):KEY 128,"RUN"+CHR$(13):
KEY 129,"DATA "
70 INK 0,23:INK 1,0:BORDER 23:MODE 2:WINDOW 1,65,1,25
80 S=0
90 WIDTH 65
500 REM ********** START OF ROUTINE **********
510 FOR n=1 TO 5
520 ws=-1
530 WHILE ws
540 READ z$(n)
550 z$(n)=z$(n)+CHR$(32)
560 WHILE LEN(z$(n))>1
570 cv=INSTR(z$(n),CHR$(32))
580 q$=LEFT$(z$(n),cv-1):z$(n)=MID$(z$(n),cv+1)
590 IF q$="@" THEN PRINT #S,CHR$(8);", ";:GOTO 620
600 IF q$="X" THEN ws=0:GOTO 620
610 PRINT #S,q$+CHR$(32);
620 WEND:WEND
630 PRINT #S
640 NEXT
650 REM ********** END OF ROUTINE **********
```

Chapter Five

A PROGRAM FOR FAST WRITING

The routine 'RAPAWORD', on the previous page, is designed to take advantage of the easy way in which a small BASIC program can use the PRINT command to keep words in one piece when moving to a fresh line. At no time does it cut any word in two. Its great advantage is that one can type directly into the computer without having to worry about words being broken at a line ending. In many cases this is a great advantage to the writer who finds it difficult to break his train of thought in order to deal with extraneous matters, like the spacing of words in a sentence, for words can continue to flow into the computer until it gives a squawk; and then it is only necessary to delete back to the last complete word, use ENTER, and start a new line with just the word 'DATA' and a space.

It does not justify the right hand line ends, however, for it leaves them just as ragged as does an ordinary typewriter. It has another snag, also. It will not accept a comma in any way. This BASIC program treats all commas entered directly into the text as if they were spaces. There is a line in the program that will put a comma onto the screen, though (at 590), where provision is made, that, if a lower case ' m ' with a space each side of it is entered in place of the comma, it will print a ',' in its proper place on the screen. The DMP-1 printer, however, does not recognise the code 'CHR$(8)', which (to the computer) means 'move the cursor to the left', and so prints a comma, but with a space between the word and the comma.

However, both the computer and the printer will recognise a semi-colon (;), and if you can remember to put one of these in place of every comma, it will do the job well.

Because of the use to which I put this program, this does not matter to me, for I use it in place of a piece of paper and a pencil when first writing articles. I will give an example later. For the moment it will be best if I give a complete breakdown of the program.

Ignoring lines 10 to 50, which have been explained earlier, the

first line of consequence is 60, which pre-programs the same two keys as in the previous program. We do not, however, need the 'PRINT #S,' sequence in this program, so that is omitted. We do, however, need the word, 'DATA ' with a space after it, to put on each line after the number, so that is put in line 60 also. With the aid of this programmed key (after the use of AUTO for the line number), we can now start each line with just a single key strike of '1' (right near to 'ENTER'), and the full stop followed by some spaces if it is a new paragraph. The space after DATA is already programmed into the key, so that an interrupted sentence can be continued at once.

The next line (70) specifies the background colour, and the ink to be used; as well as doing away with the border by making it the same colour as the background. It also introduces MODE 2, with narrow characters. Incidentally, MODE 2 has no effect on the printer. Then comes the WINDOW command. This, as in the previous program, stipulates that all the characters in each row amount to 65; mainly because that is the WIDTH to be used for the printer, and one can then see what it is going to look like when shown on the screen. The WINDOW command has been explained in the description of the previous program.

In line 80 we give a value to the variable 'S': it is made to be equal to '0'. When you want to see on paper what you have typed, then change the '0' in this line to '8'. Don't forget to change it back to '0' when the printer has finished.

Line 90 contains a direct command for the printer only. It ensures that everything is printed to that width. Both this command, and the WINDOW command in line 70, must be given the same number if, at some time, you wish to change the width of your work.

We now come to the actual routine that prevents the computer from cutting a word in two. It starts with line 500, which is merely a REM line. Line 510 is a 'FOR' line which is tied to the number of paragraphs you have entered with this program. It was '5' with the sample program I ran when testing it. There will be a fuller explanation of its function when we come to use this program later in this chapter.

In line 520 starts the WHILE-WEND routines that can be used to keep the program returning to the beginning again, 'ws' is made

to equal –1. While ws is equal to –1 it continues to move. As soon as 'ws' is made equal to '0', (see line 600), it comes to a stop. In line 600 it says: 'IF q$='X' THEN ws=0'. Then comes GOTO 620, which passes over the PRINT command in line 610, and moves on to 630 (print a space) and to 640, which is the 'NEXT' that sends the computer back to line 510. Put in a rather crude way, this means that when the computer comes across an X in its reading through the lines it brings into play the FOR-NEXT loop, and starts a new paragraph.

During the explanation of these lines there will be many of these digressions, for the lines in such a routine are usually interconnected. We will now return to the earlier lines.

In line 530 is the first of the two WHILE-WEND loops. In 540 is the READ command. This tells the computer to look for DATA somewhere in the program. It can be placed anywhere, and such DATA is to be given the variable z$ (a string variable that can handle words). It is also qualified by being followed by (n). This 'n' is the variable we met in line 510. On the first passage through the program it is equal to '1', so the computer will continue to READ DATA until it meets a 'X'. Then line 600 becomes operative – but we have dealt with that, so we shall go back to line 540. On the second pass through, 'n' will be equal to '2', so will take the second paragraph, and proceed on.

We meet some rather more involved sequences in line 550 and onward. Here we have: 'z$(n)=z$(n)+CHR$(32)'. This means that from then on z$(n) means the word plus a space after it, for the code 'CHR$(32)' really means a space. It could have been put in a different way, but the computer responds well to the codes.

Line 560 starts a second WHILE-WEND loop: this time it will continue until the word and space is greater than '1'. Line 570 contains the command word 'INSTR', which searches the string of words, and makes 'cv' equal to the one chosen.

Now comes the line that makes sure that only complete words are printed in the text on the screen, for the string handling commands (LEFT$ and MID$) are used to give the variable 'q$' the position of such words, and in line 610 they are printed – but not before (in lines 590 and 600) the variable 'q$' is compared with a character 'm', and a character 'X', in which cases some different commands are given. In 590, if q$='m', then it prints a comma, as

explained earlier; in 600, if q$='X', it stops printing that line to order the computer, eventually, to NEXT, in order to start the sequence all over again.

It is well worth while studying line 580 thoroughly, for it is the key to how this program does its job. LEFT$ chooses the character at a position (from z$(n)) that is to the left of 'cv' (which was defined in line 570) including 'cv'. The '-1' means that it must be one character to the left of what that 'LEFT$' has chosen. Then comes a colon (:), which separates off a new command, in which z$(n) is now given a new value by using MID$. In this case 'z$(n)' is made to represent the words after the characters defined by 'cv'. This means that on the next time through, 'z$(n)' represents *only* the characters after the one chosen by INSTR in line 570.

This is not easy to understand at first, but for beginners I should explain that the use of INSTR, and of LEFT$ and MID$ in a situation like this, can produce most of the mysterious things that computers do – in this matter of juggling with words and letters. In this case 'q$' is made to represent a word with a space each side of it, and LEFT$ selects that word for printing, while MID$ makes sure it is not used a second time. The whole matter is made clear when we remember that the computer is treating the string in 'q$' as a single character for the purpose of printing, and therefore, if there is not room for it on that present line, it goes to the next line to print it.

Beginners at programming, and some that have been doing it for a little time, should make every effort to understand how it is done, and in particular how the commands are phrased, for the efforts to do this will be repaid many times over.

The FOR-NEXT loop starting at line 510 controls the move from one paragraph to another, so deserves a little more explanation. It works by separating all the text (until a 'X' is met), into one complete batch. This is done by using the variable 'n' as a way of qualifying the string variable z$, which otherwise would consider all the text as one great whole. That is why (n) is in use following every 'z$'. A FOR-NEXT loop always has a number such as 1 TO 5 (or sometimes A TO Z, where this is of more use) to make sure that the procedure inside the loop is repeated more than once. When I first wrote the program it had five paragraphs in the text I used, so the number there is 1 TO 5. Without that

'X' to stop it, each of the paragraphs would follow directly behind each other, as if the whole were one large paragraph. With its help each new paragraph starts on a separate line.

On its first passage through the program the string is equal to 'z$(1)'. Then it meets a 'X', and is forced to go on to the NEXT, where it is sent back to line 510. This time the string is equal to 'z$(2)', until it meets another 'X'. And so it can go on until the number in 'FOR n=1 TO 5' is met.

Another point should be cleared up before we start using this program: the use of a comma. As explained previously the comma cannot be accepted by the computer during the running of a DATA line. I use a semi-colon (;), and if the printed copy is to go out to an Editor, I always go over it with liquid paper to remove the top dot of the semi-colon. This serves admirably.

In some cases, however, a proper comma may be wanted, and I then use the letter 'm' spaced out between the words in this way: 'bread m and cheese'. This prints out onto the screen perfectly well, because of line 590, but, as mentioned earlier, it will not be printed properly by the printer; it turns out always as: 'bread , and cheese', because the DMP-1 has no means of knowing the meaning of 'CHR$(8)'.

Therefore I use it only when the typing I am doing needs to be seen on the screen only. This occurs when I wish to doctor, or alter any sentences that would take so much pencil and rubber work that it would leave a scribbled jumble. When this program is used to put text on the screen, and the excellent EDITing facilities of the Amstrad are used, it is easy to deal with.

One thing more should be mentioned again: the need to keep one's eye on the number of paragraphs used. Unless that number in the FOR line (510), is maintained at the same level as those of the (X)s, the computer will only print the number of para's that conform to that number. For instance, if there was a 10 in line 510, and five paragraphs, then five paragraphs would be completed, and a 'DATA exhausted' message would be printed. If it were the other way about, with 5 in line 510, and ten paragraphs, the computer would print five paragraphs, and just forget the others.

When I am in a hurry I put a large number in that FOR line (510), (sometimes FOR n=1 TO 50), and then get on with the

typing. I don't need to worry about a screen message when I am EDITing.

An example of the way it can be used is shown at this chapter end. This was only a short piece, accompanied by some drawings, and was sent as it emerged from the printer, but with the addition of a small heading.

Copy this into your computer, and save it as a pattern from which you can build up programs of your own. It can be adapted to a variety of uses. Line 40 gives the name used to save it.

It will be seen that semi-colons have been used in place of commas; it took only a few moments to erase the top dots. The heading was put in front of the routine used to print the text (at line 200), and consisted of two lines of space, the heading, and two more lines of space. The routine carried on after that, and it was all printed on a single sheet. The series of nine articles, each with three drawings, was used by a gardening magazine, and the same method has been in use for a year or two now.

Examining the first three lines (1000-1020), which cover two paragraphs, note the full stop to ensure the use by the computer of the four spaces that start it. In line 1010 the same device is used, but this time it is a longer paragraph, which stretches on sufficiently to need a second numbered line (1020). Take note of the way line 1010 is finished – at a word end, with nothing to terminate it. Then notice how 1020 is started – with a space before the first word (put there by DATA). This ensures that in printing it, on the screen or the printer, the beginning of line 1020 is tight against the end of line 1010, but with a space as well. It has followed closely behind what has gone before.

This means that when typing into this type of program, the writer does not have to think of anything more than his writing. At least, no more than he has to if he is writing with a pen and paper. With the numbering set to AUTO, and the use of the pre-programmed key (1), he can carry on writing easily. He just uses ENTER at the end of the line, and that produces the number with a space after it; then he dabs the '1' to produce DATA and a space, and carries on with the next word. Notice that 'ENTER' and the '1' are near one another on the keyboard.

Some mention has been made of 'discs' and their capacity earlier. At this stage in writing this book, near the end of a fifth

chapter, I find that it is all on one side of a disc, and that, with the 'BAK' portion of this chapter still on disc (17k) there is still 45k free for future use. That, plus the 17k of the 'BAK', gives 62k for another chapter. Sufficient to deal with a chapter of anything short of 19k. However, I shall turn to the second side of the disc for succeeding chapters, because it is always safer to have plenty of disc space in hand.

I give my CAT reading here.

```
BUTFLY1.BAS  3k     WRITER2.BAS 19k  WRITER5.BAS 17k
RAPAWORD.BAS 3k WRITER3.BAS 17k  WRITERX.BAS  2k
WRITER0.BAS  5k     WRITER4.BAS 26k
WRITER1.BAS 15k     WRITER5.BAK 17k
45k free
```

You will see that the CAT readings are read down the first column, then down the second column, etc. All these programs are a part of this book, each with its own heading. This makes such small items as the 'Contents' page into 2k, but it is well worth while, for then they can be recalled, altered, and printed without any trouble. WRITERX is the Contents page, and WRITER0 is the Introduction.

While we are on CAT readings, there is a simple way to print them on paper. ENTER CAT, and when it comes up on the screen, use the EDITing facility to put 'PRINT #8,' in front of it, with the double apostrophe before, and after, the part you want to be printed. If you used 'KEY 1' to put the PRINT #S, on the screen, don't forget to alter the 'S' to '8'.

If you have so many items on the disc that they will not all go into a PRINT statement (too many for it to hold), then use a second line, having finished the previous line at a good place. Four lines of CAT will not quite go on, so it is best to finish at the end of the third line, and use another line for the rest. I had to do this with my present entry of CAT.

While on the matter of 'EDITing', those of you new to the Amstrad may not have realised that there are three ways to EDIT. For convenience it pays to select one method, and I have found the 'COPY CURSOR METHOD' to be the easiest and simplest method of all. It is fully described on page F2.8 of the manual.

Any use of the second method at times is liable to be confusing, for it will be by using the 'CLR' key, instead of the 'DEL' key. It is better to stick to one method only.

```
10 REM ****************************************
20 REM ********* BUTTERFLY ARTICLES **********
30 REM ****************************************
40 REM ************  BUTFLY1  ***************
50 REM ****************************************
60 KEY 138,"CLS:LIST"+CHR$(13):KEY 128,"RUN"+CHR$(13):KEY 12
9,"DATA "
70 INK 0,23:INK 1,0:BORDER 23:MODE 2:WINDOW 1,65,1,25
80 S=0
90 WIDTH 65
200 PRINT #S,"_":PRINT #S:PRINT #S:PRINT #S,SPC(20);"THE RED
 ADMIRAL":PRINT #S:PRINT #S
500 REM ********** START OF ROUTINE **********
510 FOR n=1 TO 5
520 ws=-1
530 WHILE ws
540 READ z$(n)
550 z$(n)=z$(n)+CHR$(32)
560 WHILE LEN(z$(n))>1
570 cv=INSTR(z$(n),CHR$(32))
580 q$=LEFT$(z$(n),cv-1):z$(n)=MID$(z$(n),cv+1)
590 IF q$="@" THEN PRINT #S,CHR$(8);", ";:GOTO 620
600 IF q$="X" THEN ws=0:GOTO 620
610 PRINT #S,q$+CHR$(32);
620 WEND:WEND
630 PRINT #S
640 NEXT
650 REM ********** END OF ROUTINE **********
```

```
1000 DATA .  There is little sexual difference in the Red
Admiral; but the female may be slightly larger; and have mor
e pronounced red patches. X
1010 DATA .  The upper surface of both is velvety-black; w
ith scarlet bands on all four wings. There are white spots
and bands towards the tips of the forewings. The underside
of the forewing is bright with red; black; white and fawn pa
tches; but the
1020 DATA hind wing is mottled brown; like a leaf or piece o
f bark. X
1030 DATA .  It can be seen from March-October; mainly bec
ause this butterfly migrates from the Continent; though a fe
w hibernate here. X
1040 DATA .  The Red Admiral goes everywhere--fields; gard
ens; town and country--and likes over-ripe fruit. Eggs are
laid on the upper leaves of nettles; one egg to each plant;
and these hatch in about seven days. X
1050 DATA .  The caterpillar varies in some individuals.
Normally it is velvety-black with black spines; spotted with
 tiny white spots; and with a broken yellow line along its s
ide. Others may have buff or yellow spines; and some are ch
eckered with
1060 DATA lemon-yellow; so that the background appears to be
 greenish grey.  This caterpillar takes 23 days to mature. X
```

Chapter Six

ADDING PAGING TO THE FAST WRITER

There may be occasions when the writer will wish to use the fast writing method for more than a page or two, and therefore I have devised a method of doing that. It is not very elegant, but it does the job – within certain limits. At this chapter end is a program that does the job. It will be seen to be the same program as previously used, but with some additions. Those extra lines will be dealt with here.

The first alteration is to line 80, where A is given a value of '1'. This is similar to the same provision made in the list of a previous program, for when its value is changed by adding 1 after each page is printed, it will continue to number the pages correctly. Therefore, line 80 is now:

```
80 S=0:A=1
```

In line 200 there are two alterations: the title has been changed to 'SOME GARDEN BUTTERFLIES' instead of being 'THE RED ADMIRAL'; and an addition has been made to the end, for there is now an additional command: GOTO 500, which is the beginning of the word wrapping routine. That line is therefore now:

```
200 PRINT #S,'_':PRINT #S:PRINT #S,SPC(25);A:
A=A+1:PRINT #S:PRINT # S,SPC(16);'SOME
GARDEN BUTTERFLIES':PRINT #S:GOTO 500
```

Don't forget that where I have put a single apostrophe, each side of the bottom dash '_', there should be double apostrophes, and similarly around the title. This, as explained earlier, is caused by the use of the first program in this book for entering the lines.

After line 200, and before line 500, come six lines that are used only by GOSUB commands, which is why the command 'GOTO 500' was put in line 200. These six lines are put this early in the program because the writing of text (from line 1000 onward), is then much easier.

Lines 250 to 290 are there to provide spaces at the page end and beginning; in a similar way to those in the first program. This time, though, they are entered in a slightly different way. The ASC CODE for moving to a lower paragraph is CHR$(10). This is used in a form that will make it repeat as many times as we want, by using the form: STRING$(7,CHR$(10)). The first number inside the brackets is the number of times the cursor will move down the page. This saves the repetition of PRINT #S: so many times. It is hardly worth doing for as small a number as 1, but in the interests of uniformity it is done in this present case. Therefore, the five lines 250 to 290 are given in this way. The line 300 is merely a REM line, to remind you of the letter to be used for sending the computer back to a line to print spaces and a dash, a number and a space, before the text starts again. Here are the six lines:

```
250 PRINT #S,STRING$(4,CHR$(10)):PRINT #S,
'_':PRINT #S:PRINT # S,SPC(28);
A:A=A+1: RETURN

260 PRINT #S,STRING$(3,CHR$(10)):PRINT #S,
'_':PRINT #S:PRINT # S,SPC(28);
A:A=A+1: RETURN

270 PRINT #S,STRING$(2,CHR$(10)):PRINT #S,
'_':PRINT #S:PRINT # S,SPC(28);
A:A=A+1:RETURN

280 PRINT #S,STRING$(1,CHR$(10)):PRINT
#S,'_':PRINT #S:PRINT # S,SPC(28);
A:A=A+1: RETURN

290 PRINT #S,'_':PRINT #S:PRINT #S,SPC(28):
A:A=A+1: RETURN

300 REM 25 lines=U; 26 lines=V; 27 lines=W;
    28 lines=Y; 29 lines=Z. Remember to count in th
heading and space on the first page.
```

Allowing for 3 lines at the top: the number, and the space above

and below it, there are 30 lines left out of the 33 each page will carry. It is not correct to carry on the text until it reaches the bottom, and the minimum of spaces below the text should be 2, but preferably 3 or 4. With lines 250 to 290 I have allowed for anything from 25 to 29 lines of text before the line which must be the last one, but you should manage with 26, 27, or 28. We will deal with the different ways of counting the lines later.

Looking at line 260, we see that 'PRINT #S,' is still used at the beginning. This is because it must work that way. The 3 is there to control the number of lines down it must go by order of CHR$(10), and then there is another 'PRINT #S,' to command that the Dash '-' is printed, another 'PRINT #S:' to lower it a space more, and then a 'PRINT #S,' and after that 28 spaces along the line to place the 'A' (which will be a number) in the best place to be more or less centred. Incidentally, at this point you can position that page number where you want, by altering the number of spaces in that SPC(25) position. I keep it central because there is more control over the paging in that place; if an extra page is added it makes no difference.

To return to line 260. After the number of the page is in a position that suits you, the next part is 'A=A+1' to ensure that the next page number is increased – as explained earlier. Then comes the 'RETURN', which sends the computer back to the command after the one that sent it to 260. In this case to line 610, at the point where a command tells it to GOTO 660, which sends the computer back into the word wrapping routine.

All the lines 250 to 290 work in exactly the same way, but a different number of spaces are put at the end of different text lengths. Line 300 is a REM line to remind you of the letter to put at the numbered line end – of which more later.

The next lines to be described (they are additions) are the ones at 600 to 640, in the routine. They are given here.

```
600 IF q$='X' THEN ws=0:GOTO 660
605 IF q$='U' THEN ws=0:GOSUB 250:GOTO 660
610 IF q$='V' THEN ws=0:GOSUB 260:GOTO 660
620 IF q$='W' THEN ws=0:GOSUB 270:GOTO 660
630 IF q$='Y' THEN ws=0:GOSUB 280:GOTO 660
640 IF q$='Z' THEN ws=0:GOSUB 290:GOTO 660
```

These work in the same way that was explained in the previous chapters, except that these work in response to the letters 'U to Z'. 'X' produces just a paragraph end, making sure that a new paragraph starts. The rest incorporate a GOSUB line that sends them to one or other of the lines we have recently described (250 to 290). Their working will become clearer when we describe the actual entering of text into the program.

For the present it is enough to say that when the computer compares the groups of characters it has collected, with any of those in an 'IF' line and finds a match, it carries out whatever command follows that match. In this case it ends that paragraph, and comes to GOSUB 250 (or 260, etc.) and goes to that line to carry out those orders. At the end it comes back to find 'GOTO 660', and at that line starts the routine again by going to the 'FOR' line at 510.

Line 590 has been left in this routine for those who wish to use a comma in their text. As I have explained earlier, the use of DATA imposes some restrictions, and one of these is the use of a comma; the computer ignores it when printing on the screen, so that it is not printed. If it is essential to use a comma on the screen, then an 'm', printed with a space each side of it, results in the printing of a comma in its proper place, because of the way in which line 590 acts. Incidentally, the use of the colon causes a similar trouble, but rather more serious, for it causes the rest of the line following the colon to be left out, jumping to the next line number without a break of any sort. It is for these, and similar reasons, that I normally work with the first program in this book; except where I want to work at much greater length on a paragraph or more.

The text chosen for this demonstration of the working of the pagination in this present program is an extension of the text I used a little earlier: about butterflies. There are one or two small rules that must be observed: first of all see that there are no numbered lines with more than two or three lines of words in them. That is: see that there are no more than 65 characters each in two lines – with perhaps a few more characters to end at a completed sentence. This will be more easily understood when you study the way I have entered my text into the lines. If you use longer numbered lines there can be trouble at times. This is not a difficult condition, for it merely means more numbered lines, and

you are not likely to run out of numbers – even when steps of ten are used – you will run out of computer memory long before you run out of numbers for lines.

The second condition is that you use a semi-colon instead of a comma at all times. This has been explained a little earlier in this chapter.

The third condition is that you make no attempt to enter the letters at the end of each numbered line other than an 'X' where you wish a paragraph to end. The other letters can only be put in after you have RUN the text onto the screen, and can then count the actual number of lines it makes there. The details of this will be explained more fully after it is entered.

Be sure to enter the program lines first. They are at this chapter end.

1000 DATA . THE RED ADMIRAL. There is little sexual difference in the Red Admiral; but the female may be slightly larger; and have more pronounced red patches.

1010 DATA The upper surface of both is velvety-black; with scarlet bands on all four wings. There are white spots and bands towards the tips of the forwings.

1020 DATA The underside of the forewing is bright with red; black; white and fawn patches; but the hind wing is mottled brown; like a leaf or piece of bark.

We will deal with those three lines (1000–1020) before going any farther, for they illustrate a point. Note first that each numbered line finishes at a sentence end, with a full stop and nothing else. Also, that, as they appear in the list, there are no more than three lines of text to each numbered line. In fact they could be anything from about that length to just over three lines without affecting our needs. These are: that when the lines are RUN and we start counting the number of lines that are to be used in a page, that there will be no more difference than three, or at the most four lines, when a sentence end is reached as near as possible to one of the numbers: 26, 27 or 28, because these leave a reasonable bottom margin.

There will be times when one of those numbers is reached at some point that is within a paragraph, but if you then throw the

listing on the screen, and it shows that the sentence ending is inside a numbered line, then you must go back a sentence (or go forward a sentence) to find a numbered line end that coicides with one of the permitted numbers. It can usually be found in a range of 26 to 28, but the extra lines dealing with 25 or 29, in lines 250 and 290, are there to deal with those exceptions. You must count the lines that are already RUN, and not those in the list, for the computer works on them to produce a few extra spaces at the line ends, and it is the RUN lines of text that will be going on the pages.

Don't forget: that is the main reason you restrict your text writing to two or three lines in each numbered line. We shall now continue with the text for this trial run.

1030 DATA . It can be seen from March-October; mainly because this butterfly migrates from the Continent; though a few hibernate here.

1040 DATA The Red Admiral goes everywhere – fields; gardens; town and country – and it likes over ripe fruit. Eggs are laid on the upper leaves of nettles; one egg to each plant; and these hatch in about seven days. X

1050 DATA . The caterpillar varies in some individuals. Normally it is velvet-black with black spines; spotted with tiny white spots; and with a broken yellow line along its side.

1060 DATA Others may have buff or yellow spines; and some are checkered with lemon yellow; so that the background appears to be greenish grey. This caterpillar takes 23 days to mature. X

1070 DATA . THE ORANGE TIP. The male Orange Tip has its upper wing surfaces marked with orange and a black tip on a white base. The upper wing surfaces are lightly marbled grey and white.

1080 DATA The underwings are similarly marked but the hind wings are more strongly marked. The female is similar, but is not orange tipped. Y

1090 DATA . Normally the Orange Tip is seen during May-June; but some may appear in August-September. It is fairly common in Britain; except north of the Moray in Scotland.

1100 DATA It favours flowery hedges; meadows; and woody borders. It hibernates as a crescent-shaped chrysalis; usually suspended in a hedge. X

1110 DATA . The caterpillar; which takes 25 days to mature; feeds on cruciferous plants; cuckoo flower; yellow rocket; garden honesty, etc.

1120 DATA It is green on top; lightening down its sides to blue-green and then white. Underneath it is dark green. This caterpillar is quite slender; and very even in its thickness throughout. X

1130 DATA . THE BRIMSTONE. This is a species with different sex colouring. The male has all its upper wing surfaces bright lemon with an orange spot on each wing (larger on the rear wings).

1140 DATA Underneath it is duller; tending towards buff on the hind wings. The female has two colourings; one of greenish white; and the other more yellow.

1150 DATA Her orange spots are duller than those of the male. The points on all four wings are the same in both sexes. X

1160 DATA . It appears early; usually March-October; but occasionally in February; and is common in the south of England and in Wales.

1170 DATA It hibernates on ivy or holly; clinging under a leaf; and because of its shape and colour is perfectly concealed there.

1180 DATA Eggs are laid on the buckthorn (both species); singly on the underside of leaves in May-June. X

1190 DATA . The caterpillar takes about 28 days to mature; and is glaucous green on the back; tending towards blueness at the white spiracular stripe along its side. Y

1200 DATA The underside; below the line; is yellowish-green; as are the legs. There are minute black spikes all over it. X

You may have noticed that at the end of line 1080 is a 'Y'. This is a page ending, and was found by RUNning the program, and counting the lines of text – including the title and its space below. You will find there are 28 lines, and so the letter Y is chosen. Now, count the lines of the listing, allowing two for the title as before. You will find there are 31 lines to that page end, and that you reached 28 at the end of line 1070. From this you can see that the list has to be RUN on the screen before the page can be counted: the computer has made the text 3 lines longer at that point; by moving to a new line before it has reached the full 65 characters – in order to avoid cutting a word in two.

This should emphasise the point I made earlier, about the necessity for counting the lines from the RUN print-out, and not from the list. Remember, the top of the sheet (the space, number, and space) has been provided for by the line that prints the dash (-). That gives 3 lines. Then a count of 28 leaves a margin at the bottom of two lines. I hope this is now clear.

When doing this job (after all the text is typed in) it is a help if you have written on paper the little table contained in line 300: 25=U,26=V,27=W,28=Y,29=Z. It should then be easy.

A sufficient amount of text has been included in this list to allow for the third page to be started; in order for you to see that the paging has been carried out correctly, for pages 1 to 3 are there. If you now count (on the list) the number of lines from the first use of 'Y' to the second one, you will arrive at 31 lines. Provided you remembered that this is not a first page, and there is no heading to count in. However, there are just 28 lines on the RUN print-out, and the third dash lands on the perforated line correctly.

These details have been emphasized here in order that, if you are using a different printer, and one that allows a different number of printed lines, you can adapt the program easily.

```
10 REM ****************************************
20 REM ********* BUTTERFLY ARTICLES **********
30 REM ****************************************
40 REM *************  BUTFLY2  ***************
50 REM ****************************************
60 KEY 138,"CLS:LIST 1000-"+CHR$(13):KEY 128,"RUN"+CHR$(13):
KEY 129,"DATA "
70 INK 0,23:INK 1,0:BORDER 23:MODE 2:WINDOW 1,65,1,25
80 S=0:A=1:REM PRINT #S,CHR$(27);"1";CHR$(3):PRINT #S,CHR$(2
7);"A";CHR$(24)
90 WIDTH 65
200 PRINT #S,"_":PRINT #S:PRINT #S,SPC(25);A:A=A+1:PRINT #S:
PRINT #S,SPC(16);"SOME GARDEN BUTTERFLIES":PRINT #S:GOTO 500
250 PRINT #S,STRING$(4,CHR$(10)):PRINT #S,"_":PRINT #S:PRINT
 #S,SPC(28);A:A=A+1:RETURN
260 PRINT #S,STRING$(3,CHR$(10)):PRINT #S,"_":PRINT #S:PRINT
 #S,SPC(28);A:A=A+1:RETURN
270 PRINT #S,STRING$(2,CHR$(10)):PRINT #S,"_":PRINT #S:PRINT
 #S,SPC(28);A:A=A+1:RETURN
280 PRINT #S,STRING$(1,CHR$(10)):PRINT #S,"_":PRINT #S:PRINT
 #S,SPC(28);A:A=A+1:RETURN
290 PRINT #S,"_":PRINT #S:PRINT #S,SPC(28);A:A=A+1:RETURN
300 REM 25 lines = U; 26 lines = V; 27 lines = W; 28 lines =
 Y; 29 lines = Z. Remember to count in the heading and the s
pace under it on the first page.
```

```
500 REM ********** START OF ROUTINE **********
510 FOR n=1 TO 300
520 ws=-1
530 WHILE ws
540 READ z$(n)
550 z$(n)=z$(n)+CHR$(32)
560 WHILE LEN(z$(n))>1
570 cv=INSTR(z$(n),CHR$(32))
580 q$=LEFT$(z$(n),cv-1):z$(n)=MID$(z$(n),cv+1)
590 IF q$="@" THEN PRINT #5,CHR$(8);", ";:GOTO 660
600 IF q$="X" THEN ws=0:GOTO 660
605 IF q$="U" THEN ws=0:GOSUB 250:GOTO 660
610 IF q$="V" THEN ws=0:GOSUB 260:GOTO 660
620 IF q$="W" THEN ws=0:GOSUB 270:GOTO 660
630 IF q$="Y" THEN ws=0:GOSUB 280:GOTO 660
640 IF q$="Z" THEN ws=0:GOSUB 290:GOTO 660
650 PRINT #5,q$+CHR$(32);
660 WEND:WEND
670 PRINT #5
680 NEXT
690 REM ********** END OF ROUTINE **********
```

Chapter Seven

USING THE DMP 2000 WITH THE PROGRAMS

Since writing the previous six chapters the DMP 2000 has been delivered, and installed in my set-up in place of the DMP-1. I had hoped it would be an improvement, but it far exceeds anything I had anticipated. Instead of the single type face, with the capability of extended words of the DMP-1, the DMP 2000 has many type faces, each one of them far superior, and much clearer to read.

In addition it has many commands that can be incorporated in the computer programming to produce various results, including the spacing of the lines. It is not my intention to describe these (they are well demonstrated in the manual with the DMP 2000) except where they impinge on the form of programs to which this book is devoted, and that I shall do in the following chapters; starting with the use of the DIP SWITCHES.

The dip switches on the DMP 2000 are the same in regard to the use of Nos. 1, 2, 3, and 4 as with the DMP-1, but with this newer machine there is no need to alter the position of any of them. No. 4 can be left as it is on arrival, and will produce lists and text with normal single spacing, because there is a method of programming that will alter any text to the double spacing required for our purposes. Therefore, do *not* alter the dip switches as I recommended for the DMP-1. Instead, alter the line 100 in the WRITER1 program (page 8) as follows.

```
100 WIDTH 65:PRINT #S,CHR$(27);"A";CHR$(12):
GOTO 1000
```

Don't forget that where I have put single apostrophes around the "A", you will have to use double apostrophes in the program.

That interjection, from PRINT #S to CHR$(12) needs some explanation. It is taken from page 4.10 of the DMP 2000 manual, where it has: ESC A+n, for the variable n/72 inch paper feed. This is perfectly clear, except for one thing. We are told that n stands for the number we wish to put over 72, but we are not told that it is presented in brackets after the CHR$ code.

Beginners among us, and more experienced programmers, too, are aware that codes using CHR$(?) are used for many purposes, and may not tumble to its use here; even though an example is given on that page. Let me therefore explain that: when one of those formulae are given after 'TO SELECT' in the manual, and there is a '+ n' at its end, the number we choose is to go in brackets after the 'CHR$' sign.

The next question to arise is: where does that 12 come from? A little very elementary maths is called for. The normal single line spacing on the DMP 2000 is 1/6th. of an inch. We have to find the equivalent number to put over 72 to produce the same result. 6 into 72 equals 12, so multiply 1 by 12, and put that over 72. That produces 12/72nds, which is exactly equal to 1/6th. Therefore that line, as entered at present, leaves the line spacing at 1/6th. inch.

But, if we want twice that spacing, (as we do for presenting our copy to Editors) we can now alter the number 12 to 24 (after the CHR$), and the printer will then proceed to print out the copy with double spacing between the lines. The paper feed rate between lines, is now set to 24/72, which is the equivalent of 2/6ths. or one third of an inch. Put in another way: the mean distance between lines of double spacing produces three lines to the inch, while single spacing gives six lines to the inch.

However, that is not quite all. The computer is now set to a line space for the printer of 24/72. In effect, because the 'PRINT #S,' starts that command, the printer will continue to produce double spacing until it is altered. If, for instance, a 'LIST #8' was asked for, it would be produced in double spacing.

To send the computer a signal to revert to 12/72 (1/6) is very simple. Immediately after a printed copy of the text has been produced, LIST the program, and stop it where lines 90 and 100 are available, and alter the 8 to an 0, and the 24 to a 12. Switch off the printer to cancel out its previous orders, and then switch it on again. Now RUN the program (onto the screen) for a short distance in order to ensure that the computer has read through line 100 again. In line 100 the 'CHR$' code is now 12/72, and the printer, if told to 'LIST #8', will do so with single spacing. Another method is to give a separate message to the printer as soon as it has finished its print-out of the text. This will be without a line number, as follows:

```
PRINT #8,CHR$(27);"A";CHR$(12)
```

Do not forget to put double apostrophes where I have put the single ones. This will make the printer revert to single line spacing if you tell it to: 'LIST #8'.

To ensure there is no mistake about this, a copy of that program from page 8 (WRITER1), is shown in its modified form to suit the DMP 2000 (without its dip switches altered) at the end of this chapter. This will be sufficient for anyone to use as a means of producing 'justified' copy, using the standard type face provided by the DMP 2000, provided he follows the full description of how to 'justify' given in chapter 4.

Writers who own a DMP 2000 may have jumped directly to this chapter, having skipped over the preceeding chapters. I should warn them it is essential to read chapters 2, 3, and 4, before moving to this chapter, for the explanations in those chapters refer to almost any printer; they are mainly computer program details, and may require changing for other computers.

It is my practice with the DMP 2000 to proceed with all text entering, or altering, in the single line space setting of 12/72 in line 1000; changing that only if I need a double line space print-out.

There are a number of other changes that can be made to the program when using a DMP 2000, changes which affect the printer, but have to be put into the program. For instance there are the different sorts of type face available. These, and the way to produce them are adequately described in the DMP 2000 manual, and they should be studied thoroughly. The three main ones from the writer's point of view are the 'Proportional', 'Italics' and the 'NLQ' (Near Letter Quality) options. 'Standard' is the one selected by the printer when first switched on.

These different type faces are very well decribed in the manual, but at one point I was left searching for ways of preventing the printer staying in the same type face all the time. However, what it boils down to is that you have to cancel each type face (except for Standard), by using a different code before proceeding to use a different type face. This can be puzzling at times, for the ways of coupling them up are not very clear. I append here an extra program which displays most of the type faces available, and it will print out a sample of each one, if 'GOTO 5000' is used. In it you will find the ways you can cancel one type face before going on to the next. It can sit at the end of your present program, with a stop at 4900.

```
5000 REM *** TYPE FACES ***
5010 PRINT #8,'This is Standard'
5020 PRINT #8, CHR$(27);'x';CHR$(1);'This is
     N L Q Standard';CHR$(27);'x';CHR$(0)
5030 PRINT #8,CHR$(27);'p';CHR$(1);'This is
     Proportional';CHR$(27);'p';CHR$(0)
5040 PRINT #8,CHR$(27);'W';CHR$(1);'This is
     Double Width';CHR$(27);'W';CHR$(0)
5050 PRINT #8,CHR$(27);'-';CHR$(1);'This is
     Underlined';CHR$(27);'-';CHR$(0)
5060 PRINT #8,CHR$(27);'S';CHR$(1);'This is
     Subscript';CHR$(27);'T'
5070 PRINT #8,CHR$(27);'S';CHR$(0);'This is
     Superscript';CHR$(27);'T'
5080 PRINT #8,CHR$(27);'M';'This is
     Mini';CHR$(27);'P'
5090 PRINT #8,CHR$(27);'4';'This is
     Italics';CHR$(27);'5'
5100 PRINT #8,CHR$(27);'G';CHR$(27);'4';'This is
     Italics - Double strike';CHR$(27);'H';CHR$(27);'5'
5110 PRINT #8,CHR$(27);'E';'This is
     Bold ';CHR$(27);'F'
5120 PRINT #8,CHR$(15);'This is
     Condensed';CHR$(18)
5130 PRINT #8:PRINT #8
5140 LIST 5000-5140 #8
```

Once again I must point out that in that program, as given here, you must replace all single apostrophes with double ones. This applies to the letters or numbers, and the actual text to be printed by the program; there are two apostrophes in lines 5010 and 5120, and six in lines 5020-5110, with ten in 5100. In line 5010 is the Standard type face, for it is the one always selected by the printer when switched out of another type face by a suitable command, or when first switched on.

In this program (at 5000), the various type faces have been chosen for the manner in which they are commanded, and rejected. Lines 5020 to 5050 are similar in that they all use the number 1 to turn them on, and 0 to turn them off. CHR$(27) is a

way of telling the printer that what follows immediately after it in the apostrophes is not to be printed, but is a code. That code can be a letter or number. Make sure you use either upper or lower case for this code, as shown here. The code tells the printer which of the type faces to choose. Following it is the CHR$(1) (or (0)), as mentioned above. Note the punctuation mark used: following #8 it is a comma, all the rest of them are semi colons. The dash between apostrophes in line 5050 is the one on the key holding the equal mark (=) as well – not the one on the zero (0). Incidentally, while mentioning the zero: to make the printer produce the correct 0 sign, and not the capital O, then turn the DS2-1 dip switch to ON (the first in the second block). This is done by flicking it down. Make sure to switch off the printer before doing this.

So, line 5020 means: PRINT #8, (to the printer); CHR$(27), (ESC, or not to be printed); 'x' (This is the code to use); CHR$(1) (turn it on). Then in double apostrophes comes the text to be printed (in the type face chosen by the previous code 'x'). Following that comes the same sequence of signals, except that the end number is 0 (zero) and not 1. That turns it off.

Put briefly it says, print in the type face 'x' and then revert to the Standard type face.

Lines 5030 to 5050 are the same, except that the code number or sign, is different. 'p' means Proportional, 'W' means Double width, and the dash means underline those words.

At first sight lines 5060 and 5070 appear to be similar to the preceeding lines, but they are not. Here the 'CHR$(1)' sign is not the turning on code. A combination of the letter 'S' and the 'CHR$(1)' turns on the Subscript type face, while the code for Superscript is 'S' with 'CHR$(0)'. In this case the actual kind of code appears to be triggered by the 1 or 0. In both cases they are both 'turned off' by the use of 'T' after the use of CHR$(27).

Subscript appears to be the same as Superscript but is really different in its position on the line compared with other larger texts. Subscript is level with the bottom of Standard (it runs on the same bottom line), while Superscript is level with the top of the line. That means you could use Superscript to provide the small 'th', after a number such as 4. Or you could perhaps provide a fraction, with the top number in Superscript, then a diagonal (/), followed by the lower number in Subscript. This would need many characters, but we can deal with that later.

Line 5080, for the Mini type face, is different again, for a single 'M' is used after the CHR$(27), and that is all. It is turned off again by using 'P' (after the CHR$(27)). No other effort is needed to produce a very nice and smaller version of the Standard type face.

The Italic version, in line 5090, is also good looking. It works in the same way as the Mini line (5080), but '4' is needed to produce it, and '5' to revert to Standard (after CHR$(27)) in both cases, of course.

For this Italic line I have not used the 'turning off' signal in any way, for the following line is for Italics, with a Double strike code added. This is in line 5100, which is written as if the Italics had been turned off. This is done deliberately to show how one code is added to another. One complete code follows the other with just a semi-colon between them.

By now you have learnt sufficient about these codes to see that 'G' turns this 'Double strike' code on, and that 'H' turns it off. And that '4' turns on the Italics, while '5' turns them off. Perhaps it has occurred to you that I need not have added the 'CHR$(27)' and '4' to line 5100, and you would be correct; it was already in the printer memory. However, for the beginner an example of how it is done is worth lots of explanation.

When you have entered and saved this program at 5000, then delete the section: CHR$(27);'4' from line 5100. You will find that this make no difference to the final result. The 'Italics' code is still in the memory, so there is no need to put it in again.

Line 5120 produces the Condened type face. This is a fine example of what the DMP 2000 can do, for it is clear, and really good looking. It is produced by a simple command: 'CHR$(15), and is removed from printer memory by 'CHR$(18)'. The simplest, and yet most useful of type faces.

Try a LISTing of this program, just entering (without a line number) 'RUN 5000'. The first PRINT command tells the printer which type face to use, and the second one (at line 5140) orders a list on the printer. You will be pleased with the result in the Condensed type face.

Line 5130 makes the printer leave two line spaces before it acts on line 5140, which is to LIST program 5000 from line 5000 to 5140 on the printer. This small program will print each of the type faces

named, and then produce the list of the program. It is worth keeping as a reference, for above the space is the type face to see, and below it is the way to produce it. I have left it at the end of this Chapter Seven, but if you wish to use it separately, then the word 'TYPEFACE', having just eight characters, can go onto a disc on its own.

Earlier in this chapter I mentioned the use of Subscript and Superscript for fractions. Here is a line that will do that. It is placed at line 10000.

```
10000 PRINT #8,' This measures ';CHR$(27);'S';
CHR$(0);'7';CHR$(27);'T';'/':CHR$(27);'S';
CHR$(1);'8ths.';CHR$(27);'T';' of an inch.'
```

Once more, do not forget to use double apostrophes in place of the single ones I have used in that line. There are '18' of them in all.

When you GOTO 10000 the result will please you – provided you have entered it correctly, and are using a DMP 2000. It is nice to have such a facility available; but what a lot of characters! It would be better to be able to use shorter versions of each of the codes. And this is possible, using string variables, which are a letter (or more) followed by the dollar sign ($), which is made to equal a string of characters. For instance: 'aa$=PRINT #8,CHR$(15)' is an instance of one string variable being the same as seventeen characters; after it has been entered into the computer's memory.

This is done simply by adding a 'GOSUB 6000' to the end of line 70, and at line 6000 having a subroutine that will make all the variables equal to something. If there is then a 'RETURN' at the end of section 6000, the computer will go there at the end of line 70, put the variables into memory, and return to line 80 to carry on with the program.

On page 3.11 of the DMP 2000 manual is a description of how to do this. However, it does not cover all the many possible variations you could use. Therefore I will devote time in the next chapter to outlining a scheme to do this. For the present I will deal with the present four codes used in that line, using f, ee, and e, because two of the four are the same.

```
70 INK 0,23:INK 1,0:BORDER 23:GOSUB 6000
6000 f$=CHR$(27)+'S'+CHR$(0):ee$=CHR$(27)
     +'T':e$=CHR$(27)+'S'+CHR$(1)
6010 RETURN
```

Don't forget the six double apostrophes, in place of singles in line 6000. There is another point about the making of string variables that should be noted: where you would use a semi colon to separate off any two parts of a code in the normal way, when forming a variable it is necessary to use the plus sign (+) in place of the semi-colon. This is essential.

Now, with line 70 containing GOSUB 6000, and lines 6000 and 6010 in place, the computer will respond to the entering of any of those variables with the code it contains – provided only that RUN is not used; GOTO can be used instead, but not RUN, for that flushes out all the variables when used, and then the variables we have used will not contain the 'CHR$' commands. Each of the variables will contain only the letter or number that is within the double apostrophes. So don't use RUN in such circumstances.

Because I have just made that same mistake (for over an hour I couldn't get the variables to do their job) I have now altered line 60 in this program (in which the keys are redefined). Key 128 is now re-programmed to: 'GOTO 10', instead of to 'RUN', as it was before. (Key 128 is the '0' key on the square block of numbers). GOTO does all that RUN does, without flushing out the variables. With all this in mind, now enter this program:

```
10000 REM * * * * * * * * TEST * * * * * * *
10010 PRINT #8,' This measures ';f$;'7';ee$;'/'
      e$;'8ths.';ee$;' of an inch.'
10020 PRINT #8,' This measures ':CHR$(27);'S';
      CHR$(0);'7';CHR$(27);'T';'/';CHR$(27);
      'S';CHR$(1);'8ths.';CHR$(27);'T';' of an inch.'
```

There are 10 single apostrophes in line 10010 that must be changed to double apostrophes, and 18 in line 10020 that must be changed similarly. The reason has been explained earlier.

If you now use 'GOTO 10000', not 'RUN 10000', the two lines will be displayed on the printer, and will be exactly alike. They

will show more clearly than any explanation can, that the use of the string variables is a great saving, both in typing and in space.

They are shown printed out on page 64.

Do remember, however, that the use of RUN when using such variables as these (variables that contain commands of the CHR$ type) will destroy the function of the variables. In future use only 'GOTO' for programs that may have such variables in them, reserving 'RUN' for the first command after loading the program.

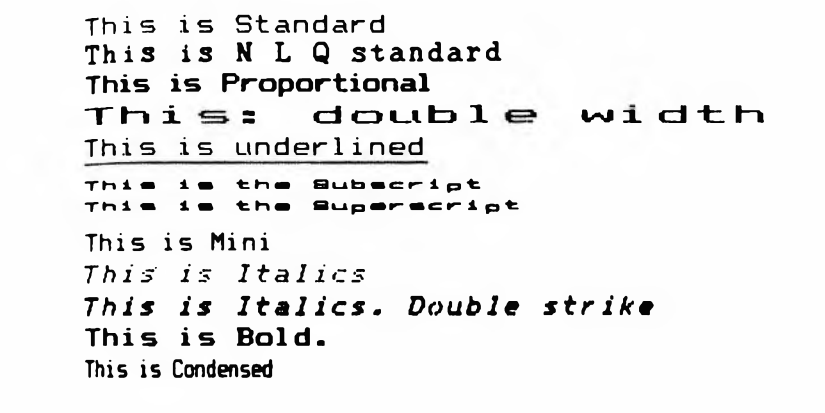
The three test programs, at 5000, 6000, and 10000, are all given at the end of this chapter. They have been printed out in the Condensed type face, using 'WIDTH 60'.

There is a method of printing a double apostrophe in the program, and that is with the use of a re-defined key. It has not been mentioned previously because some regular typists would find it difficult to type. It entails using the letter 7 in the block of letters on the right of the keyboard. This key is re-defined as follows:

```
KEY DEF 10,0,162
```

Put this at the end of line 60 (with a colon first) so that it is the fourth key re-defined, but in a different way. In this fourth case the key code number is taken from page 16 appendix 3 of the Amstrad manual (not page 15 as are the other three keys). It is shown in the list WRITER1B. Now, whenever you want a double apostrophe inside a statement that key can be used.

```
10 REM ************************************
20 REM **** SIMPLE APPLICATIONS OF THE ****
30 REM *** AMSTRAD CPCs  FOR THE WRITER ***
40 REM ************* WRITER1B *************
50 REM ************************************
60 KEY 138,"CLS:LIST 1000-"+CHR$(13):KEY 128,"RUN"+CHR$(13):
KEY 129,"PRINT #S,"
70 INK 0,23:INK 1,0:BORDER 23
80 MODE 2:WINDOW 1,65,1,25
90 S=0:A=3:REM to 8 (incl)
100  WIDTH 65:PRINT #S,CHR$(27);"A";CHR$(12):GOTO 1000
260 PRINT #S:PRINT #S:PRINT #S:PRINT #S,"_":PRINT #S:PRINT #
S,SPC(30);A:PRINT #S:A=A+1:RETURN
270 PRINT #S:PRINT #S:PRINT #S,"_":PRINT #S:PRINT #S,SPC(30)
;A:PRINT #S:A=A+1:RETURN
280 PRINT #S:PRINT #S,"_":PRINT #S:PRINT #S,SPC(30);A:PRINT
#S:A=A+1:RETURN
800 REM from 1010 only:- 19 lines(260):20 lines(270):21 line
s(280):  The rest:  26, 27, or 28.
1000 PRINT #S,"_":PRINT #S:PRINT #S,SPC(20);A:A=A+1:PRINT #S
:PRINT #S:PRINT #S,SPC(5);"SIMPLE APPLICATIONS OF THE AMSTRA
D CPCs FOR THE WRITER":PRINT #S:PRINT #S:PRINT #S,"Chapter O
ne":PRINT #S:PRINT #S
```




```
5000 REM * * * * * * * * TYPE FACES * * * * * * * *
5010 PRINT #8,"This is Standard"
5020 PRINT #8,CHR$(27);"x";CHR$(1);"This is N L Q standard";
CHR$(27);"x";CHR$(0)
5030 PRINT #8,CHR$(27);"p";CHR$(1);"This is Proportional";CH
R$(27);"p";CHR$(0)
5040 PRINT #8,CHR$(27);"W";CHR$(1);"This: double width";CHR$
(27);"W";CHR$(0)
5050 PRINT #8,CHR$(27);"-";CHR$(1);"This is underlined";CHR$
(27);"-";CHR$(0)
5060 PRINT #8,CHR$(27);"S";CHR$(1);"This is the Subscript";C
HR$(27);"T"
5070 PRINT #8,CHR$(27);"S";CHR$(0);"This is the Superscript"
;CHR$(27);"T"
5080 PRINT #8,CHR$(27);"M";"This is Mini";CHR$(27);"P"
5090 PRINT #8,CHR$(27);"4";"This is Italics"
5100 PRINT #8,CHR$(27);"4";CHR$(27);"6";"This is Italics. Do
uble strike";CHR$(27);"H";CHR$(27);"5"
5110 PRINT #8,CHR$(27);"E";"This is Bold.";CHR$(27);"F"
```

```
5120 PRINT #8,CHR$(15);"This is Condensed";CHR$(18)
5130 PRINT #8:PRINT #8
5140 WIDTH 60:PRINT #8,CHR$(15):LIST 5000-10020 #8
5150 PRINT #8,CHR$(18)
5990 STOP
6000 REM * * * * * * MAKING VARIABLES * * * * * *
6010 f$=CHR$(27)+"S"+CHR$(0):ee$=CHR$(27)+"T":e$=CHR$(27)+"S
"+CHR$(1)
6100 RETURN
6900 STOP
10000 REM * * * * * * * * TEST * * * * * * * *
10010 PRINT #8,"  This measures ";f$;"7";ee$;"/";e$;"8ths.";
ee$;" of an inch."
10020 PRINT #8,"  This measures ";CHR$(27);"S";CHR$(0);"7";C
HR$(27);"T";"/";CHR$(27);"S";CHR$(1);"8ths.";CHR$(27);"T";"
of an inch."
```

```
  This measures 7/8ths. of an inch.
  This measures 7/8ths. of an inch.
```

Chapter Eight

MORE USE OF THE DMP 2000

In Chapter Seven I promised to give details of the making of variables to carry the different type faces of the DMP1. One method of doing it is given in the manual, but I feel that it is not quite what I wanted. Therefore I have adopted a system of a single letter for starting the type face, and two of that same letter to cancel it, beginning with 'a'. In addition it was the intention to make it as a unit, transferable from one program to another. With this in mind I put this variable making program at the end of any program in which it will be needed; starting with line 5000; so that it could be MERGEd with any program, so long as room is left for it at 5000 to 5200.

The Amstrad 464, fitted with disc drives, is not keen on any use of MERGE, even though it is supposed (by the manual), to do so. However, the 464 has a tape recorder built in which will do the job very easily. It is my practice, therefore, to keep a spare tape in the recorder for just this purpose. I give this list at the end of this program. It is called MAKING VARIABLES, and a suitable title for recording it is 'MAKEVAR'. Because the title is only 7 characters long it will go on either tape or disc.

To use the tape for 'MERGE', it is only necessary to first be sure there is a space in the program in the computer's memory to allow for lines 5000 to 12000. Then ENTER '|tape' (it doesn't matter whether upper or lower case is used). Now, with the tape number set to coincide with the beginning of the SAVEd program, type in 'Merge', followed by a space and two double apostrophes. Then follow the directions on the screen. This is all that is needed. When the ready signal is given, do not forget to enter '|disc'. If you are not used to discs, the '|' is the one on the key next to P, entered with the shift key.

Owners of Amstrad CPC 664s and 6128s will, of course, have to use a separate cassette recorder.

Before explaining the program 'MAKING VARIABLES', it will be necessary to add one point: at line 90 of WRITER1B, add a couple of GOSUBs, preceded by a REM, so that line 90 now becomes:

```
90 S=0:A=1:REM GOSUB 5000:GOSUB 6000
```

The REM is put there at this point because, while writing the rest of the program those GOSUBs will not be needed. If you wish to print out the result, then you must alter line 90 only. Change the 0 to 8, and take away the REM. This makes it ready. Another warning should be given here. After running the printer with those GOSUBs operative, you must alter back line 90: to do this change the 8 to 0, and put back the REM. In addition you must use RUN to clear the memory of the variables. It is assumed you are following the clear directions given in Chapter Seven to always use GOTO instead of RUN in normal conditions. Also that only command codes that apply to the 'DMP 2000' are put in those lines at 5000 and 6000.

This will perhaps be more clear when it is explained that an order to the printer has a different effect when it goes to the screen, and they come into effect when those GOSUB orders are made operative. When the REM is reinstated, 8 is changed to 0, and RUN is used, they are flushed out.

The locating of the perforation mark on the DMP 2000 is not the same as on the DMP-1, although I use the same principle. On pressing 'ON LINE' on the printer the paper runs back as well as forward by turning the knob. With the perspex cover off I move the paper so that the twelfth hole has not quite reached to the back of the casing edge over which it runs after printing. Some experiments will be called for to get it exactly right, but the result is that with the same paging described for the DMP-1, the DMP 2000 will always put the dash on the perforation.

After those digressions we return to the explanation of the lines at 5000, and at 6000. They are at this chapter end, and a look at them will help. The description of the various codes in use have been given in Chapter Seven. This time note the use of a to k, and aa to kk as the variables used. With the aid of the list given here, it should be easy to select whichever you want.

a=NLQ (Near Letter Quality)
b=Proportional Lettering
c=Double Width lettering
d=Underlined words
e=Subscript (lower level)
f=Superscript (upper level)

g=Mini (smaller than Standard)
h=Italics
i=Double Strike (Makes lettering heavier)
j=Bold (similar to standard, but bolder)
k=Condensed (16 characters to the inch)

'a' or 'aa' applies to Near Letter Quality lettering. a$ puts it into use, and aa$ cancels it. 'b' produces Proportional lettering, (which has less space around slim letters such as 'i' or 'l'). b$ brings it into use, and bb$ cancels it. When using these variables it is easy to see that when a single letter is used to start off a type face, then double that letter will stop it, and bring it back to Standard – or whatever else you command it to. Having once looked up the letter that applies to that face, then there is little thinking to do about cancelling it.

I have printed it out on a piece of card I keep on top of the monitor. There it is in use very often, although I am beginning to memorise some of the letters already.

The use of codes is shown here, and in line 10000 at the end. Line 12000 is the command used to print the list.

'This is a ';k$;'demonstration ';kk$;'of using a few ';h$;'type faces ';hh$;'in a sentence, and includes ';j$;'four ';jj$;'different faces.'

There are fourteen single apostrophes that have to be changed to doubles in that statement. The double apostrophes are around the words to be printed. The codes are separated from them by a semi-colon on each side. Remember the distinction from the way codes are separated in the making of a variable. In the variable a + sign is used to join parts of a variable.

The routine at 6000 is directed at the printer only. It is not used except for this purpose: line 6020 tells the printer to commence printing 3 characters from its normal starting place on the left hand margin. The letter 'l' (lower case L) is here used as the code to trigger this off; the '3' after CHR$ states the number of characters. This is done so that the perforations on each side of the paper can be trimmed off afterwards, leaving a clean sheet, 11 × 8.25 inches.

In line 6010 the 'A', followed by the number after CHR$ will put the printer into double line space mode, as explained in the

previous chapter more fully. This code, 'A', means something different to the computer, and is one of the reasons why it is kept separate by REM (in line 90), and flushed out by RUN, when the printer is not being used. If not treated thus it will make the text or list, on the screen, appear alternatively in reverse colours: white on black, instead of black on white.

The explanations in this and the previous chapter of the ways to use the DMP 2000 with the first 'writer' program are complete enough now to enable anyone to use it competently. It remains to deal with the 'fast writer' program, first described in Chap. Five, and with the improved listing between chapters Six and Seven, which had paging added to it.

To that 'BUTFLY2' listing can now be added some more lines, that will adapt it for use with the DMP 2000. We will first add to it the alteration of 'RUN' to 'GOTO 10' in line 60, for the key that re-programs the 0 in the block of numbers. This is to prevent trouble after using the printer, as explained before. A simple alteration to line 90, which carries the WIDTH command at present, will be the addition of the line spacing code, and a code to permit using a different type face. The BOLD code is being used in this instance, but you can change that to anything you wish. The line must also have a 'REM' in front of it while you are working on the screen, which will be removed or replaced when you change the 0 to 8, or vice versa, (with 8 the REM comes out). The corrected line is given here.

```
90 REM WIDTH 65:PRINT #S,CHR$(27);'A';
CHR$(24):PRINT #S,CHR$(27);'E'
```

Remember to use double apostrophes in place of the single ones in that line; there are four of them.

The way the first code (CHr$(27);'A';CHR$(24)) works has been explained in the previous chapter. Because the REM is replaced when using the screen, there is no need to change the 24 to 12 in this instance.

The second code used after 'WIDTH' is the one to define the type face to be used throughout. In this case the code is for the 'BOLD' type face. It is printed rather more black than is the 'STANDARD', but takes longer to print out. If you change this

BOLD for anything else, remember always to switch off the printer before starting to use a different type face; this makes sure the old type face is not left in the printer's memory.

This revised program, now given the label of 'BUTFLY2B', for use in either tape or disc, is given at the end of this chapter. The line used to provide this listing, without any line number, is now:

```
WIDTH 60:PRINT #8,CHR$(15):LIST 10-690 #8
```

This gives a list exactly 3.5 inches wide, in the Condensed type face, which suits the size of these pages. Both the WIDTH command, and the type face command, can be altered to suit your own requirements. Without the WIDTH command, the line would give a list that stretched right across the sheet – provided you had switched off, and on, the printer first; for it would otherwise have in its memory the 'WIDTH 65' used in the program.

Similarly, the type face code can be changed; that is, all the wording between 'WIDTH' and 'LIST'. The use of the numbers after the word LIST (and before the '#8') tells the printer just how much of the listing to print.

The writer who has not yet started working with a computer, and who might like to try it, would be well advised to obtain an Amstrad CPC464 with disc drive, and a DMP 2000 printer. With a set-up like that, and by using the program 'Writer1' (page 8), any writer will be able to provide professional looking copy. A knowledge of how to handle and modify the program for your own purpose will be gained by the careful study of this book. The index will give a guide to the different sections for future reference.

```
10 REM ************************************
20 REM **** SIMPLE APPLICATIONS OF THE ****
30 REM *** AMSTRAD CPCs  FOR THE WRITER ***
40 REM ************* WRITER1B *************
50 REM ************************************
60 KEY 138,"CLS:LIST 1000-"+CHR$(13):KEY 128,"RUN"+CHR$(13):
KEY 129,"PRINT #S,"
70 INK 0,23:INK 1,0:BORDER 23
80 MODE 2:WINDOW 1,65,1,25
90 S=0:A=1:REM GOSUB 5000:GOSUB 6000
100  WIDTH 65:PRINT #S,CHR$(27);"A";CHR$(24):GOTO 1000
260 PRINT #S:PRINT #S:PRINT #S:PRINT #S,"_":PRINT #S:PRINT #
S,SPC(30);A:PRINT #S:A=A+1:RETURN
270 PRINT #S:PRINT #S:PRINT #S,"_":PRINT #S:PRINT #S,SPC(30)
;A:PRINT #S:A=A+1:RETURN
280 PRINT #S:PRINT #S,"_":PRINT #S:PRINT #S,SPC(30);A:PRINT
#S:A=A+1:RETURN
800 REM from 1010 only:- 19 lines(260):20 lines(270):21 line
s(280):  The rest:  26, 27, or 28.
1000 PRINT #S,"_":PRINT #S:PRINT #S,SPC(28);A:A=A+1:PRINT #S
:PRINT #S:PRINT #S,SPC(5);"SIMPLE APPLICATIONS OF THE AMSTRA
D CPCs FOR THE WRITER":PRINT #S:PRINT #S:PRINT #S,"Chapter O
ne":PRINT #S:PRINT #S
```

```
10 REM ***************************************
20 REM ********* BUTTERFLY ARTICLES ***********
30 REM ***************************************
40 REM *************  BUTFLY2B ****************
50 REM ***************************************
60 KEY 138,"CLS:LIST 1000-"+CHR$(13):KEY 128,"GOTO 10"+CHR$(
13):KEY 129,"DATA "
70 INK 0,23:INK 1,0:BORDER 23:MODE 2:WINDOW 1,65,1,25
80 S=0:A=1
90 REM WIDTH 65:PRINT #S,CHR$(27);"A";CHR$(12):PRINT #S,CHR$
(27);"E"
200 PRINT #S,"_":PRINT #S:PRINT #S,SPC(25);A:A=A+1:PRINT #S:
PRINT #S,SPC(16);"SOME GARDEN BUTTERFLIES":PRINT #S:GOTO 500
250 PRINT #S,STRING$(4,CHR$(10)):PRINT #S,"_":PRINT #S:PRINT
 #S,SPC(28);A:A=A+1:RETURN
260 PRINT #S,STRING$(3,CHR$(10)):PRINT #S,"_":PRINT #S:PRINT
 #S,SPC(28);A:A=A+1:RETURN
270 PRINT #S,STRING$(2,CHR$(10)):PRINT #S,"_":PRINT #S:PRINT
 #S,SPC(28);A:A=A+1:RETURN
280 PRINT #S,STRING$(1,CHR$(10)):PRINT #S,"_":PRINT #S:PRINT
 #S,SPC(28);A:A=A+1:RETURN
290 PRINT #S,"_":PRINT #S:PRINT #S,SPC(28);A:A=A+1:RETURN
300 REM 25 lines = U; 26 lines = V; 27 lines = W; 28 lines =
 Y; 29 lines = Z. Remember to count in the heading and the s
pace under it on the first page.
```

```
500 REM ********** START OF ROUTINE **********
510 FOR n=1 TO 300
520 ws=-1
530 WHILE ws
540 READ z$(n)
550 z$(n)=z$(n)+CHR$(32)
560 WHILE LEN(z$(n))>1
570 cv=INSTR(z$(n),CHR$(32))
580 q$=LEFT$(z$(n),cv-1):z$(n)=MID$(z$(n),cv+1)
590 IF q$="a" THEN PRINT #5,CHR$(8);", ";:GOTO 660
600 IF q$="X" THEN ws=0:GOTO 660
605 IF q$="U" THEN ws=0:GOSUB 250:GOTO 660
610 IF q$="V" THEN ws=0:GOSUB 260:GOTO 660
620 IF q$="W" THEN ws=0:GOSUB 270:GOTO 660
630 IF q$="Y" THEN ws=0:GOSUB 280:GOTO 660
640 IF q$="Z" THEN ws=0:GOSUB 290:GOTO 660
650 PRINT #5,q$+CHR$(32);
660 WEND:WEND
670 PRINT #5
680 NEXT
690 REM ********** END OF ROUTINE **********
```

```
5000 REM ******** MAKING VARIABLES ********
5010 a$=CHR$(27)+"x"+CHR$(1):aa$=CHR$(27)+"x"+CHR$(0)
5020 b$=CHR$(27)+"p"+CHR$(1):bb$=CHR$(27)+"p"+CHR$(0)
5030 c$=CHR$(27)+"W"+CHR$(1):cc$=CHR$(27)+"W"+CHR$(0)
5040 d$=CHR$(27)+"-"+CHR$(1):dd$=CHR$(27)+"-"+CHR$(0)
5050 e$=CHR$(27)+"S"+CHR$(1):ee$=CHR$(27)+"T"
5060 f$=CHR$(27)+"S"+CHR$(0):ff$=CHR$(27)+"T"
5070 g$=CHR$(27)+"M":gg$=CHR$(27)+"P"
5080 h$=CHR$(27)+"4":hh$=CHR$(27)+"5"
5090 i$=CHR$(27)+"6":ii$=CHR$(27)+"H"
5100 j$=CHR$(27)+"E":jj$=CHR$(27)+"F"
5110 k$=CHR$(15):kk$=CHR$(18)
5120 RETURN
6000 REM ********************************
6010 PRINT #S,CHR$(27);"A";CHR$(18)
6020 PRINT #S,CHR$(27);"1";CHR$(3)
6030 RETURN
10000 PRINT #S,"This is a ";k$;"demonstration ";kk$;"of usin
g a few ";h$;"type faces ";hh$;"in a sentence, and includes
";j$;"four ";jj$;"different faces."
12000 WIDTH 60:PRINT #8,CHR$(27);"1";CHR$(8):PRINT #8,CHR$(1
5):LIST #8
```

Chapter Nine

POSTSCRIPT

Further experience with using the WRITER1B program during the period between writing the previous chapters and the proof reading have demonstrated the versatility and adaptability of the program, for some minor modifications have been carried out.

These show how each writer can modify or adapt this program to suit his own requirements. It should be emphasized, though, that it is essential to study and learn all that is explained in the earlier chapters in order to be able to alter programs to suit your own purposes.

The new program, JAMES7, is shown at the end of this chapter, and the differences between this and WRITER1B, will be shown here line by line.

Line 60 contains the first difference, for in it is a means of using a double apostrophe anywhere in a line of text that is bounded by the two double apostrophes.

This has been mentioned in a previous chapter, but here is the explanation of how it works. The computer, in reading the line of text, comes to that first double apostrophe – but it does not read it as a double apostrophe. To the computer it is the symbol for the key 2 when it is used in conjunction with a shift key. Therefore, if we can get another key to produce the double apostrophe, it will not treat that as if it is an apostrophe, or that it comes from key 2 on the main keyboard, but will handle it as it does other letters. In other words it is key 2 being used together with the shift key that triggers off the mechanism to separate off the text.

Therefore, if we can find a different key to carry the small double apostrophe, we can use that. In appendix III, page 8 of the Amstrad manual, is No. 162 in the character set. It is the ideal character for our use, for it is smaller than the proper double apostrophe, its two parts are separated more than in the larger one, and when the two are printed on the screen they look different. Try that now. Enter 'PRINT CHR$(162)', and then just press the shift key and 2 together. You will see two different looking double apostrophes. The machine will also use its 'Syntax error' message, but that will not matter. You will be able to compare the two.

So which key shall we redefine to carry this symbol for us? The one I have chosen in this case is the number 7 in the block of numbers to the right of the main keyboard. To give that a new character a different code to any we have used is needed. This code is the one now given at the end of line 60, and is the only one to use here. It is KEY DEF 10,0,162. On key 10 we now want the symbol 162. The KEY numbers for this purpose are shown on page 16 of appendix III in the Amstrad manual. It only works on using the key WITHOUT THE SHIFT KEY. If you use the shift key it will only print 7.

The only other difference in that line is the transposition of KEY 129 to the first position, instead of the third. This has no effect; it is merely done that way because I happened to type it in first. There are now four commands in that line, and they can be put in any order.

In line 70 there is no change, but in line 80 the WINDOW is changed. It now gives 66 characters to the line, and the text is placed more centrally on the screen. The way this has been done is clear from a comparison of the first two numbers in the command: 1,65, has now become 7,72. The only difference to our working is that line 100 must have the WIDTH changed to 66.

Line 90 has changed considerably. Between the A=4 (A=101 in JAMES 7) and the final REM is now a list of commands for the printer. One of them has been transferred from line 100, and that is the one that spaces out the lines when printing. It is now the third in the list of four printer commands in 90, and its spacing has been changed from 12 to 24 (to give double spacing of the lines).

A REM has been placed at the commencement of these commands, and the first after that is an order to the printer to print in one direction only. It was found (and it may only be a fault in my own machine) that occasionally the printer would come to a stop at a word it had started to spell wrongly at the end of a line. Using this command to print only from left to right has cured this fault for me. It is the command using 'U' +1 for its purpose. It takes a little longer to print each page.

The second command in line 90 uses 'I'+ a number to indicate the distance the printing has to start from the edge of the paper. This places the printed text on the paper at a suitable position to give a better margin on the left hand side of the text. A number

is chosen that suits your own way of working, and you can cut the paper edges off to centre the text in the sheet. If you use a hand guillotine to cut the edges (and this is a reasonable expense for a writer) it is a good plan to put a mark on the board to indicate the cutting places.

The third command has been dealt with, and the fourth is the one that chooses the type face to use. In this case I have used the PROPORTIONAL one, but you should make your own choice. The previous chapters have dealt with that.

The REM at the end of line 90 is for your own use. I find it useful to keep a record of the number of the last page in the chapter I am writing.

In use line 90 is left with the S=0 and a REM after A=101 in cases where you want it to print to the screen. When you need to use the printer, then you must change that 0 to an 8 (S=8), and take out the REM. On wishing to stop using the printer, and print to the screen, you MUST remember to change the 8 into an 0 and replace the REM. Then you MUST use RUN to clear all the odd variables from the computer. When the computer is ordered to RUN it flushes out variables, and re-reads them. The REM stops it from doing this re-reading of that line.

Line 100 is now a simple WIDTH and GOTO line.

The lines 260, 270, and 280 are now changed to 240, 250, and 260. This has been done in order to leave a larger margin at the bottom of each page. At the same time those repeated uses of PRINT #S, have been reduced by using the STRING$ command. If you compare lines 240 on JAMES7 with 260 on WRITER1B you will see how this is done. CHR$(10) tells the computer to print on a line lower, and the string of 4 makes them skip four lines.

Line 800 has been changed to conform with the new lines 240, 250, and 260, and is just a REM line to remind you of which line you use for the number of lines you count.

Line 900 is a new REM line to remind you of the new way to locate the perforated lines between pages. You may wish to use different wording. The new positioning obviates the waste of a plain sheet each time you position the paper. The new position is with the leading edge of the paper just showing (⅛th. inch) beyond the ribbon. The amount showing is subject to a little trial and error, but in any case is an improvement on the method used previously.

In line 1000 is shown the change that has taken place to let you position the paper more accurately. The printing of a dash has been removed, although it is still in place for all the rest of the pages. With this removed (by taking out the first two PRINT #S, statements and the '_') the line starts with the first printing of the page number. That is the only difference, apart from a REM at the end to remind you of the number of lines to count at the beginning of the first page. Under this system the line 1010 starts with the seventh line of counting, and there is now no difference between the counting of the first page and all the succeeding ones.

If you have read and understood the previous chapters, the changing of WRITER1B to JAMES7 should cause you no trouble. The new program is suitable for almost any use a writer can think up in his particular profession; whether it be short articles, or a long novel. A study of the DMP 2000 handbook, and some changes here and there, should enable anyone to produce clean copy.

There has been much talk about word processors, and the way they can prevent bad spelling. I have never yet found one that could differentiate between 'there' and 'their', and this is one of the mistakes a writer does not care to think about; it can happen far too easily. There are many more words that are liable to be confused; think of 'too', 'to', and 'two'. To a computer they are all spent properly, but the sense of them is different.

So far I am convinced that a BASIC program, such as JAMES7, is a far better proposition than a word processor as such. It is the easiest sort to adapt to a different manner of use, for it can quickly be changed into a program to write your letters, for example. There are many other possibilities.

You may probably think of other modifications to the program. If you do, then write to me care of the publisher, for I shall be very pleased to hear about them.

```
10 REM ************************************
20 REM ****** JAMES of LITTLE HEATON ******
30 REM ************************************
40 REM ************ JAMES 7  **:**********
50 REM ************************************
60 KEY 129,"PRINT #S,":KEY 138,"CLS:LIST 1000-"+CHR$(13):KEY 12
8,"RUN"+CHR$(13):KEY DEF 10,0,162
70 INK 0,23:INK 1,0:BORDER 23
80 MODE 2:WINDOW 7,72,1,25
90 S=0:A=101:REM PRINT #S,CHR$(27);"U";CHR$(1):PRINT #S,CHR$(27
);"1";CHR$(6):PRINT #S,CHR$(27);"A";CHR$(24):PRINT #S,CHR$(27);
"p";CHR$(1):REM to 111 (incl)
100 WIDTH 66:GOTO 1000
240 PRINT #S,STRING$(4,CHR$(10)):PRINT #S,"_":PRINT #S:PRINT #S
,SPC(30);A:PRINT #S:A=A+1:RETURN
250 PRINT #S,STRING$(3,CHR$(10)):PRINT #S,"_":PRINT #S:PRINT #S
,SPC(30);A:PRINT #S:A=A+1:RETURN
260 PRINT #S,STRING$(2,CHR$(10)):PRINT #S,"_":PRINT #S:PRINT #S
,SPC(30);A:PRINT #S:A=A+1:RETURN
800 REM for paging:- 24 lines GOSUB 240, 25 lines (250), 26 lin
es (260)
900 REM * * To start set leading edge of paper just 1/8th. inch
 from the ribbon
1000 PRINT #S,SPC(20);A:A=A+1:PRINT #S:PRINT #S,SPC(20);"JAMES
OF LITTLE HEATON":PRINT #S:PRINT #S,SPC(25);"Chapter Seven":PRI
NT #S:REM 6 lines
```

INDEX

Notes

Notes

Notes

Notes

Notes

Notes

Notes

Notes

Please note following is a list of other titles that are available in our range of Radio, Electronics and Computer books.

These should be available from all good Booksellers, Radio Component Dealers and Mail Order Companies.

However, should you experience difficulty in obtaining any title in your area, then please write directly to the publisher enclosing payment to cover the cost of the book plus adequate postage.

If you would like a complete catalogue of our entire range of Radio, Electronics and Computer books then please send a Stamped Addressed Envelope to:

BERNARD BABANI (publishing) LTD
THE GRAMPIANS
SHEPHERDS BUSH ROAD
LONDON W6 7NF
ENGLAND

160	Coil Design and Construction Manual	£2.50
202	Handbook of Integrated Circuits (ICs) Equivalents and Substitutes	£2.95
205	Hi-Fi Loudspeaker Enclosures	£2.95
208	Practical Stereo and Quadrophony Handbook	£0.75
214	Audio Enthusiast's Handbook	£0.85
219	Solid State Novelty Projects	£0.85
220	Build Your Own Solid State Hi-Fi and Audio Accessories	£0.85
221	28 Tested Transistor Projects	£2.95
222	Solid State Short Wave Receivers for Beginners	£1.95
223	50 Projects Using IC CA3130	£1.25
224	50 CMOS IC Projects	£2.95
225	A Practical Introduction to Digital ICs	£1.75
226	How to Build Advanced Short Wave Receivers	£2.95
227	Beginners Guide to Building Electronic Projects	£1.95
228	Essential Theory for the Electronics Hobbyist	£2.50
BP1 - 14	First & Second Books of Transistor Equivalents & Substitutes	£3.50
BP2	Handbook of Radio, TV, Industrial and Transmitting Tube and Valve Equivalents	£0.60
BP6	Engineer's and Machinist's Reference Tables	£1.25
BP7	Radio and Electronic Colour Codes Data Chart	£0.95
BP27	Chart of Radio, Electronic, Semiconductor and Logic Symbols	£0.95
BP28	Resistor Selection Handbook	£0.60
BP29	Major Solid State Audio Hi-Fi Construction Projects	£0.85
BP33	Electronic Calculator Users Handbook	£1.50
BP34	Practical Repair and Renovation of Colour TVs	£2.95
BP36	50 Circuits Using Germanium Silicon and Zener Diodes	£1.50
BP37	50 Projects Using Relays, SCRs and TRIACs	£1.95
BP39	50 (FET) Field Effect Transistor Projects	£1.75
BP42	50 Simple LED Circuits	£1.95
BP44	IC 555 Projects	£2.50
BP45	Projects in Opto-electronics	£1.95
BP48	Electronic Projects for Beginners	£1.95
BP49	Popular Electronic Projects	£2.50
BP53	Practical Electronics Calculations and Formulae	£2.95
BP54	Your Electronic Calculator and Your Money	£1.35
BP56	Electronic Security Devices	£2.50
BP58	50 Circuits Using 7400 Series ICs	£2.50
BP59	Second Book of CMOS IC Projects	£1.95
BP60	Practical Construction of Pre-amps, Tone Controls Filters and Attenuators	£1.95
BP61	Beginners Guide to Digital Techniques	£1.95
BP62	The Simple Electronic Circuit & Components (Elements of Electronics - Book 1)	£3.50
BP63	Alternating Current Theory (Elements of Electronics - Book 2)	£3.50
BP64	Semiconductor Technology (Elements of Electronics - Book 3)	£3.50
BP65	Single IC Projects	£1.50
BP66	Beginners Guide to Microprocessors and Computing	£1.95
BP67	Counter Driver and Numeral Display Projects	£2.95
BP68	Choosing and Using Your Hi-Fi	£1.65
BP69	Electronic Games	£1.75
BP70	Transistor Radio Fault-finding Chart	£0.95
BP71	Electronic Household Projects	£1.75
BP72	A Microprocessor Primer	£1.75
BP73	Remote Control Projects	£2.50
BP74	Electronic Music Projects	£2.50
BP75	Electronic Test Equipment Construction	£1.75
BP76	Power Supply Projects	£2.50
BP77	Microprocessing Systems and Circuits (Elements of Electronics - Book 4)	£2.95
BP78	Practical Computer Experiments	£1.75
BP79	Radio Control for Beginners	£1.75
BP80	Popular Electronic Circuits - Book 1	£2.95
BP82	Electronic Projects Using Solar Cells	£1.95
BP83	VMOS Projects	£1.95
BP84	Digital IC Projects	£1.95
BP85	International Transistor Equivalents Guide	£3.50
BP86	An Introduction to BASIC Programming Techniques	£1.95
BP87	50 Simple LED Circuits - Book 2	£1.35
BP88	How to Use Op Amps	£2.95
BP89	Communication (Elements of Electronics - Book 5)	£2.95
BP90	Audio Projects	£1.95
BP91	An Introduction to Radio DXing	£1.95
BP92	Electronics Simplified - Crystal Set Construction	£1.75
BP93	Electronic Timer Projects	£1.95
BP94	Electronic Projects for Cars and Boats	£1.95
BP95	Model Railway Projects	£1.95
BP97	IC Projects for Beginners	£1.95
BP98	Popular Electronic Circuits - Book 2	£2.25
BP99	Mini-matrix Board Projects	£1.95
BP101	How to Identify Unmarked ICs	£0.95
BP103	Multi-circuit Board Projects	£1.95
BP104	Electronic Science Projects	£2.25

BP105	Aerial Projects	£1.95
BP106	Modern Op amp Projects	£1.95
BP107	30 Solderless Breadboard Projects – Book 1	£2.75
BP108	International Diode Equivalents Guide	£2.25
BP109	The Art of Programming the 1K ZX81	£1.95
BP110	How to Get Your Electronic Projects Working	£1.95
BP111	Audio Elements of Electronics – Book 61	£3.50
BP112	A Z 80 Workshop Manual	£3.50
BP113	30 Solderless Breadboard Projects – Book 2	£2.25
BP114	The Art of Programming the 16K ZX81	£2.50
BP115	The Pre-computer Book	£1.95
BP117	Practical Electronic Building Blocks – Book 1	£1.95
BP118	Practical Electronic Building Blocks – Book 2	£1.95
BP119	The Art of Programming the ZX Spectrum	£2.50
BP120	Audio Amplifier Fault-finding Chart	£0.95
BP121	How to Design and Make Your Own P.C.B.s	£1.95
BP122	Audio Amplifier Construction	£2.25
BP123	A Practical Introduction to Microprocessors	£1.95
BP124	Easy Add-on Projects for Spectrum, ZX81 & Ace	£2.75
BP125	25 Simple Amateur Band Aerials	£1.95
BP126	BASIC & PASCAL in Parallel	£1.50
BP127	How to Design Electronic Projects	£2.25
BP128	20 Programs for the ZX Spectrum and 16K ZX81	£1.95
BP129	An Introduction to Programming the ORIC 1	£1.95
BP130	Micro Interfacing Circuits – Book 1	£2.25
BP131	Micro Interfacing Circuits – Book 2	£2.25
BP132	25 Simple Shortwave Broadcast Band Aerials	£1.95
BP133	An Introduction to Programming the Dragon 32	£1.95
BP134	Easy Add-on Projects for Commodore 64, Vic 20, BBC Micro and Acorn Electron	£2.95
BP135	Secrets of the Commodore 64	£1.95
BP136	25 Simple Indoor and Window Aerials	£1.75
BP137	BASIC & FORTRAN in Parallel	£1.95
BP138	BASIC & FORTH in Parallel	£1.95
BP139	An Introduction to Programming the BBC Model B Micro	£1.95
BP140	Digital IC Equivalents and Pin Connections	£5.95
BP141	Linear IC Equivalents and Pin Connections	£5.95
BP142	An Introduction to Programming the Acorn Electron	£1.95
BP143	An Introduction to Programming the Atari 600/800 XL	£1.95
BP144	Further Practical Electronics Calculations and Formulae	£4.95
BP145	25 Simple Tropical and MW Band Aerials	£1.75
BP146	The Pre BASIC Book	£2.95
BP147	An Introduction to 6502 Machine Code	£2.50
BP148	Computer Terminology Explained	£1.95
BP149	A Concise Introduction to the Language of BBC BASIC	£1.95
BP150	An Introduction to Programming the Sinclair QL	£1.95
BP152	An Introduction to Z80 Machine Code	£2.75
BP153	An Introduction to Programming the Amstrad CPC 464 and 664	£2.50
BP154	An Introduction to MSX BASIC	£2.50
BP155	International Radio Stations Guide	£2.95
BP156	An Introduction to QL Machine Code	£2.50
BP157	How to Write ZX Spectrum and Spectrum + Games Programs	£2.50
BP158	An Introduction to Programming the Commodore 16 and Plus 4	£2.50
BP159	How to Write Amstrad CPC 464 Games Programs	£2.50
BP161	Into the QL Archive	£2.50
BP162	Counting on QL Abacus	£2.50
BP163	Writing with QL Quill	£2.50
BP164	Drawing on QL Easel	£2.50
BP169	How to Get Your Computer Programs Running	£2.50
BP170	An Introduction to Computer Peripherals	£2.50
BP171	Easy Add-on Projects for Amstrad CPC 464, 664, 6128 and MSX Computers	£2.95
BP173	Computer Music Projects	£2.95
BP174	More Advanced Electronic Music Projects	£2.95
BP175	How to Write Word Game Programs for the Amstrad CPC 464, 664 and 6128	£2.95
BP176	A TV-DXers Handbook	£5.95
BP177	An Introduction to Computer Communications	£2.95
BP178	An Introduction to Computers in Radio	£2.95
BP179	Electronic Circuits for the Computer Control of Robots	£2.95
BP180	Computer Projects for Model Railways	£2.95
BP181	Getting the Most from Your Printer	£2.95
BP182	MIDI Projects	£2.95
BP183	An Introduction to CP/M	£2.95
BP184	An Introduction to 68000 Assembly Language	£2.95
BP185	Electronic Synthesiser Construction	£2.95
BP186	Walkie-Talkie Projects	£2.95
BP187	A Practical Reference Guide to Word Processing on the Amstrad PCW 8256 and PCW 8512	£5.95
BP188	Getting Started with BASIC and LOGO on the Amstrad PCW 8256 and PCW 8512	£6.95
BP189	Using Your Amstrad CPC Disc Drives	£2.95
BP190	More Advanced Electronic Security Projects	£2.95
BP191	Simple Applications of the Amstrad CPCs for Writers	£2.95
BP192	More Advanced Power Supply Projects	£2.95
BP193	Starting LOGO	£2.95
BP194	Modern Opto Device Projects	£2.95
BP195	An Introduction to Communications and Direct Broadcast Satellites	£3.95
BP196	BASIC & LOGO in Parallel	£2.95

BERNARD BABANI BP191

Simple Applications of the Amstrad CPCs for Writers

- Many dedicated word processor programs are expensive and so sophisticated that using them detracts the writer's thoughts from the actual job in hand, that is creative writing.

- This book shows how an Amstrad CPC464, 664 or 6128 with disc drives and DMP1 or DMP 2000 printer can be turned into a simple but adequate word processor by using a BASIC program of only fifteen lines. Each of the program lines are dealt with in detail so that even someone encountering BASIC for the first time should be able to use it with the minimum of difficulty.

- A useful addition to the library of all CPC owning authors, writers, journalists, students or anyone who has to prepare or present a typed report, article or manuscript.

ISBN 0-85934-165-8

00295

£2.95

9 780859 341653