ARROW

MADE-EASY

USING YOUR ORIC ATMOS

Garry Marshall

Using your Oric Atmos Made Easy

Garry Marshall

Arrow Books

Using your Oric Atmos Made Easy
Garry Marshall

Arrow Books Limited
17-21 Conway Street, London W1P 6JD
An imprint of the Hutchinson Publishing Group
London Melbourne Sydney Auckland
Johannesburg and agencies throughout
the world

First published 1984

Set in 12pt Stymie medium

Printed and bound in Great Britain
by Commercial Colour Press, London E7

ISBN 0 09 938790 5

CONTENTS

LIST OF FIGURES

Figure No.

Foreword

The main aim of this book is to provide, in the easiest terms possible, an introduction to the Oric Atmos. After guiding you through the process of getting started, it gives an appreciation of the uses to which the Atmos can be put and then an explanation of how it can be made to carry out these tasks. In this way it provides answers to the key questions "How do I use it?" and "What can I use it for?"

The book has three parts. The first is very much an introduction to the computer. It deals with how to get the computer ready for use, how to tell it what to do, and the kinds of things that you can tell it to do. It also describes other items that can be connected to the computer and used in conjunction with it. The Atmos has a number of sockets where various things can be plugged into it, and is well equipped in this way to provide the basis of an expanding system.

The second part of the book deals with how to tell the computer what to do using its own language, that is, how to program the Atmos in BASIC. The third part explores the uses to which the computer can be put by making use of programs that have already been written by someone else. If a program is already available to make the computer do something that you want it to do, then it is only sensible to make use of it rather than writing a similar program from scratch yourself. These two parts correspond to the two main ways of using the computer. Either you can tell it what to do youself, which is interesting, challenging and educational, but time consuming. Or you can use a program that already exists to make the computer carry out a particular task that you need.

Whichever way you choose to use the Atmos, and there is room for both ways, you will find that it is an acquisition that is of immense value. It can be used for profit, for education and for entertainment, and can come to play a part in many different aspects of life. This book can help to get you started and then guide you along the road to using the Atmos to its fullest.

About the Author

Garry Marshall is Principal Lecturer in the Department of Electronic and Communications Engineering at the Polytechnic of North London. He is a regular contributor to computer magazines, including Computing Today and devises a computer puzzle each week for The Observer. He has written 9 books, the majority on computing, including Learning to Use the PET (Gower), Programming with Graphics (Granada) and Programming Languages for Micros (Newnes).

Chapter 1

Introduction to the Oric Atmos

The Oric Atmos is, of course, a personal computer. It must be plugged into the mains electricity to provide it with the power that it needs to operate, and it must also be connected to a television set. Outwardly, it has the appearance of a typewriter keyboard, although it has a few extra keys that an ordinary typewriter does not. When it is ready to use, the Atmos will do nothing until you tell it to or, in some other way, make it. This is because there is almost no limit to the different things that the Atmos can do, and so it must be told what it is to do before it can be made to do it. Like any other computer, the real strength of the Atmos lies in its versatility. It is not like a kettle, which can only heat water, or a washing machine which can only perform its simple function. It has many uses.

A consequence of the computer's versatility is that it is not quite as easy to use a computer as it is a kettle or a washing machine. It is necessary to know *how* to tell a computer what it is that you want it to do. It is also necessary to know *what* the computer itself is capable of doing, particularly if you want to take full advantage of it and to be sure that it is being used for all the things that it is capable of doing for you. This means not only that we must be aware of the computers' capabilities but also that we must know how to communicate our needs to it.

Communicating with the Atmos

You communicate with the Atmos by using its keyboard. Although it is obviously not necessary to be a skilled typist to key in the short words that are the most commonly used commands, you will find that as you progress with the use of the computer there is more and more type and that if you can type 'properly' it will take much less time than if you use one finger 'hunt and peck' keying. So you communicate with the computer by typing at its keyboard and, correspondingly, it communicates with you by showing you what it is doing on the television screen.

Also, when you tell the computer what to do, you cannot use just any words that you choose as you could if you were telling another person what to do. The meaning of "Pass it to me" would, in most circumstances, be quite clear to the person being addressed; But to a computer, with no knowledge of the context in which the command was given, the meaning 'it' would not be apparent, and so the command would not be understood and, therefore, could not be obeyed. For just this kind of reason, the instructions that are given to a computer must be expressed in its own language. They will then have absolutely clear and precise meanings to the computer and it will be able to obey them without fail. The computers' own language is known as BASIC. (It stands for Beginners All-purpose Symbolic Instruction Code). And if the prospect of learning another language is off-putting, remember that you have learnt at least one already and that BASIC is really only a small part of one of them (English) that is specially adapted for telling computers to do the kinds of things that they can do.

What the Atmos can do

So whan *can* the Atmos do? In essence it can store and process information. It can also communicate information, but to do this it must first be connected to the item it is to communicate with or to a means of communication such as the telephone network. By itself, then, the Atmos can store and process information, but this begs the question of what information is. Information may be numbers or letters. It can also be special symbols or strings of letters, that is, words or sentences. In fact, as far as the Atmos is concerned, information is anything that can be typed at its keyboard.

Some examples may help to illustrate the forms that information can take and the ways in which it is processed. A word processing program accepts and stores text typed at the keyboard, and so in this case the information that is stored is words, sentences and paragraphs, although the basic elements are letters and punctuation marks. Once it is stored the information can be processed in any way that the word processor permits. Most word processors can arrange the text in lines of a specified length, make the right hand edge of the text neat and tidy by carrying the last word on each line to finish precisely at the end of the line (this is done by putting extra spaces between the words of each line to push the end of the line across to the necessary position) and arrange the presentation of the end of each paragraph by leaving a specific number of blank lines between one paragraph and the next and indenting the beginning of the new paragraph. As a second example, a database program can store information so that it can be retrieved again in much the same way as it can in a filing system. A company may use it to keep the records of all its transactions and an individual can use it to keep the details of all the videotapes in his or her collection. In this case the information can consist of words, for the title of a videotape, and numbers, for the value of a transaction. The information can be processed by sorting all the records into a particular order or to select all the records meeting a particular criterion. To give a third example, a program for an adventure game must contain descriptions of the various places that can be visited during the game, and these must be stored so that a picture of each can be displayed complete with its treasure and its hazards when it is visited during the course of the game. In this case, the information consists of the descriptions which, in turn, are composed of letters and numbers.

Applications for the Atmos

The consideration of information has, incidentally, brought out some typical examples of the uses to which the Atmos can be put: games, for entertainment and relaxation, word processing for anyone who has to deal with words, whether in writing lettters, producing documents or in writing a book, and databases for any application requiring the storage and retrieval of information. It is notable in all these cases that the applications are direct developments of previous practice and that using a computer for them brings with it definite improvements. Computer games are much more attractive and compulsive than their non-computer predecessors. Among the reason for this are that the computer games are faster moving, more colourful and more realistic than their predecessors. The word processor is the development of the typewriter that has been made possible by the new technology. It makes many aspects of document production much easier than before, including the correction of mistakes, the presentation of documents and the communication of documents. Database programs show to considerable advantage over conventional filing systems. They make the retrieval of information much quicker and easier, particularly when large amounts are involved. There are also slightly unexpected advantages. For example, when recording the details of a company's transactions, the preferred order in which records are stored will be different for different people. The accountant will want one order, perhaps the most profitable first and the least profitable last. A Salesman will want a different order, perhaps just his clients in the order that he deals with them. It is clear that in a conventional filing system the papers can only be stored in one order. But when the records are stored in a computer it can rapidly sort them into the particular order that best suits the current users.

Other applications for the computer include planning with the use of a special program called a *spreadsheet*; education, using the computer to present or to allow the exploration of a body of knowledge; and problem solving in science and engineering.

Should you write your own programs or buy them?

But no matter what use the computer is put to, it must be instructed as to how to behave. If it is used to solve a problem it must be told how to find the solution: if it is to process words it must be told how to do that. In all these cases, the computer must be given a program, that is a set of instructions that tell it how to perform the task in question. When the program is *run* by the computer it automatically carries out all the instructions in the program it has been given one by one. When it has carried out all the instructions in the program it will inevitably have completed the task. Its ability to be programmed is the secret of the computers' versatility. Giving it one program inables it to do one thing. Subsequently giving it another program can make it do another, perhaps quite different, task.

If you know exactly what you want your Atmos to do, you can write a program in BASIC to tell it how to do the task as soon as you are familiar with BASIC. On the other hand, if you can find a program already written by someone else that makes the Atmos do what you want it to then you can buy the program. Programs specially written for the Atmos are supplied by Tansoft. This shows that you do not have to be able to write programs in BASIC to be able to use your Atmos. But if you can write programs you will naturally aquire more familiarisation with the Atmos and with its capabilities.

Undoubtedly, the only way to get the computer to do *exactly* what you want is to write your own programs. Besides making the computer a really personal tool, this activity proves for many a rewarding and absorbing matter. With a lesser knowledge of BASIC than is necessary to write reasonably sophisticated programs it is possible to amend other peoples' programs so that they approach more closely what you want of them. With no knowledge of BASIC you can only use other peoples' programs to tell your computer what to do. Any owner of an Atmos wanting to learn about his computer should learn to program it in BASIC. But it can also be used to advantage with general purpose programs that can be bought from suppliers such as Tansoft for applications such as word processing and information storage. Both a knowledge of BASIC and some commercial programs are needed so that you can enjoy, and employ, your Atmos to the full.

Summary

The Atmos is a small, rather powerful personal computer. Like any other personal computer, it is extremely versatile and capable of achieving many different tasks. To make it carry out a particular task it must be given a program, that is, a list of instructions, that tells it how to do that task. But when one task is completed it can be given another program to inable it to carry out another, perhaps quite different, task. Its programs must be written in a special computer language called BASIC. By mastering BASIC, the users of the Atmos can write programs to make the computer perform any task that they would like done. Programs for specific tasks can also be brought. But whether the Atmos runs the users' own programs or purchased programs, it can be used for many activities from games through education to business applications such as word processing and the storage and retrieval of information.

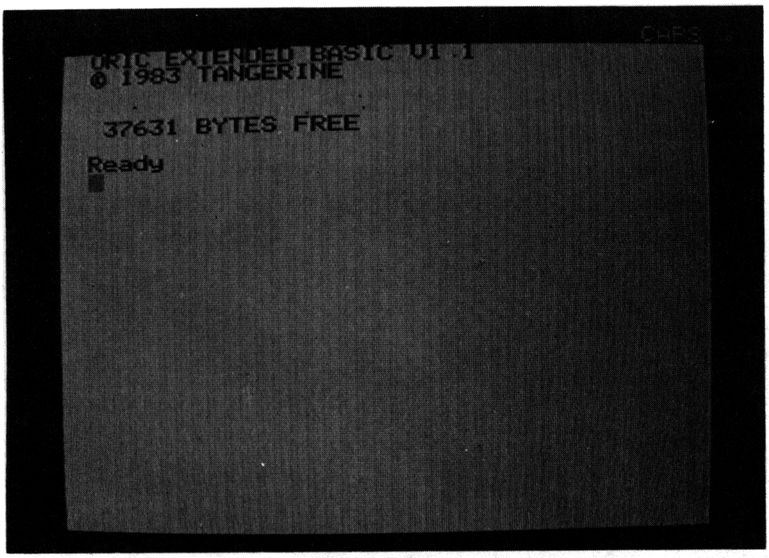
Chapter 2

Getting started

This chapter deals with all the matters that you need to know about how to get started with the Oric Atmos. Actually getting the Atmos ready to use in the first place is just about as easy as it is with a television set, consisting of little more than plugging it in. After describing how to prepare the Atmos for use we examine the keyboard and the purposes of the various keys on it. Then a few of the more common commands for the Atmos are introduced for, as we know, the Atmos does nothing until we tell it what it is that we want it to do. Finally, we examine the various items that can be plugged into the sockets at the back of the computer, thereby expanding the capabilities of the computer and enabling it to become the heart of a quite large system.

Getting the Atmos ready for use

When the box containing the Atmos is first opened, it is found to contain not just the computer itself, but also various cables. To prepare to use the Atmos first of all plug in your television set and turn it on, tuning one of the spare channels to channel 36 if you can, for this is the channel used by the Atmos to produce its display on the television screen. Do not worry if you have no means of locating channel 36, though, for we shall see how to find it easily in a moment. It is also necessary to have another mains socket available for the Atmos. A double socket is best so that there is a socket each for the television and the computer. Now find the cable that is supplied with your Atmos that has a mains plug on one end. Connect the *other* end of this cable to the socket at



```
                                              CAPS
ORIC EXTENDED BASIC V1.1
© 1983 TANGERINE

 37631 BYTES FREE

Ready
```

Figure 2.1 The Atmos intial display.

the back of the Atmos that is marked 'POWER' (it is on the right as you look at the back of the computer) and then plug into the mains. The Atmos is now on and is operating. Next you should find the cable with a connector of the same kind as that on the aerial lead for your television set at one end and connect it to the aerial socket of your television set. The other end of this cable connects to the socket at the back of the Atmos marked

UHF (it is on the left as you look at the back of the computer). When this is done, and it is much quicker to do than to describe, if your television set is turned to channel 36 you will see either at once or quite soon the display shown in Figure 2.1. If your television is not tuned in and all you can see is 'snow', then all that is now necessary is to tune it until you do see this display!

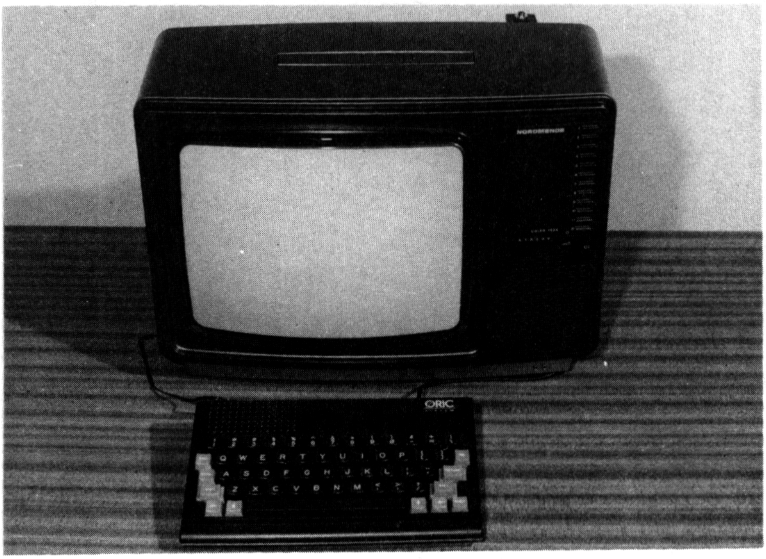


Figure 2.2 The Atmos properly connected and ready to use.

Do not worry about connecting things in the wrong way because it is simply not possible. The connectors at the ends of the various cables are all different and each can be connected only to the proper socket. Be sure to plug in the Atmos before you connect it to the television set, because if you do not do things in this order you can get a pattern of vertical bars on the screen rather than the display of Figure 2.1. The properly connected Atmos should appear as shown in Figure 2.2. There is still one cable remaining in the Atmos' box. This is to attach a cassette unit to the Atmos, and we shall deal later with the way that it is connected.

The meaning of the various items that appear in the initial display is as follows.

ORIC EXTENDED BASIC V1.1 identifies the version of BASIC that the Atmos provides to its users for working programs.

37631 BYTES FREE shows how much memory is available to the user of the Atmos. One byte of memory can store one character, that is, one letter, number or symbol. Consequently, the Atmos can store programs and data consisting of anything up to 37631 characters.

Ready indicates that the Atmos is ready to accept a communication from its user, perhaps in the form of a command or a program.

■ The solid square that blinks on and off quite rapidly is known as the *cursor*. It indicates, by its position, where the next character to be typed at the keyboard will appear on the screen.

If no amount of trying to tune your television set will give a picture that is not blurred or shimmering, you will have to adjust the controls that can be found underneath the Atmos. Their positions can be seen in Figure 2.3. The settings of the two picture controls are set as the Atmos leaves the factory to positions that cause the Atmos to give a good display on most television sets. But it is possible that adjustments may be needed for some television sets. Detailed instructions for adjusting these controls are given in the Atmos' manual. You would be well advised not to touch them unless it is absolutely necessary, as their positioning is quite critical.

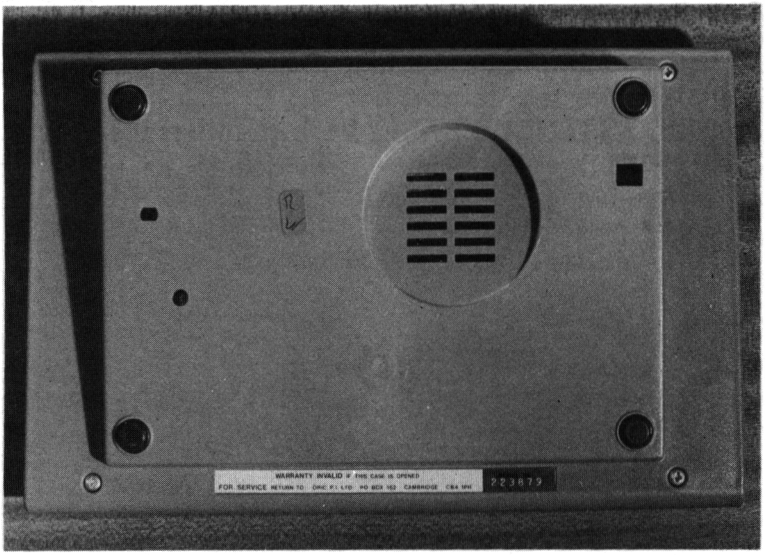

WARRANTY INVALID IF THIS CASE IS OPENED
FOR SERVICE RETURN TO ORIC F.I LTD PO BOX 162 CAMBRIDGE CB4 1PH
223879

Figure 2.3 The bottom of the Atmos showing the controls for running the TV display and the reset button.

The keyboard

The keyboard of the Atmos has 58 keys, including the long space bar at the bottom of the keyboard. Most of the keys are black, but some are red, the distinction between the two sets being that the black keys cause a character to be displayed on the screen while, in general, the red keys are used to change the effect of pressing a black key. The keys are arranged as shown in

Figure 2.4, with the black keys set in rows in the same way as on a typewriter keyboard, in what is known as the QWERTY layout after the first few keys along the top row of letters. The keys all have a full movement and give an audible click when they are pressed. It is quite possible for a trained typist to type at a rapid rate on the keyboard.

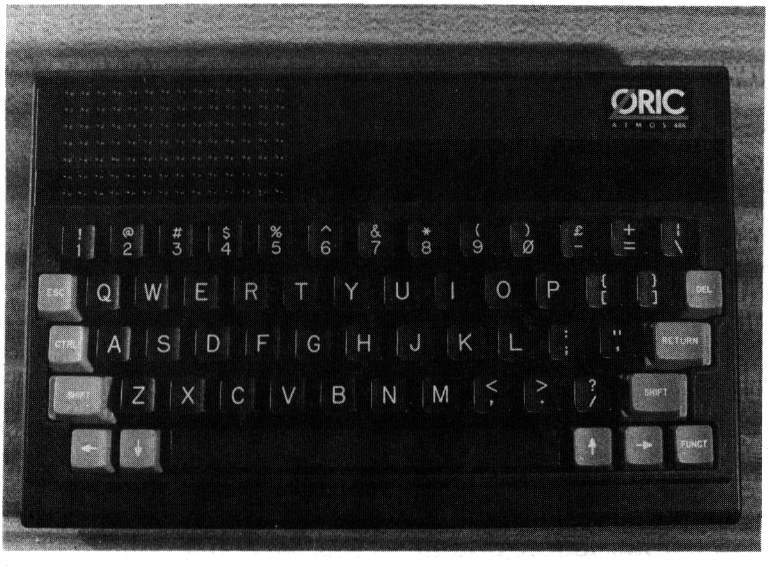

ORIC
ATMOS 48K
ESC Q W E R T Y U I O P DEL
CTRL A S D F G H J K L RETURN
SHIFT Z X C V B N M SHIFT
FUNCT

Figure 2.4 The Atmos' keyboard.

When used, the keyboard gives essentially the same effect as a typewriter keyboard, but with some differences. Considering the black keys first, keys marked with just a letter will initially produce the corresponding capital letter at the position of the cursor on the television screen. (You may have noticed the word 'CAPS' above the top right corner of the display area on the screen, which indicates that the letter keys will produce capital letters). The other black keys all have two symbols marked on them, and pressing any of these keys will place the lower of the two symbols on the screen. To give an example, pressing the key marked with a figure eight below an asterisk will cause the eight to be displayed. Turning now to the first of the red keys, holding down one of the SHIFT keys which are located at either end of the second row of keys from the bottom of the keyboard, and then pressing one of the keys marked with two symbols will cause the upper symbol to appear on the screen. So holding down a SHIFT key and then pressing the key marked with an eight and an asterisk will cause the asterisk to be displayed. But holding down a SHIFT key and pressing a letter key will make no difference, and a capital letter will still result. To produce a small letter, that is, a lower case letter, it is necessary first to hold down the red key marked CTRL and at the same time to press the key marked T. After doing this, and releasing both keys, the word 'CAPS' will disappear from above the display area, and pressing any of the letter keys will cause the corresponding lower case letter to be displayed. The key marked CTRL is referred to as the control key. Holding it down again and pressing T will cause 'CAPS' to reappear and make the keys give capital letters again. In this way, the control key and T cause the Atmos to switch from giving capital letters to giving lower case letters or vice versa. This is typical of the way that the control key is used to cause the computer to switch from one mode of operation to another, and then back again.

At this stage, we know how to type capital and small letters, numbers, punctuation marks and any of the other symbols that appear on the keys. Pressing the long bar at the bottom of the keyboard gives a space. Just by pressing the appropriate keys we can make any character appear on the screen. No matter what symbol we type, it appears at the position marked by the cursor and then the cursor moves one position to the right ready to position the next character to be typed. When a line on the screen has been filled, the cursor automatically moves to the beginning of the next line so that the next character to be typed is placed there. By typing some text you can watch the lines that you type build up on the screen. Since all the keys repeat, merely holding down a key will quickly fill a line or move with the character marked on that key. But the Atmos will not accept anything you type that consists of 80 or more characters. Pressing the key for the 75th, 76th, 77th, 78th and 79th characters in anything you type causes a bell to ring to remind you that the limit is approaching. If you go on to key the 80th character, the Atmos will print a '/', move the cursor to the beginning of the next line, and ignore what you have just typed. You will also find that when you are typing on the bottom line of the screen and reach its end, the contents of the screen all automatically move up by one line, or 'scroll' upwards, to leave the cursor at the beginning of a blank bottom line. The previous top line vanishes. So, when typing at the Atmos' keyboard, characters appear on the screen in essentially the same way as they would appear on paper when using a typewriter. In fact, the Atmos makes it rather easier because there is no need to perform the equivalent of the typewriter's carriage return to start on a new line, or to change the paper when a page is full.

The functions of the red keys that have not been mentioned so far are as follows.

The keys marked with arrows. These keys, positioned in pairs on either side of the space bar, are for moving the cursor around the screen. They move it in the directions marked on the keys so that text can be positioned anywhere on the screen by moving the cursor to the position at the beginning of the text by using these keys before typing the text itself.

RETURN. This key terminates a line of text and at the same time sends it to the computer for the computer to act on. If, for example, a command has been typed then pressing RETURN will send it to the computer which will then obey it.

DEL. This key deletes the symbol that has just been typed and, more generally, the key to the left of the cursor's position. It allows typing errors to be corrected as soon as they are made.

CTRL. We have seen that holding down CTRL and pressing T causes the Atmos to switch between giving capital letters and small letters. In much the same way, holding down CTRL and pressing F will cause the click associated with pressing a key to be suppressed or to start again. Holding down CTRL and pressing Q will cause the cursor to vanish or to reappear. By contrast, holding down CTRL and pressing G always causes the bell to ring. Other effects associated with CTRL will be covered as they are needed, and a complete list of them is given in Appendix 1.

ESC. This key, called the escape key, is used in conjunction with other keys in the same way as the control key to produce different colours and effects on the display screen. A full list of them is also given in Appendix 1.

FUNCT. This key, the function key, alters the characters produced by the letter keys in much the same way as the SHIFT key.

Giving commands to the Atmos

We give commands to the Atmos just by typing the correct words, and the Atmos will obey them as soon as we press RETURN. When issuing a command it must be typed in capital letters, so if 'CAPS' is not showing above the display area, then press CTRL and T. Even if you type a command correctly in lower case letters the Atmos will respond, when you press RETURN, with

```
? SYNTAX ERROR
```

This is its way of indicating that you have typed something which is not one of the commands to which it can respond.

The first command we will introduce is for clearing the screen so that we can start work in a position equivalent to that of someone starting to write with a clean sheet of paper. To clear the screen, all that is necessary is to type

```
CLS
```

and press RETURN. Try it!

Since we have already seen that computers store information, process it and display the results, let us see next how we can tell the computer to do these things. We can instruct the computer to store an item in its memory by using a command starting with the word 'LET'. Then we write a name, which can be any name we choose as long as it starts with a letter and consists of letters and numbers only. This will be the name under which the item is stored. Then we write an equals sign (=) and finally the item that is to be stored.

To store the number 2.5 under the name 'number', we write

```
LET NUMBER = 2.5
```

and as soon as we press RETURN it is done. The word 'let' can be omitted, so that we can write just

```
NUMBER = 2.5
```

but for the moment we will retain 'LET' to show consistency with the other commands that we can give, all of which have a characteristic key word associated with them.

A word, that is, a string of letters, or indeed any sequence of characters, can be stored in nearly the same way. The difference is that we must tell the computer that we are storing something that is not a number, and we do this by placing a dollar sign ($) at the end of the name we choose. Additionally, the word to be stored must be enclosed in quotation marks. By doing these things we can store the word 'ATMOS' under the name A$ by

```
LET A$ = "ATMOS"
```

or, as before, just by

```
A$ = "ATMOS"
```

A complete sentence can be stored just as easily by

```
LET SENTENCE$ = "LET'S USE THE ATMOS"
```

or

```
SENTENCE$ = "LET'S USE THE ATMOS"
```

If we have a whole number to store, we can store it in the same way as any other number, but it takes up less space in memory if we store it under a name that ends with a percentage sign (%). So, we could write

LET TWO = 2

but it is better to have

LET TWO % = 2

SUMMARY BOX 1

Storing information

LET name = item

1. **The effect of the command is to store the item of information 'item' under the name 'name'.**
2. **LET may be omitted.**
3. **'name' must start with a letter and then can be followed by any letters or numbers.**
   It must end with $ if anything other than a number is being stored.
   It should end with % if a whole number is being stored.
4. **'item' can be a number or a string of characters enclosed in quotation marks.**

Now that we know how to store items of information in the computer's memory, we can look at how to process them. In its simplest forms, processing can also be done with the LET command. If we consider numbers first, we can write arithmetic expressions exactly as we would in ordinary arithmetic, and using brackets, except that the usual multiplication sign is replaced by an asterisk (*) and the usual division sign by a slash (/). And on the right hand side of the equals sign in a LET command, we can write an arithmetic expression rather than just a number. The computer obeys such a command by finding the value of the arithmetic expression and storing it under the given name. If we type

```
LET X = 2 * 3 + 1.5
```

and press RETURN, the computer will calculate 7.5 as the value of the arithmetic expression on the right and store this number under X, the name given on the left. After this command, typing

```
LET Y = X - 1.1
```

causes 6.4 to be stored under Y because the value of the expression on the right is the number stored under X less 1.1, and this is stored under the name given on the left.

SUMMARY BOX 2

Dealing with numbers

LET name = arithmetic expression

1. **The effect of the command is to find the value of 'arithmetic expression' and to store it under 'name'.**
2. **LET may be omitted.**
3. **'name' is any name under which a number may be stored.**
4. **'arithmetic expression' is any properly written arithmetic expression that involves names under which numbers are stored, numbers and the arithmetic operators +, −, *, /.**

Words and character strings can also be processed using the LET command, and we will give one simple example of how this can be done. The plus sign can be used with words, as well as with numbers, although in this case it takes a very different meaning from its usual arithmetic one. The result of 'adding' one character string to another is a single character string consisting of the first string followed at once by the second. Thus, after

```
LET A$ = "ATMOS"
```

and

```
LET P$ = "PHERE"
```

the value of A$ + P$ is "ATMOSPHERE", and the effect of

```
LET Q$ = A$ + P$
```

is to store the single word "ATMOSPHERE" under the name Q$.

SUMMARY BOX 3

Dealing with words

LET name = word expression

1. **The effect of the command is to find the value of 'word expression' and to store it under 'name'.**
2. **LET may be omitted.**
3. **'name' is any name under which a word may be stored.**
4. **'word expression' is any number of words or names under which words are stored all separated by plus signs.**
5. **The value of 'word expression' is the single word consisting of all the individual words in the expression one immediately after the other**

The command for displaying information on the screen begins with the word 'PRINT'. When followed by a string of characters inside quotation marks it will print these characters on the screen. By typing

```
PRINT "THIS IS ATMOSPHERIC"
```

and pressing RETURN, we cause the message

```
THIS IS ATMOSPHERIC
```

to appear on the screen.

But if PRINT is followed by a name, or a list of names, it causes the items stored under these names to be displayed on the screen. In this way, we can examine items that we have stored in the computer's memory and also the results of processing them. So, after giving the commands

```
LET F = 1.2
LET G = 2 * F + 0.1
LET A$ = "ATMOS"
LET B$ = A$ + "PHERIC"
```

we find that

```
PRINT G
```

gives 2.5, that

```
PRINT B$
```

gives ATMOSPHERIC, and that

```
PRINT A$, F, B$
```

gives ATMOS 1.2 ATMOSPHERIC

SUMMARY BOX 4

DISPLAY

PRINT character string
and
PRINT list of names

1. **The effect of PRINT followed by a string of characters inside quotation marks is to display the string of characters.**
2. **The effect of PRINT followed by a list of names is to display successively the items stored under each name in the list.**

More commands

The commands that have been explained in the previous section for storing, manipulating and displaying information are much the same on any computer. In this section we shall examine some commands that are much more specific to the Atmos.

So far, the Atmos' display has shown black letters and symbols on a white background, giving the same appearance as black letters printed on white paper as in a book. But the Atmos is a colour computer, and we can change the colours of its display. The commands that are provided for this are INK and PAPER. As you have probably guessed, INK is for changing the colour in which letters and symbols are displayed on the screen, and PAPER is for changing the background colour against which they are displayed. The names of these commands are easy to remember because INK has the same effect as if you change the colour of the ink in your pen when writing on paper, and PAPER gives the same effect as changing the colour of the paper you are writing on. We must tell the Atmos what colour we want to change to, and this is done by following INK and

PAPER by a number from 0 to 7 which represents a colour as shown in the following table.

Table 1 **Colours and their numbers**

Number	**Colour**
0	Black
1	Red
2	Green
3	Yellow
4	Blue
5	Magenta
6	Cyan
7	White

To see blue letters on a yellow background, all that is necessary is to give the commands

```
INK 4
PAPER 3
```

pressing RETURN after each. Now you can experiment with the colours to see which combination gives the most pleasing effect, and also that which is most restful to your eyes in a session of computing. Also, if the display on your television set is anything less than perfect this might be a good time to use the controls underneath the Atmos to correct it, as the display of yellow is a vital part of the tuning procedure. (Again, the manual gives all the details of the tuning method). If you use the same colour for INK and PAPER, the text on the screen becomes invisible, and the cursor is the only thing that shows on a display that is a complete block of single colour. But the text is still present, and changing one of the INK or PAPER colours will make it visible again.

If you prefer to use the names for the colours, rather than their numbers, which is a much more natural thing to do, then you can store the numbers under the corresponding names with the commands

```
LET BLUE = 4
LET YELLOW = 3
```

and so on, and then give commands which have obvious meanings, such as

```
INK BLUE
PAPER YELLOW
```

This seems preferable to having to remember the numbers to the colours (we use the computer's memory to hold them rather than our own), but there is a snag which arises from the Atmos' rules for the names under which items are stored. Although a name can be any sequence of letters and numbers that begins with a letter, the Atmos only uses the first two characters of the name. The only purpose of having long names is to make them easier for *you* to read or to remember their purposes. A consequence of this is that if you issue the command

```
LET BLUE = 4
```

and then type

```
LET BLACK = 0
```

the subsequent command

```
PAPER BLUE
```

will cause the background colour to become black! As you have probably seen, this is because LET BLUE = 4 has caused the Atmos to store 4 under the name BL, but then LET BLACK = 0 has caused it to store 0 under BL, thereby replacing the previously stored 4. Then PAPER BLUE is treated by the Atmos as PAPER BL, which gives PAPER 0, and when it is obeyed it causes the background to become black. From this, we must learn the lesson that when choosing names, in any circumstances, it is necessary to ensure that no two names begin with the same two characters or unexpected outcomes will occur.

SUMMARYBOX5

Colour

INK number
PAPER number

1. **The effect of INK is to set the colour in which characters appear on the display and of PAPER is to set the colour of the background on which they appear.**
2. **'number' must be a number from 0 to 7. Each corresponds to a colour as shown in the table above.**

The Atmos also allows you to create drawings on its screen. In this way you can create pictures or *graphics*. To prepare the computer to do this, you must first give the command

```
HIRES
```

On pressing the RETURN key, you will see the screen clear to black except for the bottom three lines which remain in the current INK and PAPER colours. The black part of the screen provides a 'CANVAS' on which drawings can be created. Anything that is typed appears in the bottom part of the screen. Before introducing the graphics commands, a little background explanation is necessary.

The 'canvas' provided by the Atmos consists of a rectangular array of dots. The dots are arranged in rows and columns, and there are 240 columns and 200 rows. The columns are numbered from 0 to 239 and the rows from 0 to 199 as shown in Figure 2.5. This means that any dot on the canvas can be identified by giving the number of

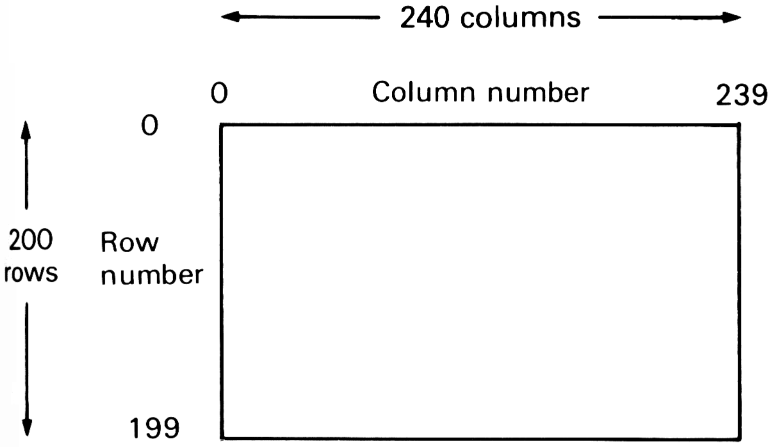


Figure 2.5 The graphics screen of the Atmos.

the column that it is in and the number of the row that it is in. (For those familiar with geometry this is exactly the same as identifying the position of a point by giving its X – coordinate and its Y – coordinate). When it is producing graphics, the Atmos maintains a *graphics cursor* to mark the position at which the next thing will occur in just the same way as it maintains a cursor during its normal mode of operation. The graphics cursor can be positioned by using the command CURSET. To position it at the point in column 50 and row 50, and to plot a point there, we give the command

```
CURSET 50, 50, 1
```

The command DRAW allows us to draw a line starting from the current position of the graphics cursor. To draw a line from there to a point 25 columns to the right and 25 columns down, we give the command

```
DRAW 25, 25, 1
```

The graphics cursor moves as the line is drawn, and after the command has been obeyed, it is positioned at the end of the line. So we can now draw a second line starting from the end of the first one with

```
DRAW -25, 25, 1
```

The negative sign with the column number means draw to a position 25 columns *to the left* of the current position of the graphics cursor. A negative sign with the row number would mean draw to a position so many rows *up* from the cursor position. The shape drawn by the previous two commands and these:

```
DRAW 50, 0, 1
DRAW -25, -25, 1
DRAW 25, -25, 1
DRAW -50, 0, 1
```

is shown in Figure 2.6.

When you have finished drawing pictures in this way, you can return the Atmos' screen to normal by giving the command

```
TEXT
```

and pressing RETURN.

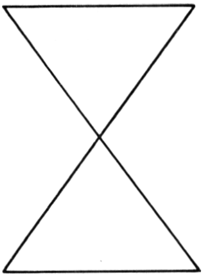


Figure 2.6 A shape created with DRAW.

SUMMARY BOX 6

Graphics

HIRES
CURSET column, row, number
DRAW c, r, number
TEXT

1. HIRES sets the Atmos for displaying graphics.
   CURSET positions the graphics cursor
   DRAW draws a line.
   TEXT returns the Atmos from graphics display to its normal mode of display.
2. 'column' and 'row' give the number of the column and row in which the graphics cursor is to be positioned.
3. When a line is drawn with DRAW it is plotted from the current position of the graphics cursor to a point 'c' columns and 'r' rows from it.

4. 'number' can be a number from 0 to 3, but if it is 1 the CURSET plots a point and DRAW draws a line in the foreground colour.

Expanding the computer

There are several sockets at the back of the Atmos at which items of equipment can be connected to it. They are shown in Figure 2.7. The sockets are all clearly labelled. Perhaps the best way to describe how the Atmos can be expanded is to examine in turn the uses intended for these sockets. We will consider the sockets

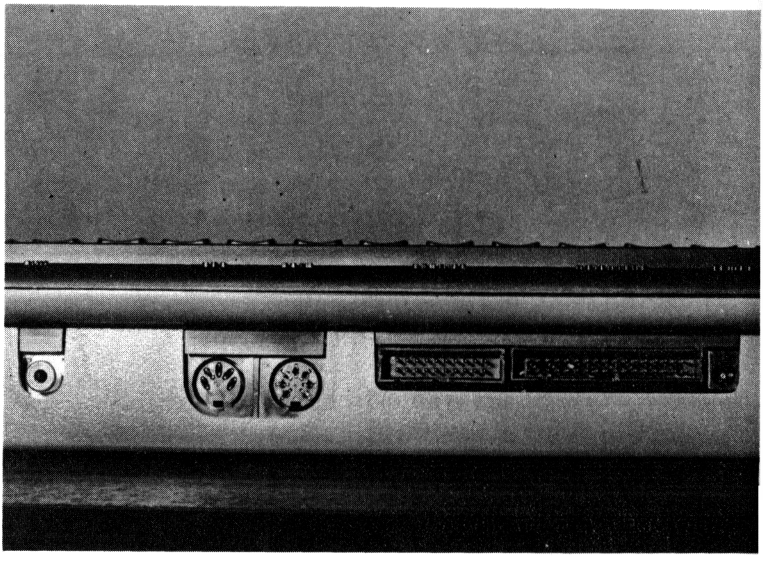


Figure 2.7 The sockets at the back of the Atmos.

taking them in order from left to right as they appear when viewing the Atmos from behind. The uses of the two sockets at the extreme left and right that are labelled VHF and POWER have already been covered in the course of describing how to get the Atmos ready for use. The purposes of the others are listed below.

RGB. RGB stands for red-green-blue. This socket is for attaching a monitor rather than a television set to the Atmos to provide a screen for its display. The picture on a monitor is much clearer and sharper than that displayed on a television set. This makes a monitor more suitable for displaying graphics and for viewing during long periods of use. But a monitor is more expensive than a television set and, unlike a television set, is not normally to be found in the home.

Tape. This DIN socket is for attaching a cassette player to the Atmos using the cable supplied with the computer. The cassette player is used mainly to make a copy of programs that have been typed into the Atmos, so that they can be saved permanently, and to enable programs that have previously been saved on a cassette to be transferred to the computer to be run by it. Use of the cassette unit is covered in the next chapter.

Printer. This socket is for the connection of a printer to the Atmos. It provides a standard, so-called Centronics, connection so that any printer of this type can be connected at once to the Atmos. If the computer is to be used, for example, for word processing, then it is clearly essential to have a printer attached to the Atmos so that the documents that are produced can be printed. It also allows a printed copy of the program that is in the computer to be made so that the printed copy can be taken away from the computer and examined at leisure, perhaps to correct or amend the program.

Expansion. Unlike the other sockets which are specialised, this one allows a large number of different things to be attached to the Atmos. Disc drives can be attached to provide a much faster means of saving and entering programs than the cassette player. A light pen can be attached to provide an alternative to the keyboard as a means of communicating with the computer, as can a joystick and for the same purpose although it tends to be used in games. A modem can also be connected to allow the Atmos to communicate information over the telephone network either by sending it or receiving it.

Summary

The Atmos is made ready for use by a simple procedure that involves plugging it into the mains electricity and attaching it to a television set. The procedure is foolproof because the cables that are supplied can only be plugged into the sockets for which they are intended. From the keyboard a full range of letters, numbers and symbols can be typed. Generally, they are typed at the black keys, although the red keys must be used to modify the effects of the black keys in order to allow the full range of characters to be produced. The red keys all have special purposes, but the CTRL and ESC keys in particular can be used in conjunction with other keys to produce effects ranging from ringing a bell to changing the colours of the display. Once the keyboard is mastered, commands can be typed for the Atmos, and this chapter has introduced commands for handling information, for changing the colours in the display and for creating graphics. We are now ready to start to write programs.

Chapter 3

Writing simple BASIC programs

We can now begin to write some simple programs of our own, because the commands that we met in Chapter 2 provide us with a small BASIC vocabulary. As we progress, we shall see that the programs we can create using only the commands with which we are familiar are quite restricted. But BASIC provides more facilities than we have met, and as we feel the need to do extra things with our programs we shall find that BASIC supplies us with the means to do it. In this way, we shall introduce more of the features of BASIC as they are needed to create programs for performing particular tasks. This approach brings out the reasons that BASIC possesses the features that it does as well as illustrating the uses to which they can be put.

First programs

As we know, a program is a list of instructions. In BASIC, an instruction is a numbered command, that is, a command preceded by a number. The number is usually referred to as a line number. An instruction is entered, in exactly the same way as a command, by typing it and pressing the RETURN key. But the computer reacts differently when it receives a program instruction to when it receives a command. Whereas a command is obeyed at once, an instruction is stored. The computer stores a program by storing all the individual instructions of the programs. The reasons for storing the program are that once it is stored the computer can execute it as often as you like (in contrast to a command, which must be typed every time it is issued) and that it can be corrected or amended just by changing the necessary instructions rather than by retyping it all.

When the instructions of a program are being entered they can be typed in any order because the Atmos automatically stores them in the order given by their line numbers, placing the instruction with the lowest line number first, that with the next lowest second, and so on up to the last instruction, which has the highest line number. When the Atmos executes the program that is stored in it, it does so by taking the instructions one by one, in the same order in which they are stored, and obeying the command part of each.

SUMMARY BOX 7

A program

1. **A program is a list of instructions.**
2. **An instruction is a line number followed by a command.**
3. **Instructions are stored in the increasing order of their line numbers.**
4. **A program is executed by obeying its instructions one by one in the order given by their line numbers.**

We can now write a program, and since it is the first one, we will make it do the simple task of storing two numbers, finding the difference between them and displaying the result. Although this task is small enough for us to plan a way of doing it in our heads, we shall put the planning down on paper because we shall soon come to tasks that do need quite careful planning before we can begin to write the programs that tell the computer how to carry them out. If we were telling the computer how to do this task by giving it commands, we should begin by storing two numbers in the computer. This can be done by

```
LET FIRST = 6.8
LET TWO = 2.3
```

We could find their difference with

```
LET SUB = FIRST - TWO
```

Storing a caption to identify the result would be a good idea, and could be done by

```
LET C$ = "THE DIFFERENCE IS"
```

so that we could display the result with

```
PRINT C$,SUB
```

We have given five commands, which have been sufficient to accomplish the task. These commands give us the basis of our program. All we have to worry about now is making sure that the computer stores them in the right order so that they will be executed in the proper order when the program is run. To do this we must add line numbers to the commands, to turn them into program instructions, making sure that the command that we gave first gets the lowest line number, that which we gave second the next lowest, and so on. It might seem natural to use the numbers 1, 2, 3. . . as line numbers, but we shall use 10, 20, 30 and so on. The reasons for this will become apparent later on, but it is basically so that there are gaps between the line numbers in case we want to add some extra lines to the program.

From this, we get our first program as:

```
10 LET FIRST = 6.8
20 LET TWO = 2.3
30 LET SUB = FIRST - TWO
40 LET C$ = "THE DIFFERENCE IS"
50 PRINT C$;SUB
```

Each line is entered just by typing it and pressing RETURN. To see the program that is stored in the computer at any time, you give the command

```
LIST
```

and, as soon as you press RETURN, the program that is stored in the computer is displayed on the screen. To execute the program that is stored in the computer, give the command

```
RUN
```

If the program that is given above has been typed *exactly* as it is presented, after typing RUN and pressing RETURN you will see the line

```
THE DIFFERENCE IS 4.5
```

before the 'READY' message and the cursor are displayed again. If you do not get this line, and in particular if you get

```
?SYNTAX ERROR
```

do not worry, it is probably because what you have typed is not exactly the same as the program that is listed above. See if you can find where your program differs and if it does simply retype the line or lines that are different and then run the program again.

SUMMARY BOX 8

Examining and executing a program

LIST
RUN

1. **The command LIST causes the program that is stored in the computer to be displayed on the screen.**
2. **The command RUN causes the program that is stored in the computer to be executed**

Our first program should illustrate that when the computer runs the program stored in it, it selects the command parts of the program instructions one by one, in the order given by the line numbers, and obeys them. The end result is exactly the same as when the commands are typed individually in the same order, but in either case the order is important. To get the commands in the wrong order would be to give the computer the wrong method for carrying out its task. Of course, the computer completes its task much more quickly by running its stored program than it does when we have to type the commands individually .

A program to store these words and then to display a phrase made up from them can be written along the same general lines as the first program. Although it will deal with words rather than numbers, it will store, process and display them in the same pattern. Before we start to enter the new program we should give the command

NEW

to clear the previous program from the Atmos' memory. If we do not do this, we may well end up with parts of the old program mixed up with the new one, giving a stored program that is neither one thing nor another.

SUMMARY BOX 9

Clearing a program

NEW

1. **This command clears the computers memory of the program that is stored in it, and should be used before entering a new program.**

Our second program is:

```
10 LET A$ = "STORE "
20 LET B$ = "THE "
30 LET C$ = "PROGRAM "
40 LET Z$ = A$ + B$ + C$
50 PRINT Z$
```

Running this program will give the display

```
STORE THE PROGRAM
```

Note that each word is stored with a space following it. If this is not done, when the words are 'added' by line 40, the result will be a string of letters with no gaps and all the words will run on from each other. It is easy to make this program produce another phrase. Typing

```
40 LET Z$ = BS + C$ + A$
```

will cause this line 40 to replace the previous one, and when the amended program is run it will give the display

Our third program is a graphics program. We can write a program that creates the shape of Figure 2.6 by incorporating the commands that we gave in Chapter 2 for producing this shape into a program. The resulting program is:

```
 10 HIRES
 20 INK 4
 30 PAPER 3
 40 CURSET 50,50,1
 50 DRAW 25,25,1
 60 DRAW -25,25,1
 70 DRAW 50,0,1
 80 DRAW -25,-25,1
 90 DRAW 25,-25,1
100 DRAW -50,0,1
```

The consequence of putting the instructions in the wrong order can be demonstrated graphically with this program, for by changing the line numbers of a few instructions to give the following program:

```
 10 HIRES
 20 INK 4
 30 PAPER 3
 40 CURSET 50,50,1
 50 DRAW 25,25,1
 60 DRAW 25,-25,1
 70 DRAW 50,0,1
 80 DRAW -25,25,1
 90 DRAW -25,-25,1
100 DRAW -50,0,1
```

We find that the shape it produces is that shown in Figure 3.1 which is quite different from that given by the original program.

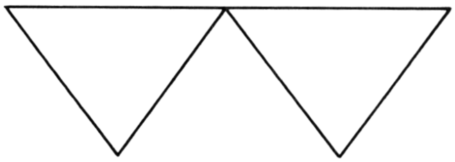


Figure 3.1 A second shape created by DRAW.

Editing

Now that you have typed a few programs, it is worth considering how to correct any mistakes that may occur during typing. These methods of correcting errors apply to the text of programs or to anything else that you enter. They are worth knowing because they are the easiest and quickest ways to correct the errors that inevitably occur while typing.

Whatever you are typing, you can correct a mistake if you notice it as soon as it is made by deleting it with the DEL key. The DEL key always deletes the character to the left of the cursor, and if the mistake that you want to correct is not at the end of the text then, with the cursor at the end of the text, you can always delete all the characters up to the erroneous one and then start typing again from there. It is tempting to use the cursor movement keys to move the cursor to the left of the erroneous character and then press the DEL key to delete it. This can be done, and it leaves a space in the position that was occupied by the deleted character in which the correct character can be typed. But this procedure is not recommended, for it can be used to create text which appears correct on the screen but which is not acceptable to the Atmos. For instance, if you are trying to type RUN, but type RAN and notice it before pressing RETURN while the cursor is still after the N, then you could use the left-arrow key to move the cursor to the left (onto the N) and press delete to delete the A, leaving the cursor on the space created between the R and N. Then typing U and moving the cursor one place to the right puts RUN on the screen, as required, and leaves the cursor after the N. But pressing RETURN now causes the Atmos to give the message ?SYNTAX ERROR. It cannot recognise RUN when it has been created in this way as the command to run a program despite the fact that it appears to be perfectly correct on the screen.

In practice, it would be much quicker to abandon the incorrect line at once, by holding down CTRL and pressing X, and to type RUN properly from scratch.

There is an alternative, which is to use CTRL and A to copy the character on which the cursor is positioned. By using this combination of keys, our RAN can be turned to RUN by copying the R, typing a U and then copying the N. To show how this is done in detail, we assume that we are at the beginning of a new line having pressed CTRL and X to abandon RAN or having just pressed RETURN after RAN to be greeted by a ?SYNTAX ERROR message. Use the cursor movement keys to place the cursor on the R press CTRL and A together to copy the R and make the cursor move over the A. Then type U, causing the cursor to move onto the N and press CTRL and A to copy the N. This sends R, U and N in succession to the special memory that is used to hold all typed messages. Pressing RETURN now sends the command RUN to the Atmos to be obeyed. In fact, when the cursor is positioned over a particular character, the effect of pressing CTRL and A is exactly the same as pressing the key with that character on it. We could give the command RUN by finding the letters R, U and N anwhere on the screen, perhaps in a program listing, and placing the cursor on them in succession and pressing CTRL and A before finally pressing RETURN. The snag with using CTRL and A to copy the letters of a command in this way is that it is not possible to see what has been copied before RETURN is pressed to cause it to be obeyed. For this reason it is probably still preferable to abandon an improperly typed command and retype it properly, particularly as most of the commonly used commands are quite short.

When correcting or changing a program, there are several methods that can be used. Any instruction can be replaced simply by typing a new version of the instruction and giving it the same line number. An instruction can be inserted at any point in a program by giving it a line number between those of the two instructions betwen which it is to be placed. An instruction can be deleted just by typing its line number and pressing RETURN. But if a fairly long instruction contains just a simple error, then it is better to use CTRL and A to copy the correct part and to type only the correction rather than to retype the whole instruction. If the correction is only a question of typing one character to replace another, then by using CTRL and A to copy the correct characters and typing the required replacement over the erroneous character there is no need to leave the line of the instruction. But if the correction requires the insertion of more characters than are to be removed, then it is necessary to leave the line, because if the extra characters are typed on the same line they will replace characters that should remain there to be copied. To cope with this, the cursor movement keys may be used to take the cursor off the line occupied by the instruction, so that the letters required may be typed without obliterating anything, and then to move it back onto the line where copying can resume. The EDIT command is provided to facilitate this, and the effect of EDIT followed by a line number is to display the instruction with that line number below the cursor with blank space around it for typing insertions.

To illustrate all this, consider altering the line

```
50 PRANT OC
```

so that it reads

```
50 PRINT ORIC
```

We begin by typing the command

```
EDIT 50
```

When the instruction is displayed, use the cursor movement keys to place the cursor on the 5 at the beginning of the instruction. Holding down CTRL, press A repeatedly until the cursor is on the A in the middle of PRANT. Then type I to replace the A and press CTRL and A repeatedly until the cursor is on the final C. We want to insert R and I without obliterating the C, so we press the cursor down key, type R and then I, and press cursor up once and cursor left twice to put the cursor back on the C. Press CTRL and A to copy it, and finally press RETURN to send the corrected line to the Atmos to replace the previous one. Provided all this has been done correctly, the next time you list the program you will see that line 50 is as we wanted it. In this case, because corrected program lines can be examined by listing the program, it is possible to view the results of editing by using CTRL and A before running the program.

SUMMARY BOX 10

Editing test

1. The DEL key deletes the character to the left of the cursor.
2. Pressing CTRL and X causes the current line to be abandoned.
3. Pressing CTRL and A copies the character under the cursor. Pressing RETURN sends the string of copied characters to the Atmos.

SUMMARY BOX 11

Editing a program

1. An instruction is inserted just by typing it and pressing RETURN.
2. An instruction is deleted by typing its line number and pressing RETURN.
3. An instruction is replaced by entering a new instruction with the same line number.
4. CTRL and A can be used to edit instructions.
5. The command EDIT followed by a line number will cause the instruction with that line number to be displayed conveniently for editing.

More BASIC instructions

In this section, we shall introduce some more BASIC instructions. The ones we have met already allow us to give rather simple programs to the computer, but if we are to write programs for more advanced tasks we shall need instructions that are capable of rather more than those we have met so far. As a basis for introducing the new features and for showing why we need them we shall consider the problem of writing a standard letter to send to one of our friends on her birthday. A simple program to do this using only the commands that we have met already is given below. It stores the lines of the letter one by one, then clears the screen and displays the lines one by one. The program is:

```
 10 LET A$ = "DEAR NANCY"
 20 LET B$ = "THIS IS TO WISH YOU A HAPPY
    BIRTHDAY"
 30 LET C$ = "AND MANY HAPPY RETURNS OF
    THE DAY"
 40 LET D$ = "LOVE"
 50 LET E$ = "GARRY"
 60 CLS
 70 PRINT A$
 80 PRINT B$
 90 PRINT C$
100 PRINT D$
110 PRINT E$
```

When this program is run, it displays on an otherwise clear screen

```
DEAR NANCY
THIS IS TO WISH YOU A HAPPY BIRTHDAY
AND MANY HAPPY RETURNS OF THE DAY
LOVE
GARRY
```

As soon as you have written this program, it may strike you that the same greeting could be sent to any of your friends if only you could change NANCY to another name. In fact, if you could give the name of the friend whose birthday it is to the computer, it could produce a personal greeting merely by inserting the name in the appropriate place in the otherwise standard letter. By providing the INPUT instruction, BASIC allows information, such as the name of a friend, to be given directly to a program for purposes such as this.

In a program, when the computer reaches an instruction such as

```
10 INPUT N$
```

it first prints a question mark on the screen and then waits for you to type in something from the keyboard and to press RETURN to show that you have finished. The question mark is displayed as a way of reminding you that you should type something to give to the computer. What you type is then accepted and stored, in this case under the name N$. The computer then goes on to deal with the next instruction. The INPUT Instruction also allows you to display a message rather than just a question mark, so that you can tell the computer to display a suitable message to remind you of what you are supposed to type. In this case we could use:

```
10 INPUT "NAME OF FRIEND";N$
```

This makes the computer display:

```
NAME OF FRIEND?
```

and await your response, accepting it for storage under N$ when you press RETURN.

With the use of this instruction, we can write the program for producing a birthday greeting to any of our friends as:

```
10 LET A$ = "DEAR "
13 INPUT "NAME OF FRIEND";N$
17 A$ = A$ + N$
```

```
 20 LET B$ = "THIS IS TO WISH YOU A HAPPY
    BIRTHDAY"
 30 LET C$ = "AND MANY HAPPY RETURNS OF
    THE DAY"
 40 LET D$ = "LOVE"
 50 LET E$ = "GARRY"
 60 CLS
 70 PRINT A$
 80 PRINT B$
 90 PRINT C$
100 PRINT D$
110 PRINT E$
```

Note that to get this program we need only edit line 10 of the previous program and insert lines 13 and 17. Further if we had not left a gap between the numbers used for line numbers there would have been no way to insert these extra lines.

SUMMARY BOX 12

Entering data from the keyboard

INPUT name
or
INPUT character string; name

1. **When executed this instruction causes a question mark or, if it is given, 'character string' followed by a question mark, to be displayed and then the computer waits for a response to be typed.**
2. **The response, which is terminated by typing RETURN, is stored under 'name'.**

BASIC provides another way of including information in a program with its READ and DATA instructions. You may wonder why yet another way of handling information is needed. The comparison of BASIC with ordinary English is helpful in answering this. In English there are usually several ways of saying much the same thing, except that each way has its own shade of meaning and emphasis that makes it particularly well suited to a given situation. The same is true with BASIC, where a particular way of telling the computer to carry out some action, and in this case of telling it how to record information, will be more suitable than another. Being able to choose from different methods of telling the computer to do what we want allows us to pick the one that is most suitable or most convenient.

Various items of data can be made available to the computer when it is runnning a program by placing them in a DATA statement in the program. A DATA statement consists of the word DATA followed by the items, all of which are separated by commas. We can place the four words 'HAPPY', 'BIRTHDAY', 'TO' and 'YOU' in a a DATA statement in this way:

```
DATA "HAPPY", "BIRTHDAY", "TO", "YOU"
```

The order of the items is important, for the first READ statement that is obeyed by the computer in a program reads the first item of data from the DATA statement, the second READ instruction that is obeyed reads the second item, and so on. If we cannot fit all the items of data into one DATA statement, or if we choose not to, then we may use several DATA statements, and the line numbers of the DATA statements indicate the order of the items of data in a quite natural way. The following program will read the words from a DATA statement and display them in the order in which they appear:

```
10 READ A$
20 PRINT A$
30 READ B$
```

```
40 PRINT B$
50 READ C$
60 PRINT C$
70 READ D$
80 PRINT D$
90 DATA "HAPPY", "BIRTHDAY", "TO", "YOU"
```

But the words could equally well have been given in four DATA statements as:

```
 90 DATA "HAPPY"
100 DATA "BIRTHDAY"
110 DATA "TO"
120 DATA "YOU"
```

as this presents the items of data in exactly the same order as they appeared in the single DATA statement used originally.

It is probably clear from this that another way of making the computer produce our birthday greeting to Nancy that uses READ and DATA is:

```
 10 CLS
 20 READ A$
 30 PRINT A$
 40 READ B$
 50 PRINT B$
 60 READ C$
 70 PRINT C$
 80 READ D$
 90 PRINT D$
100 READ E$
110 PRINT E$
120 DATA "DEAR NANCY"
130 DATA "THIS IS TO WISH YOU A HAPPY
    BIRTHDAY"
```

```
140 DATA "AND MANY HAPPY RETURNS OF THE
    DAY"
150 DATA "LOVE"
160 DATA "GARRY"
```

SUMMARY BOX 13

Including data in a program

READ name
DATA data list

1. When executed, a READ instruction reads an item of data from 'data list' and stores it under 'name'.
2. A DATA statement may contain a 'data list' consisting of several items of data separated by commas. A program may contain several DATA statements.
3. The items in a DATA statement are ordered from left to right. When there is more than one DATA statement in a program the order of the items starts at the first item in the statement with the lowest line number, and finishes at the last item in the statement with the highest line number.
4. The first time that a READ statement is executed in a program, the first item of data is read, the second READ statement to be executed reads the second item, and so on.

Making it easier to write programs

The briefest look at the previous two programs shows that they are very repetitive. Further, if we want to make the computer produce longer messages, writing the program that tells the computer how to go about it will become very boring. The fact that the programs are repetitive suggests that we ought to be able to simplify them. After all, our aim is to make the computer carry out tasks for us, but there is little advantage in this if it takes as much effort to tell the computer how to do them as it would for us to do them ourselves.

In the previous two programs, pairs of instructions such as:

```
READ A$
PRINT A$
```

are repeated over and over. Really, we would like to write down the pair of instructions once only. If we could do this and then tell the computer that as soon as it has obeyed them once it should go back and do them again, we should be able to tell the computer to do a good deal of work by writing only a few instructions.

BASIC provides us with a primitive means of doing this in the form of the GOTO instruction. It takes the form of the word 'GOTO' followed by a line number, and when it is obeyed it causes the Atmos to go to the instruction with the specified line number and to obey the command in that instruction next. By using this instruction, our birthday greeting can be produced by the much shorter program:

```
10 CLS
20 READ A$
30 PRINT A$
40 GOTO 20
50 DATA "DEAR NANCY"
60 DATA "THIS IS TO WISH YOU A HAPPY
   BIRTHDAY"
70 DATA "AND MANY HAPPY RETURNS OF THE
   DAY"
80 DATA "LOVE"
90 DATA "GARRY"
```

The program works very well except for one thing which you will notice when you run it. Although the program produces the greeting we expect, it then displays:

```
? OUT OF DATA ERROR IN 20
```

We do not really want this message to appear at the end of the output from our program (it certainly has nothing to do with Nancy's birthday). If you thought that the GOTO instruction was rather too good to be true, you may feel that your worst fears are confirmed. But at least we should investigate a little further to find the cause of this unwanted message.

When the computer runs the previous program line 20 causes it to read an item from the DATA statement, line 30 then causes it to print what it has just read, and line 40 causes the computer to go back to line 20 to read another item. Even after the last item of data has been read, line 40 still sends the computer back to line 20 to read another one. Since there are no more items of data to read, the computer cannot do what we have told it to do, and it automatically sends us a message to indicate its protest. The message 'OUT OF DATA ERROR IN 20' actually means 'there were no more data items to be found when the READ instruction with line number 20 was being carried out.

This occurrence is typical of what can happen when a GOTO instruction is used by itself.; The error has caused the computer to stop in this instance, but if we enter the program

```
10 PRINT "THE COMPUTER WILL NOT STOP."
20 GOTO 10
```

then running this program will cause the computer to keep displaying the sentence from line 10 for ever. To stop it, we must take some action ourselves. There are two possibilities: we can either hold down CTRL and press C or we can turn the Atmos over and press the RESET button (its position is shown in Figure 2.3).

SUMMARY BOX 14

Changing the order in which instructions are obeyed.

GOTO line number

1. **The effect of this instruction is to cause the computer to go to the instruction with the line number given by 'line number' and to obey it next. The computer then carries on with the program by obeying the instruction following it.**
2. **When used to jump back to an instruction with a lower line number, it must be used with care as it creates a loop from which there is no escape.**

Having seen that the GOTO instruction when used by itself is potentially quite dangerous, we should mention that BASIC provides other instructions with which it can be used quite sensibly. But, for the purposes of repetition, the Atmos provides us with a pair of instructions that are far superior. These are REPEAT and UNTIL. When they are used, UNTIL is followed by a test, and they have the effect of causing all the instructions between the REPEAT and UNTIL to be executed repeatedly until the test following UNTIL is found to be true.

In our greeting program, we want to read items and print them out until the last item of data has been read. We know that the last item is the name that goes at the bottom of the greeting, and so the program can now be written as

```
 10 CLS
 20 REPEAT
 30 READ A$
 40 PRINT A$
 50 UNTIL A$ = "GARRY"
 60 DATA "DEAR NANCY"
 70 DATA "THIS IS TO WISH YOU A HAPPY
    BIRTHDAY"
 80 DATA "AND MANY HAPPY RETURNS OF THE
    DAY"
 90 DATA "LOVE"
100 DATA "GARRY"
```

The two lines that are to be obeyed repeatedly have been indicated to emphasise which they are. The test with UNTIL in line 50 is to see if A$ = "GARRY", that is, if the item most recently read and stored under the name A$ is 'GARRY'. This test will only be true when the last item has been read.

SUMMARY BOX 15

Repetition

REPEAT
UNTIL test

1. This pair of instructions causes the instructions placed between REPEAT and UNTIL to be executed repeatedly until the 'test' following UNTIL is satisfied.

We started with a program to send a birthday greeting to a particular person, and then made the program more useful so that it could send a birthday greeting to any of our friends. It would be even more useful if it could be used to send any greeting to any person. We shall now develop the previous program to make it do this. First, the program must ask for the name of the person to whom the greeting is to be sent, and then it must ask for the greeting itself. After that it must read the other parts of the message and display them. But we can make the program use the same simple repetitive format if we place dummy items in the data such as NAME to represent any name and GREETING to represent any greeting. As long as the computer can replace NAME when it reads it by DEAR followed by the name we have given it in response to an INPUT instruction and, similarly, can replace GREETING by the greeting we have given it, the task we have set the computer can be carried out.

Before we can tell the computer how to do this, we must introduce a further BASIC instruction. We need to tell the computer something like 'if you have read 'NAME' then replace it by 'DEAR' followed by the name we just gave you'. And for purposes such as this, BASIC provides us with an instruction involving the words IF and THEN. This instruction allows us to describe to the computer how to make decisions in much the same way as we should describe decision-making in ordinary English. The instruction takes the form of the word 'IF' followed by a test (of the same kind as used with UNTIL), the word 'THEN' and finally another BASIC command. The test can be a comparison of two values to see if they are equal or to see if they differ. (Other comparisons can also be made, and they are given in the Summary Box.) The command following 'THEN' can be almost any BASIC command. When an instruction of this type is carried out, the test is made, and if it is satisfied the command following 'THEN' is obeyed, otherwise nothing more is done and the computer moves on to the next instruction. In this way, we can let the computer decide whether or not to carry out a command by looking at the result of a test.

The instruction that tells the computer to display 'THREE' only if the value stored under X is 3 is:

```
IF X = 3 THEN PRINT "THREE"
```

The instruction that recognises when 'NAME' is stored under A$ and replaces it by 'DEAR' followed by the name stored under N$ is:

```
IF A$ = "NAME" THEN LET A$ = "DEAR " + N$
```

By using this conditional, IF-THEN, instruction, we can write our program to send any of our friends any greeting as:

```
10 INPUT "FRIEND'S NAME"; N$
20 INPUT "GREETING"; G$
30 CLS
40 REPEAT
50 READ A$
60 IF A$ = "NAME" THEN LET A$ = "DEAR " + N$
70 IF A$ = "GREETING" THEN LET A$ = G$
80 PRINT A$
90 UNTIL A$ = "GARRY"
100 DATA "NAME", "GREETING", "LOVE",
    "GARRY"
```

When this program is run, its output is similar to this:

```
FRIEND'S NAME?   JOANNE
GREETING?   HAPPY NEW YEAR
```

The screen then clears before displaying

```
DEAR JOANNE
HAPPY NEW YEAR
LOVE
GARRY
```

SUMMARY BOX 16

Making decisions

IF test THEN command

1. **The 'test' can compare two values for equality (=) for non-equality (< >). It can also see if one value is greater than (>) or less than (<) another.**
2. **The 'command' can be any BASIC command or sequence of commands.**
3. **When executed, the 'command' is obeyed only if the 'test' is satisfied.**

Saving and loading programs with a cassette player

We now turn to the use of a cassette player with the Atmos. This is usually the first thing to be connected to the computer. Its main uses are to copy the program that is stored in the computer onto a cassette, so that the program is saved in a permanent form, and to put a program that has previously been saved on cassette into the computer. The reasons for doing this are that when the computer is unplugged everything in its memory is lost, including any program stored there. If you have previously keyed in a long program, it would be discouraging to know that the next time you needed the same program you would have to type it in all over again. By saving it on a cassette, you can load it again directly from there. Also, if you buy a program a copy of it must be given to you in some form. You could be given a listing of it on paper but, again, you would have to type what is, in all probability, a lengthy program. If it is supplied on a cassette it can be loaded from the cassette player. A disc drive can be used with its discs to load and save programs in just the same way, and it is much quicker. But most households possess a cassette player while disc drives are rather expensive.

A 3-pin DIN to 3-pin DIN cable is supplied with the *Atmos and this can be used to connect the Atmos* to a cassette player via the socket marked TAPE on the computer and the one marked RECORD/PLAY on the cassette player. Figure 3.2 shows the way in which they should be connected.

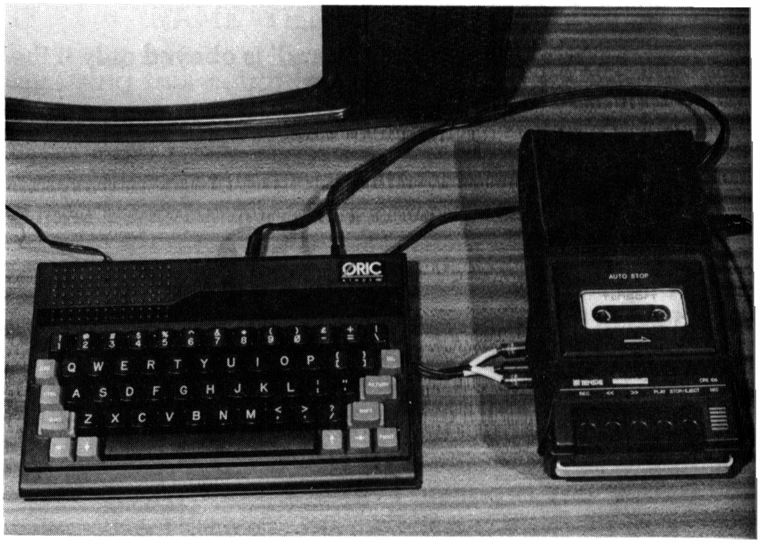


Figure 3.2 A cassette player attached to the Atmos.

For recording programs it is better to use a short C10 or C15 cassette than the longer C60 and C90 tapes. The magnetic tape itself is thicker and consequently less liable to stretch than it is on a longer tape, making it less likely for the recording of a program to be corrupted. It is also quicker to locate a program on a shorter tape, and it results in the waste of much less time should the computer not find the program you want on a cassette for any reason. The Atmos' commands for saving and loading a program on a cassette are CSAVE and CLOAD.

With your cassette player attached to the Atmos, plugged in, and loaded with a blank cassette, the procedure for saving the program that is stored in the computer by copying it on a cassette is as follows:

* Wind the tape forward a short way so that it is well past the plastic reader.

* List the program that you want to save on the screen.

* Type the command.

CSAVE "PROGRAM",S

* Press the RECORD and PLAY buttons on your cassette recorder

* Press RETURN on the ATMOS

'PROGRAM' is the name that the program will be saved under when you give this command, but you may choose any name you like with up to 16 characters in it. The S at the end stands for slow. The Atmos can save programs in either a fast or a slow way, but the slow method is more reliable.

* The message SAVING PROGRAM will appear above the display area (or SAVING followed by the name that you chose). The READY message will appear with the cursor when the program has been saved, and you should then stop the cassette player.

By following this procedure you make a copy of the program that is stored in the computer on a cassette. The program itself remains in the computer.

The procedure for loading a program that has previously been recorded on a cassette into the computer is as follows:

* Place the cassette containing the program in the cassette player, and rewind the tape.
* Type

```
CLOAD "PROGRAM", S
```

If the program was saved under a name other than PROGRAM, then you must use that name. If S for slow was used with the CSAVE command when the program was saved, then you must use S in the CLOAD command. If you just want to load the first program on a cassette (and this is a good reason for recording only one program per side on a cassette), you can type

```
CLOAD "",S
```

* Press the RETURN key on the Atmos and then press the PLAY button on the cassette player.

* The message SEARCHING... will appear above the display screen and this will change to LOADING PROGRAM while the program is being loaded. The message will vanish when loading is finished, and the READY message and cursor will reappear.

* There is one modification to CLOAD that is worth mentioning, because it allows us to verify that a program has been saved correctly on cassette. It does this by comparing what has been saved on the tape with what is stored in the computer, and when it has finished it reports the number of differences. If no differences are reported, we can be confident that an exact copy of the program has been recorded.

Immediately after saving a program using, for example the command that we gave above

```
CSAVE "PROGRAM",S
```

we can verify that we have made an exact copy of the program in the computer by the following procedure:

* Rewind the cassette.
* Type the following command

```
CLOAD "PROGRAM",V,S
```

* Press the RETURN key on the Atmos and the PLAY key on the cassette player.

* The message SEARCHING. . . will appear above the display area, followed by VERIFYING PROGRAM, and this will vanish when verification is complete. If the program has been correctly copied onto cassette, the Atmos will report

```
Ø Verify errors detected
```

otherwise it will report the number of errors that it did detect, and the whole process of saving the program must be repeated.

SUMMARY BOX 17

Loading and saving programs

```
CLOAD "name",S
CSAVE "name",S
```

1. **The effect of CLOAD is to copy the program called "name" from a cassette tape into the computer.**
2. **The effect of CSAVE is to copy the program in the computer onto cassette tape under the name "name".**

Summary

A BASIC program is a sequence of instructions, and an instruction is a numbered command, with the number showing the position of the instruction in the sequence and the order in which it must be stored and executed. We have written our first programs by noting that a program for a particular task can be constructed by placing numbers in front of each of a set of commands that we know can cause a task to be accomplished. This must be done in such a way that they remain in the same order as before. The means provided by the Atmos for editing text are described. They are particularly useful when editing or correcting a program.

By means of an extended example based on writing a program to produce a letter, we see how extra facilities are needed to make the program more generally useful and that, in every case, BASIC provides the facilities we need. In this natural way, BASIC's facilities for handling data, for repetition and for making decisions are introduced. After this, we have met many, although by no means all, of the instructions of BASIC.

Finally, we look at how a cassette player can be used in conjunction with the Atmos. It allows us to store the program that is stored in the computer so that we do not lose it when the computer is unplugged, as we would otherwise. It also allows us to load programs that are recorded on cassette into the computer so that we have a convenient way to load the programs that we have previously saved and that we have bought.

Chapter 4

Graphics and sound

The Atmos is well-equipped for producing both colour graphics and sound. We have already seen a little of its ability to produce highly detailed colour graphics, and it can produce a wide range of sounds, musical and otherwise, through its internal loudspeakers. The Atmos' BASIC posesses quite a number of BASIC instructions for graphics and for sound production. We shall meet some of them and see how they can be used in this chapter.

GRAPHICS

We have alrealy met some of the commands that are used in connection with high-resolution graphics. INK and PAPER set the foreground and background colours of a graphics display. CURSET and DRAW are used, respectively, to position the graphics cursor and to draw a line from the graphics cursor to a point of specified distance away.

We shall start by writing a graphics program that can be adapted easily to draw any shape because it reads a description of the shape from a DATA statement. By changing the DATA statement only, the program can draw any shape at all. The DATA statement gives the number of lines that have to be drawn to make the shape, and then the same number of pairs of numbers each showing where each line must be drawn to. The program is:

```
10 HIRES
20 INK 3
30 PAPER 4
40 LET C=0
50 CURSET 120,100,1
60 READ N
70 REPEAT
80 READ COL, ROW
90 DRAW COL, ROW, 1
100 LET C=C+1
110 UNTIL C=N
120 DATA 4,0,-40,40,0,0,40,-40,0
```

Under the name C a count is kept of the number of lines that has been drawn so far, and the program plots lines repeatedly until the correct number have been drawn. As it is presented, the program causes a square to be drawn.

This program shows that it is sometimes useful to have a form of repetition for situations in which the number of repetitions required is known in advance. BASIC provides us with this in the form of the pair of instructions FOR and NEXT. The advantage of using this pair of instructions in the appropriate situation is that it provides a counter that is automatically incremented, thereby saving us the trouble of writing the instructions for it ourselves. When this form of repetition is used, the instructions to be repeated are placed between a FOR and a NEXT instruction, and FOR must be followed by a name under which a counter can be stored together with the starting and finishing values for the count. To illustrate this, the previous program can be rewritten using FOR and NEXT as:

```
10 HIRES
20 INK 3
```

```
30 PAPER 4
40 CURSET 120,100,1
50 READ N
60 FOR C = 1 to N
70 READ COL, ROW
80 DRAW COL, ROW, 1
90 NEXT C
100 DATA 4,0, –40,40,0,0,40, –40,0
```

This program has two instructions fewer than the previous one because the instructions for maintaining the counter are no longer needed.

SUMMARY BOX 18

Fixed numbers of repetitions

```
FOR name = start TO finish
             block of instructions
       NEXT name
```

1 The effect of this instruction pair is to cause 'block of instructions' to be executed a fixed number of times.

2 A counter is stored under 'name'. It starts with the value 'start' and is incremented each time the block of instructions is obeyed. The final repetition is done with the value 'finish' stored under 'name'.

3 The total number of repetitions is 'finished' — 'start' +1.

When the CURSET and DRAW commands are used, three numbers must follow them. The first two specify a position, and so far we have always made the third 1. But the third number can take any value from 0 to 3 inclusive, and each gives a different effect. They are listed in the following table.

Table 2 **Effects given by the third number used with CURSET and DRAW**

Number	Effect
0	Points are plotted in the background colour.
1	Points are plotted in the foreground colour.
2	Points are inverted, so that if they were in the background colour they are plotted in the foreground colour and vice versa.
3	Points are not affected.

These effects give us ways of erasing lines, effectively by 'unplotting' them, as well as ways of plotting lines or moving the graphics cursor without plotting anything. One application of 'unplotting' is in animation. We can make a shape appear to move across the screen by plotting it in one position, unplotting it, and then moving the position across the screen before repeating the process. A program to do this with our square is:

```
10 HIRES
20 INK 3
30 PAPER 4
40 LET COL = 20
50 REPEAT
60 CURSET COL,100,0
70 DRAW 0, -40,1
80 DRAW 40,0,1
90 DRAW 0,40,1
100 DRAW  -40,0,1
110 DRAW 0, -40,0
```

```
120 DRAW 40,0,0
130 DRAW 0,40,0
140 DRAW  -40,0,0
150 LET COL=COL+1
160 UNTIL COL=160
```

The movement can be speeded up by increasing the number of columns that the shape moves across the screen from the one set by line 150 to, say, three by changing line 150 to:

```
150 LET COL = COL + 3
```

It would also be advisable to change line 160 to

```
160 UNTIL COL > 160
```

for making arbitrary changes in line 150 may mean that the value stored under COL is *never* equal to 160, but if we keep adding a fixed value to the number stored under COL, this value will *always* become greater than 160 eventually. This could make the difference between the repetitions not stopping (except when an error eventually occurs in this case) and the program stopping properly.

The way that we have written the group of four DRAW instructions twice in succession in almost identical form seems rather unnecessary. Again, BASIC provides a means of avoiding this, and of passing work from the programmer to the computer. By providing the *subroutine* facility, BASIC allows us to write a set of instructions as a self-contained unit which can then be called from a program as often as it is needed. The BASIC words that are used in this context are GOSUB followed by a line number to call the subroutine to cause the computer to return to the main program and resume at the instruction following the GOSUB. The previous program can now be rewritten, making use of a subroutine starting at line 200 to plot the shape, as:

```
10 HIRES
20 INK 3
30 PAPER 4
40 LET COL=20
50 REPEAT
60 CURSET COL,100,0
70 LET F=1
80 GOSUB 200
90 LET F=0
100 GOSUB 200
110 LET COL = COL + 1
120 UNTIL COL = 160
130 END
200 DRAW 0,-40,F
210 DRAW 40,0,F
220 DRAW 0,40,F
230 DRAW  -40,0,F
240 RETURN
```

We have introduced the END instruction in line 130. This simply marks the end of the program, and it is necessary here to prevent the computer from going on to execute the instructions in the subroutine when it has completed these in the program. Using a subroutine makes the program easier to recall than before, showing more clearly that the repetitive part of the program positions the cursor, plots a shape, unplots it and then changes the position. The effect of the GOSUB and RETURN instructions in this program in communicating between the program and the subroutine is illustrated in Figure 4.1.

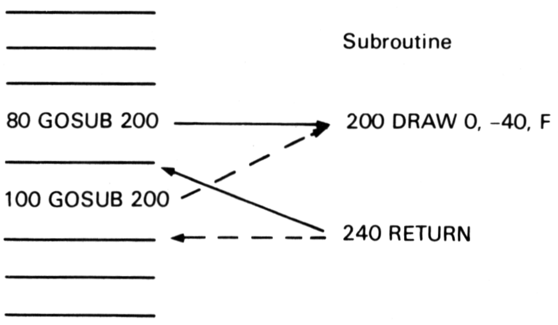


Figure 4.1 Communication between program and subroutines.

Having seen that a subroutine containing graphics commands can be used to draw a shape at a specified position, we can further illustrate the usefulness of the subroutine by creating patterns based on a single shape. We have chosen a hexagon as the shape because it can be used to build some interesting patterns. The following program calls the subroutine starting at line 500 which can draw a hexagon. It is used once to place a hexagon in the centre of the screen.

```
10 HIRES
20 INK 3
30 PAPER 4
40 CURSET 120,100,1
50 GOSUB 500
60 END
500 DRAW 0,–10,1
510 DRAW 9,–5,1
520 DRAW 9,5,1
530 DRAW 0,10,1
540 DRAW  –9,5,1
550 DRAW –9,–5,1
560 RETURN
```

Knowing that the subroutine works, we can now draw a row of hexagons by repeatedly drawing hexagons across the screen. This can be done by:

```
10 HIRES
20 INK 3
30 PAPER 4
40 FOR COL = 1 TO 10
50 CURSET COL*18,100,1
60 GOSUB 500
70 NEXT COL
80 END
500 DRAW 0, – 10,1
510 DRAW 9, – 5,1
520 DRAW 9,5,1
530 DRAW 0,10,1
540 DRAW  – 9,5,1
550 DRAW  – 9, – 5,1
560 RETURN
```

We can now fill the screen with hexagons to give the pattern shown in Figure 4.2 by repeatedly plotting rows of hexagons. The program for this is:

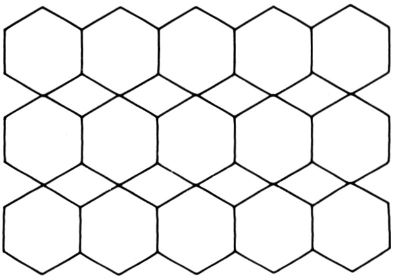


Figure 4.2   A pattern of hexagons.

```
10 HIRES
20 INK 3
30 PAPER 4
40 FOR ROW = 1 TO 9
50 FOR COL = 1 TO 10
60 CURSET COL*18, ROW*20, 1
70 GOSUB 500
80 NEXT COL
90 NEXT ROW
100 END
500 DRAW 0, – 10,1
510 DRAW 9, – 5,1
520 DRAW 9,5,1
530 DRAW 0.10,1
540 DRAW – 9,5,1
550 DRAW – 9, – 5,1
560 RETURN
```

In this pattern there are nine rows each with ten hexagons, and 90 hexagons are drawn, which means that the subroutine has been called 90 times. By calling the subroutine, we have saved ourselves an immence amount of work: first imagine writing out the six DRAW instructions for a hexagon 90 times! The pattern is a very interesting one, and can be seen as alternate rows of hexagons and diamonds or as rows of interlocking larger hexagons. The pattern can be amended quite easily to give the 'honeycomb' pattern shown in Figure 4.3. To create it, we basically need only to displace alternate rows by a small amount. But we must be careful not to tell the Atmos to try to plot off the display area, for if we do this it will give a SYNTAX ERROR. The program for the 'honeycomb' pattern is:

```
10 HIRES
20 INK 3
```

```
30 PAPER 4
40 FOR ROW = 1 TO 9
50 FOR COL = 1 TO 7
60 CURSET COL*18 + ROW*9, ROW*15,1
70 GOSUB 500
80 NEXT COL
90 NEXT ROW
100 END
500 DRAW 0, - 10,1
510 DRAW 9, - 5,1
520 DRAW 9,5,1
530 DRAW 0,10,1
540 DRAW  - 9,5,1
550 DRAW  - 9, - 5, - 1
560 RETURN
```

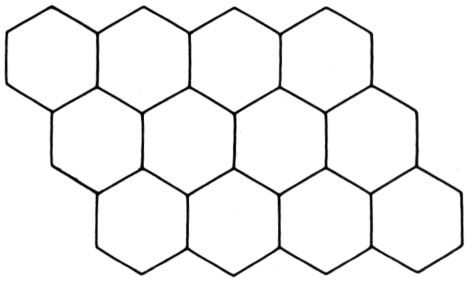


Figure 4.3   A 'honeycomb' pattern.

All these programs follow a very similar pattern, and although we have numbered them all neatly, they can all be obtained by starting from the first program for plotting a single hexagon and inserting or amending lines. But it is the use of the subroutine that makes the developments possible.

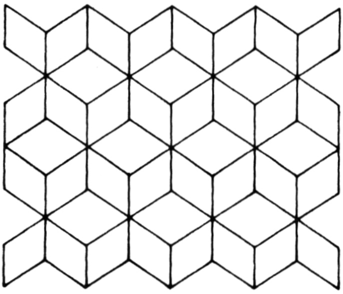
SUMMARY BOX 19

The Subroutine

GOSUB line number
RETURN

1 The instruction GOSUB causes the computer to go to the subroutine starting at the line number 'line number'.

2 The subroutine must end with the instruction RETURN which causes the computer to return to the instruction following GOSUB and to obey that instruction next.



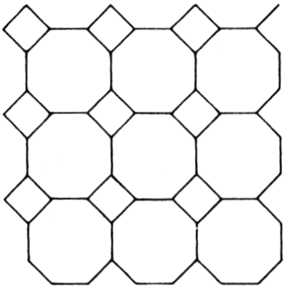


Figure 4.4 Some patterns to draw.

Figure 4.4 shows some patterns that you might care to create on the Atmos. You need to identify the pattern and its constituent shapes, but then the drawing programs will take forms very similar to the ones we have just written.

Still using the same few graphics commands, we can write an 'artist's program' which allows its user to create drawings on the screen. We shall write the program so that it responds to certian keys by drawing a short line in the direction suggested by the positions of the keys. The keys to be used are shown as they are situated on the keyboard in Figure 4.5 and the corresponding directions in which they cause a line to be drawn are shown in Figure 4.6. In this way, pressing the A key, for example, when the program is running will cause a line to be drawn up the screen, and lines can be drawn in directions at any multiple or 45 degrees from this.

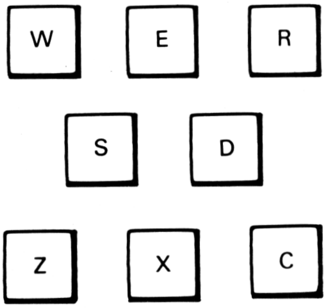


Figure 4.5 The keys used by the artist's drawing program.

We could use an INPUT instruction in the program to detect which key is being pressed, but this would not only temporarily halt the program but would also cause a distraction by displaying its prompt. Instead, we shall introduce a new command, of the kind that is used in many computer games, which simply scans the key-

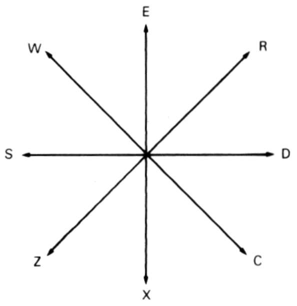


Figure 4.6 The direction associated with the keys in Figure 4.5.

board to see if a key is being pressed at the instant that it is being executed. If a key is being pressed it reports which it is, and if no key is being pressed it reports *that*. The instruction is the KEY$, and when executed it gives the single character that corresponds to the key that is being pressed or, if no key is being pressed, it gives nothing. It can be used as in

```
LET A$ = KEY$
```

to store under the name A$ either the single character or no charcter.

SUMMARY BOX 20

Scanning the Keyboard

KEY$

1 **KEY$ gives the character that corresponds to the key that is being pressed when it is executed. If no key is being pressed, it gives nothing.**

A description of our program is given in the *showchart* of Figure 4.7. Points at which a decision must be made are shown by diamonds: these will correspond to IF – THEN instructions in the program. The program based on this plan is:

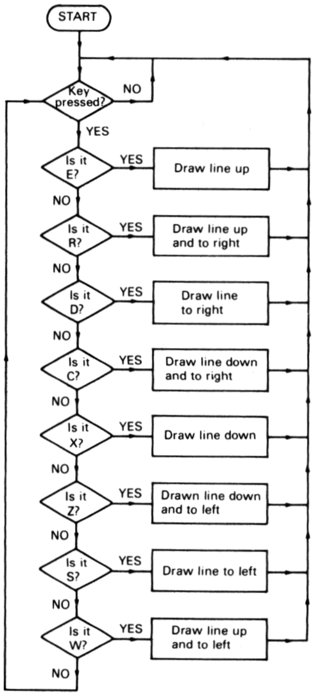


Figure 4.7 Flowchart for artist's drawing program.

```
10 HIRES
20 INK 3
30 PAPER 4
40 CURSET 120,100,0
50 LET A$ = KEY$
60 IF A$ = "E" THEN DRAW 0, – 5,1: GOTO 50
```

```
70 IF A$ = "R" THEN DRAW 4, –4,1: GOTO 50
80 IF A$ = "D" THEN DRAW 5,0,1:  GOTO 50
90 IF A$ = "C" THEN DRAW 4,4,1:  GOTO 50
100 IF A$ = "X" THEN DRAW 0.5,1: GOTO 50
110 IF A$ = "Z" THEN DRAW  –4,4,1: GOTO 50
120 IF A$ = "S" THEN DRAW  –5,0,1: GOTO 50
130 IF A$ = "W" THEN DRAW  –4, –4,1: GOTO 50
140 GOTO 50
```

This program allows any detailed shape to be drawn on the screen, but it produces a drawing as one long trace. This is because we have used 1 as the third number in all the DRAW instructions. But we can modify the program so that is can move the cursor without drawing if we amend the program slightly so that the third number can be changed. We choose the P key, only because it is at the opposite side of the keyboard to all the other control keys, to make the program switch between drawing lines and just moving the cursor when the other control keys are pressed. The amended program is:

```
10 HIRES
20 INK 3
30 PAPER 4
40 CURSET 120,100,0
50 LET F% = 1
60 LET A$ = KEY$
70 IF A$ = "P" THEN LET F% = 1 – F%: GOTO 60
80 IF A$ = "E" THEN DRAW 0, –5,F%: GOTO 60
90 IF A$ = "R" THEN DRAW 4, –4,F%: GOTO 60
100 IF A$ = "D" THEN DRAW 5,0,F%: GOTO 60
110 IF A$ = "C" THEN DRAW 4,4,F%: GOTO 60
120 IF A$ = "X" THEN DRAW 0,5,F%: GOTO 60
130 IF A$ = "Z" THEN DRAW  –4,4,F%: GOTO 60
140 IF A$ = "S" THEN DRAW  –5,0,F%: GOTO 60
```

```
150 IF A$ = "W" THEN DRAW  -4, -4,F%:
    GOTO 60
160 GOTO 60
```

Line 70 causes the program to change its mode of operation when P is pressed by changing the value stored under F% from 1 to 0 or from 0 to 1. The way in which it does this is something of a programming 'trick'. When the value stored under F% is 1, the effect of LET F% = 1 – F% is to store 1 – 1, that is 0, under F. But if the value stored under F% is 0, the effect of this instruction is to store 1 – 0, that is 1, under F%. Since we begin by storing 1 under F% with line 50, each time P is pressed the value stored under F% swtiches between 1 and 0. In turn this affects all the DRAW commands, making them switch between plotting in the foreground colour (visibly) or in the background colour (invisibly). For this reason the program is also able to unplot, or erase, parts of the picture that have already been created.

It should also be noted that in both of the artist's drawing programs given above, we have placed two commands after the THEN in the IF – THEN instructions and that they are separated by a colon. Referring back to the Summary Box for making decisions will remind you that we may place a sequence of commands after 'THEN'. When we do this, the commands must be separated by a colon. In fact, we can place more than one command on any program line as long as a colon is used to separate them. But although this makes the program listing shorter it also makes it more difficult to read and is not recommended until a thorough familiarity with BASIC, and writing programs with it, has been acquired.

Other instructions for graphics

The Atmos provides a number of instructions for graphics apart from those we have met already. They are given, with their purposes, in the following table. In this section, we shall examine a few of them and how thy can be used.

Table 3 **The remaining graphics instructions**

Instructions	Purpose of instruction
CHAR	To place characters on the graphics screen.
CIRCLE	To draw a circle
CURMOV	To move graphics cursor by a specified amount, rather than to a specified position as with CURSET.
FILL	To fill a rectangle with a specified colour.
PATTERN	To allow lines and circles to be drawn in broken lines having a specified pattern as dots and dashes rather than in a continous line.
POINT	To test whether a particular point is plotted or not (that is, whether it is in the foreground or background colours.)

With CIRCLE, a circle is drawn centred on the current cursor position. Two numbers must be given; the first for the radius of the circle and the second, in the same way as the last number with DRAW, to determine the colour in which the circle is plotted. We can draw a circle at the centre of the screen having radius 50 with the three commands.

```
HIRES
CURSET 120,100,0
CIRCLE 50,1
```

The following programe displays a pattern of circles that give the appearance of a cone.

```
10 HIRES
20 INK 3
30 PAPER 4
40 FOR ROW = 1 TO 30
50 CURSET 120, 50 + 3*ROW, 0
60 CIRCLE ROW, 1
70 NEXT ROW
```

A curved horn will result if line 50 is replaced by

```
50 CURSET 20 + 4*ROW, 20 + ROW*ROW/7,0
```

The FILL instruction is used to fill a rectanglar area with colour, and with it all the colours of the Atmos can be displayed on the high-resolution graphics screen. It is used in conjunction with CURSET which establishes the position of the top left corner of the area to be filled. Then FILL itself is followed by three numbers. The first gives the number of rows of dots in the rectangle, and the second gives the width of the rectangle in multiples of six dots. The third number gives the colour, but in such a way that 16 must be added to the numbers used with INK and PAPER. The instruction

```
FILL 50,4,18
```

will cause a rectangle 50 dots high by 24 dots wide to be filled with yellow. By referring to the representation of the graphics screen given in Figure 2.5, we can see that the first number can be anything from 1 to 200, the second from 1 to 40 and, for the colours, the last number varies from 16 to 23.

But there is a snag with the use of FILL. If the area to be filled has not previously been filled with black using colour number 16, the effect of FILL will be to colour not just the area we want, but also its extension to the right hand edge of the screen. The following program, which illustrates the use of FILL by displaying an array of coloured squares, takes care of this difficulty by first filling the entire screen with black.

```
10 HIRES
20 CURSET 0,0,0
30 FILL 200,40,16
40 LET C = 17
50 FOR ROW = 1 TO 6
60 FOR COL = 1 TO 6
70 CURSET 24*ROW, 24*COL,0
80 FILL 24,4,C
90 LET C = C+1
100 IF C> 23 THEN LET C = 17
110 NEXT COL
120 NEXT ROW
```

POINT is used to examine a dot on the screen. It must be given the column and the row of the dot. If the dot is in the foreground colour it gives the value −1, and if it is in the background colour it gives 0. It is very useful in games programs, for example to determine whether a missile strikes a target. We shall illustrate its use with a program that displays a random scattering of potential targets and a missile launcher, and then launches a missile when the F key is pressed. If the missile strikes a target then an explosion results, but if it passes harmlessly off the screen the program simply ends. A flowchart for this program is given in Figure 4.8.

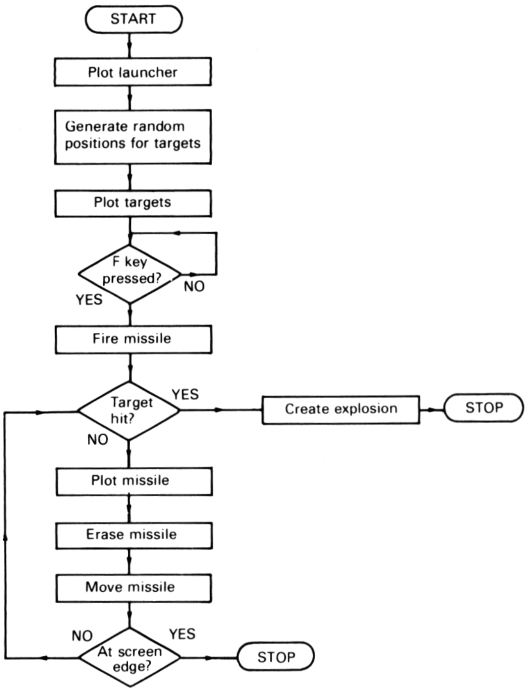


Figure 4.8 Flowchart for missile launcher program.

The program generates the random positions for the targets by using RND. This generates at random a number from 0 up to, but not including, 1. Multiplying it by, say, 200 gives a random number from 0 up to, but not including, 200. For a row number, we need a whole number, that is an interger, since it does not make sense to talk about row ten-and-a-half. BASIC provides us with INT for finding the whole number part of a number and INT (2.5), for example is 2. In this way, we can generate at random a whole number from 0 to 199 for use as a row number by INT(RND(1)*200). In similar fashion, if we only want row numbers from 50 to 189, we can generate them at random with INT(RND(1)*140) + 50. The program for this is:

```
10 HIRES
20 INK 3
30 PAPER 4
40 CURSET 12,100,0
50 DRAW 0, – 10,1
60 DRAW 10,5,1
70 DRAW  – 10,5,1
80 FOR K =  1 TO 6
90 X =  INT(RND(1)*150) + 40
100 Y =  INT(RND(1)*140) + 50
110 CURSET X,Y,0
120 GOSUB 500
130 NEXT K
140 ROW  =  95
150 COL  =  23
160 IF KEY$ = "F" THEN GOTO 180
170 GOTO 160
180 IF POINT (COL, ROW)  =   – 1 THEN GOTO 300
190 CURSET COL, ROW,1
200 CURSET COL, ROW,0
210 COL  =  COL + 1
220 IF COL> 239 THEN STOP
230 GOTO 180
300 CURSET COL, ROW,0
310 FOR X = 1 TO 10
320 CIRCLE 3*X,1
330 NEXT X
340 END
500 DRAW 0, – 50,1
510 DRAW 50,0,1
520 DRAW 0,50,1
530 DRAW  – 50,0,1
540 RETURN
```

The only new instruction in the program is STOP in 220, which when executed causes the computer to stop the execution of the program.

SOUND

The Atmos provides several ready-made commands for making sounds. They are EXPLODE, PING, SHOOT and ZAP. They give the sounds that you might expect from their names and are basically intended for use in games programs. But the Atmos can produce sounds that are a good deal more musical and flexible than this. Its sound generator has three channels, each of which can produce either a tone (a single note) or noise. Any channel or combination of channels can be activated. This means that it can play single notes, two-note and three-note chords,and combinations of notes and noise. The duration and volume of notes can be controlled, as can their envelope, that is, their attack — sustain — release pattern. With these facilities, the Atmos is a considerable music generator able to synthesise a wide range of sounds and the simulate various musical instruments.

The three instructions with which the more complex sounds can be produced are given in the next table.

Table 4 Instructions for generating music

Instruction	Purpose of instruction
MUSIC	To cause one of the Atmos' sound channels to play a given note in a particular octave at a specified volume.
PLAY	To establish which channels are to give tones and which to give noise, and to set the envelope for the sounds.

SOUND **To produce sounds of a given pitch and volume using a specified combination of channels.**

We will present a few programs using the MUSIC instruction. They rely on the fact that channel 1 is always available for porducing sound so that the complications of turning on the various channels using PLAY and SOUND can be avoided. Because PLAY and SOUND are so powerful they are quite complex and, in fact, their capabilities are probably more easily understood by a little experimentation, listening to the resulting sounds, than they are from written descriptions.

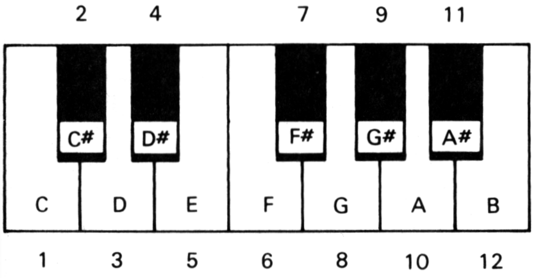


Figure 4.9 The notes of one octave on a piano keyboard and their values for use with MUSIC.

When the MUSIC command is used, it must be followed by four numbers. These give, in this order, a sound channel, an octave, a note in the octave and a volume. We shall only use 1 for the channel number. The range of sounds that the Atmos can generate covers seven octaves: the octave numbers are 0 to 6. The octave from middle C to the B above middle C has the number 2. Higher octaves have higher numbers and lower octaves lower numbers. The notes in an octave take the values from 1 to 12, with C given the value 1 and so on up the scale as represented with reference to the piano keys in Figure 4.9. For the volume, values from 1 to 15 are used, with 1 giving the softest sounds and 15 the loudest.

The programs presented below take, perhaps surprisingly, the same forms as the graphics programs given earlier. First, we can write a program to play a short sequence of notes, all of which are in the same octave, getting:

```
10 READ M
20 FOR K = 1 TO M
30 READ N
40 MUSIC 1,2,N,9
50 WAIT 50
60 NEXT K
70 PLAY 0,0,0,0
80 DATA 7,1,3,5,6,8,6,5
```

The WAIT instruction at line 50 causes the computer to wait for 50 hundredths of a second (that is, half a second) before moving to the next instruction. This is because the sounds are produced only until the next sound is instigated, and if we want the sounds to be played for long enough for us to hear them we must make the computer pause before it generates the succeeding sounds. The PLAY instruction in line 70 is necessary to turn the last note off, for otherwise it will continue to sound forever.

The program plays the first notes of a well-known tune, but there is more chance of recognising the tune if the notes have the correct duration. We can amend the program to accommodate this by reading the duration of each note with the note itself. This gives us the program:

```
10 READ M
20 FOR K = 1 to M
30 READ N,D
40 MUSIC 1,2,N,9
50 WAIT D
60 NEXT K
70 PLAY 0,0,0,0
80 DATA 7,1,50,3,100,5,50,6,100,8,100,6,100,5,50
```

If we want to play music with notes that come from more than one octave, we can read the octave number, too, as in:

```
10 READ M
20 FOR K = 1 TO M
30 READ OC,N,D
40 MUSIC 1,OC,N,9
50 WAIT D
60 NEXT K
70 PLAY 0,0,0,0
80 DATA 5,2,1,50,2,3,50,1,11,50,1,6,50,2,6,100
```

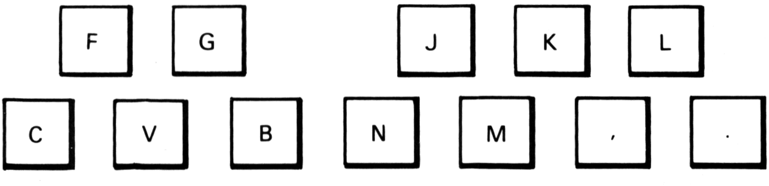


Figure 4.10 The keys on the Atmos' keyboard used by the music-playing program.

We can convert a part of the Atmos' keyboard to a stand-in for a piano keyboard so that we can play tunes on the computer by entering an appropirate program. We have chosen keys from the bottom two rows of the keyboard as shown in Figure 4.10 to play the notes of one octave. Referring to Figure 4.9 will show that the keys used start at C and have positions that correspond to those of the keys on a piano. The program will have the same 'shape' as the artist's drawing program, the flowchart for which is shown in Figure 4.7. The program is:

```
10 LET A$ = KEY$
20 IF A$ = "C" THEN MUSIC 1,2,1,9: GOTO 150
30 IF A$ = "F" THEN MUSIC 1,2,2,9: GOTO 150
40 IF A$ = "V" THEN MUSIC 1,2,3,9: GOTO 150
50 IF A$ = "G" THEN MUSIC 1,2,4,9: GOTO 150
60 IF A$ = "B" THEN MUSIC 1,2,5,9: GOTO 150
70 IF A$ = "N" THEN MUSIC 1,2,6,9: GOTO 150
80 IF A$ = "J" THEN MUSIC 1,2,7,9: GOTO 150
90 IF A$ = "M" THEN MUSIC 1,2,8,9: GOTO 150
100 IF A$ = "K" THEN MUSIC 1,2,9,9: GOTO 150
110 IF A$ = "," THEN MUSIC 1,2,10,9: GOTO 150
120 IF A$ = "L" THEN MUSIC 1,2,11,9: GOTO 150
130 IF A$ = "." THEN MUSIC 1,2,12,9: GOTO 150
140 GOTO 10
150 WAIT 25
160 PLAY 0,0,0,0
170 GOTO 10
```

The sensible use of CTRL and A will save a good deal of time when entering this program.

Summary

The Atmos has considerable facilities for graphics and sound. These are explained and the Atmos' capabilities in these areas are explored in writing programs that incorporate them. While these programs are being developed, the opportunity its taken to introduce more of the features of BASIC as they are needed. In particular, the subroutine is introduced. This is something that is particularly important when writing lengthy programs as it allows them to be devided into parts which can be written separately and which can impose some structure on the programs. With the graphics instructions that have been introduced for creating shapes and colouring areas it is possible to creat colour displays to almost any specification.

The instructions for creating sounds are also explained. There are fewer of them than for graphics, and some knowledge of music theory is necessary to appreciate to the full the ways in which they can be used. Programs that use these instructions are presented for playing music. Interestingly, they call on program structures which are the same as those used by the graphics programs.

Chapter 5

Applications for the Atmos

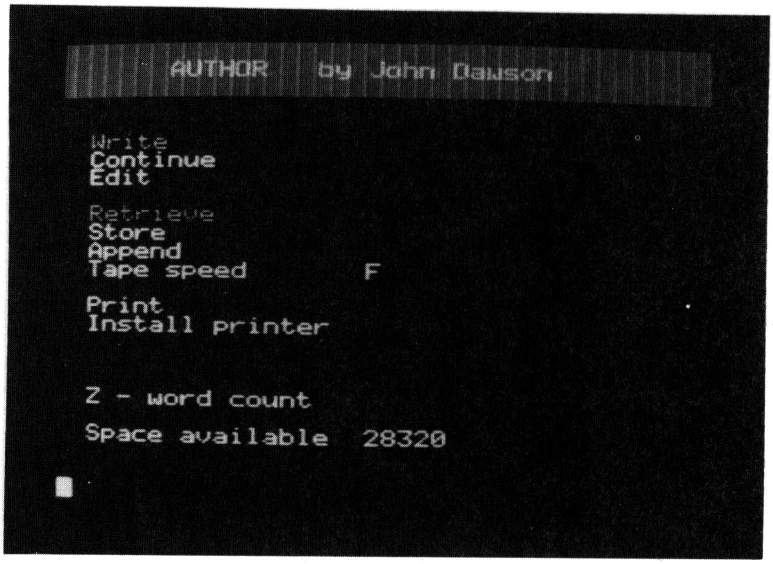
There are plenty of programs available for the Atmos that are ready to run and to make the computer do something useful for you. The most notable supplier of programs for the Atmos is Tansoft. A lot of the programs that can be bought are for games, but in this chapter we are more concerned with programs that are for profit than those that are for pleasure. Tansoft supplies a program for word processing called Author, a database program, called Oricbase, and a spreadsheet program called Oric-calc. (See Figure 5.1). With a word processing program the Atmos can be used to produce documents, a database program allows it to store information so that any item or items can be retrieved with ease, even when there is a great deal stored, and a spreadsheet is for planning or, indeed, any application where information is presented in tabular form. In this chapter we shall look at these three applications in general, examine the uses to which they can be put and explain how the Tansoft programs can be used to these ends.

Word processing and 'Author'

When 'Author' is loaded into the computer from its cassette, it runs automatically. As soon as 'Author' is running the Atmos is converted to a word processor. When Author starts to run it first clears the screen, then displays a copyright message and after a short pause gives the display shown in Figure 5.2.

Figure 5.1 The word processing, database and spreadsheet packages.

At this stage the word processor is ready to use, and the user familiar with word processing will probably have little difficulty in proceeding to explore Author's capabilities. But we shall take time out to explain what word processing is all about before giving a general explanation of how to use Author. We shall also explain many of the special terms that are used in word processing. This will enable those who are either new to word



```
     AUTHOR   by John Dawson

  Write
  Continue
  Edit

  Retrieve
  Store
  Append
  Tape speed        F

  Print
  Install printer


  Z - word count

  Space available  28320
```

Figure 5.2 The initial menu of Author.

processing or with a limited experience of it to come to Author with a general appreciation of what it can do, and with a knowledge to the relevant vocabulary, to guide their expectations of how to use Author to advantage. It is worth reiterating the point that the act of loading a word processing program has converted the Atmos from a general-purpose computer to a specialist word processor.

Word processing in general

Word processing is making use of the capabilities of a computer to store, process and print words. The computer has to do this in a way that satisfies any writer in applications ranging from writing a letter to creating a fair sized document to writing a book. Although the computer itself will still be performing much the same basic operations as when it handles numbers or creates graphics, loading a word processing program into it gives it the ability to recognise and handle words, sentences, paragraphs and pages in any text that it is given. It can then handle any of these as distinct units.

The computer does not understand the text that it is given in the way that a person would, and it would be overestimating the ability of a word processing program to think that it gave the computer this power. Such a misunderstanding could also lead to expecting more from a word processing program than it could ever deliver. A word processor simply provides the computer with some rules that allow it to recognise words, sentences and the other grammatical units. One rule, as an example, is that a number of consecutive letters followed by a space constitute a word, with the space indicating the end of the word. A second rule is that several words followed by a full stop make up a sentence, although the word processor must also be aware that a question mark and an exclamation mark can terminate a sentence as well as a full stop.

A word processing program allows its users to type at the keyboard of a computer in just the same way as a typist does at a typewriter. The text that is typed is automatically stored in the computer's memory at the same time as it is displayed on the computer's screen. Because the text has been stored it is possible for the computer to process or manipulate it in any of a variety of ways. The particular ways that are available depend on the word processor that is used, but there are certain operations that are common to almost all of them.

A word processor automatically arranges for the text that is typed to be neatly laid out, both on its screen and when it is printed. The precise form of the layout can be prescribed by the user. Among the factors affecting the layout that the user can select, or change, are the length of the lines of text, the positions of the right and left margins, and the presentation at the beginning of a new paragraph. Many word processors display the text on their screens in the prescribed way as the text is typed. Later, when it is printed, using a printer attached to the computer, it will appear in exactly the same form. This means that the document being produced can be checked to ensure that it is perfect while it is on the computer's screen and before it is committed to paper. It can then be printed for the first time with the confidence of knowing that it is correct. To print the document, all that is necessary is to give the appropriate command to the word processor.

With other word processors, the text is not displayed exactly as it will be printed. This may be because the number of characters that can be displayed on one line of the screen is less that the number that is printed on a line on paper. The Atmos' screen, for example, displays a maximum of 40 characters to a line, whereas the number printed on a line on paper is usually at least double this amount. Also, again as with Author, special characters, such as those to mark the end of a paragraph, may be shown on the screen as special symbols although they will not be printed on paper when the document is printed: they will give a particular effect — in this case to cause a new paragraph to begin. In the end it does not really matter how a word processor displays its text, because the user will adapt to the way that it works and learn to take advantage of all its facilites.

If the user finds mistakes in the text when reading it on the screen, it is obvious that he or she must be able to correct them. Word processors allow corrections to be made to the text that has been entered by deleting, replacing or inserting letters so that simple errors such as spelling mistakes can be corrected. But they also allow whole stocks of text, such as a paragraph, to be inserted, deleted or even to be moved from one place in a document to another.

It is evident that text must always be typed, to enter it, before it can be handled by a word processor. The better the user is at typing, the faster documents can be entered and the more use will be made of the word processor. These seemingly obvious points are made to bring out the importance of the Atmos having a 'proper' keyboard, unlike the Oric-1. Without it, touch typing would be impossible. As a second matter, although we are accustomed to thinking of documents as being printed on paper, when using a word processor there may be no need to use paper at all. Because the Atmos can communicate via its EXPANSION socket a document can be passed from one computer to another in electronic form. Documents can equally well be exchanged after recording them on cassettes. Either way, documents can be passed to another computer for storage and display so that their contents can be communicated and read without committing them to paper at all.

By these means, the Atmos together with Author provides the ability to produce perfectly correct documents and the means to communicate them, perhaps as electronic mail, without the need for paper.

Who needs a word processor?

Word processing makes the production of documents easier in various ways. It is interesting, and informative, to consider who can benefit by using a word processor and to examine the ways in which they can use a word processor to advantage.

The rapid and simple production of perfect documents is one major advantage of word processing that benefits all its users. All the corrections and amendments to a document that are needed can be made before it is ever printed. Word processors can also produce refinements that enhance the appearance of a document such as underlining words, placing headings centrally on a line, and placing the columns of a table in neat alignment. A letter can be arranged after it has been typed at the word processor, for example, to make it fit onto one page rather than running on to a second page that contains just one line to be followed by a signature. The word processor can make the rearrangement automatically if it is told to make the length of the lines in the document a little greater than they were previously. Anyone writing letters can benefit from this, but there are further benefits that can help many other groups of users, including students writing essays, authors writing books and businessmen producing reports and publicity releases.

A second very real benefit can be obtained by anyone producing quantities of letters that may all be slightly different, but which consist in the main of standard paragraphs drawn from a fixed repertoire. With the standard paragraphs stored by the word processor, any of these letters can be produced simply by calling up the necessary paragraphs in the correct order and making a few insertions, such as the recipient's name, to personalise it. There is no longer any need to type each letter individually. The insertions can be made using the editing facilities of the word processor. Documents of this kind have to be produced routinely by, for example, lawyers, estate agents and insurance salesman.

The authors of magazine articles and books, and students who have to write essays, are among those familiar with the task of creating a polished text after producing several draft versions. This process involves writing a first draft, crossing out parts of it, inserting others perhaps by cutting and pasting bits of paper, and then rewriting the whole if it becomes too untidy or unreadable. This gives a second draft on which the whole process may be repeated to give a further draft, and so on. This can be very time-consuming, and if each draft is typed afresh, there is a danger of introducing errors into the parts that are already satisfactory. With a word processor, amending, revising and polishing a draft are all much easier. There is never any need to retype an entire document as the word processor's editing facilities are sufficient to allow one version to be converted to the next. The new version is always displayed at once. In this way, the current version of the document is always available and perfectly legible, but previous versions can be saved in case the editing is not done properly or its results are not satisfactory.

It will take anyone who is accustomed to working with hand-written drafts a little time to become accustomed to using a word processor instead. But the almost universal experience of those who have made the transition is that to use a word processor is a much more rapid and convenient way of creating a finished document.

A word processing session

We now describe a typical word processing session with a view to illustrating how it proceeds in practice. At the same time, we shall introduce some of the terminology of word processing.

On the assumption that you have some text to turn into a document, once your word processor is ready to accept text you can sit down and start to type it. The words that you type will appear on the screen as you type them. They are also being stored in the computer's memory. Nothing unexpected happens until a line on the screen is filled. Then the first word that makes the line too long to fit on the screen is automatically placed at the beginning of the next line. This is known as *word wrap*. It is worth stressing that there is no need to press RETURN at the end of each line as the equivalent of a carriage return on a typewriter: in fact, it is wrong to do so. The word processor manages the creation of the lines itself. At the same time as the new line is being started, some word processors (but not Author) can adjust the previous line so that its final word finishes exactly at the end of the line. This is done by inserting extra spaces to push the end of the line across to the required position, and it gives the document a neat vertical margin at the right of the page as well as at the left. The process is known as *justification.*

Continuing to type gives successive lines, all of which are treated in the same way. When the end of a paragraph is reached, RETURN should be pressed at the end of its final sentence. This causes the word processor automatically to create the gap between paragraphs and to indent the beginning of the next paragraph. This is part of the *formatting* of every document that is carried out by a word processor, as are the positioning of the right and left margins and the line spacing. At all times the document is displayed on the screen (although not necessarily exactly in the form which it will be printed, as we have already explained) so that the user can see what he has typed.

The word processor provides a particular format for the documents it produces by default. If the user prefers a different format, any aspect of it can be changed after giving the appropriate *command* to the word processor.

When sufficient text has been entered, the word processor may indicate that the end of a page has been reached and start on a new one. The pages of the final document can be numbered: they can also be given a *header* , that is, a line of text to be placed at the top of each page as in the running title seen at the top of the page in many book. Sometimes it is also possible to give each page a *footer* , which is the same as a header except that it appears at the bottom of each page.

If words can be underlined or emboldened, to emphasise them, this is usually done in one of two ways. Either a special symbol is typed before and after the words in question, or a special mode is entered before the words are typed and is left afterwards.

On reaching the end of a document, or even after typing a certain amount of it, you will want to go back and check it to ensure that everything has been typed correctly, to see that there are no spelling mistakes, and that everything is to your liking. The process of correcting a document is known as *editing*, and before you can do it you must give the editing command. Editing is one area where word processin greally comes into its own and shows to considerable advantage over using a typewriter.

If your document can be displayed on the screen in its entirety during editing, then you first move the cursor to the position on the screen where a change is needed by using the cursor movement keys. Then letters can be deleted, inserted or replaced at this position by pressing the key that initiates the necessary action. Whole sentences and paragraphs can be deleted or moved just as easily. A display at the top of the screen provides a reminder of the letters that can be used during editing for all the different purposes. As soon as any editing operation is completed, the word processor reformats the text to take account of the changes that have been made.

If the document is too large to fit on the screen, the screen acts as a window through which a part of the document can be seen. Pressing the appropriate keys causes the screen window to move up and down the document or, looking at it in another way, causes the document to *scroll* up and down beneath the screen. Since a part of a document can only be edited if it is being displayed, a large document is edited by first bringing the part to be changed into the display area and then proceding in the way just described. The position of the cursor along a line is usually indicated or a *ruler* line, which is a line that appears above the displayed text that also shows that character positions on a line, the positions of the margins and the tab stops.

When you have finished editing a document, you will want to save it or to print it, or even to do both. Either is done by giving the appropriate command. This is done by returning to the main command *menu*, or list of commands, and issuing one of the commands displayed there. A document can be saved on cassette or disc depending on whether you have a cassette player or disc drive attached to the computer for storage purposes. The document can be printed as long as a printer is attached to the computer.

There are several other commands that can be given to Author or, indeed, to any other word processor. All of the commands for Author are explained later on. You will probably find with any word processor that you do not need all the commands that are provided. The ones mentioned already and a few others that satisfy your particular needs will usually be sufficient. But it is a good idea to be aware of all the commands that your word processor possesses, for if you are not you may not know that your word processor is perfectly capable of doing something that you need, especially when a new requirement emerges.

A glossary of word processing terms

This section provides a brief summary of the key terms encountered in word processing. Most of the terms were introduced in the previous section, but some others have been added for completeness. Not all the terms relate to features provided by Author, but then Author is not the only word processor available for the Atmos, and it may be that you need facilities that Author does not have. In the next section we shall look at Author in some detail.

Back-up. A spare copy of a document that is recorded in case anything untoward should happen to the current document or if the results of editing the document prove unsatisfactory.

Centring. Automatically placing a heading or a line of text symmetrically in the centre of a line.

Command. Issued when in a special command mode, it tells the word processor to carry out one of the functions of which it is capable.

Deleting. Removing letters or words from the text.

File. The form in which a document is stored on a cassette or disc.

Footer. A fixed line of text appearing at the bottom of each page of a document.

Format. The way in which the text of a document is to be arranged by the word processor.

Header. A fixed line of text appearing at the top of each page of a document.

Help facility. Provides information on how to use the word processor and its commands so that assistance is always immediately available and there is no need to refer to a manual.

Insertion. The ability to insert text at any point in a document.

Justification. Justified text is arranged evenly at both the left and the right margins.

Line spacing. The vertical spacing between lines. Typing is usually double spaced.

Marker. A mark placed in the text which will not show when the document is printed and which serves to identify a block of text so that it can be moved, for example.

Menu. A list of items, such as commands, from which one may be selected.

Page break. Indicates to a printer that it must start to print on a new sheet of paper.

Ruler. A line usually placed above the displayed text on which margin and tab settings are shown, and with the aid of which they can be changed. It also shows the position of the cursor along a line.

Search and replace. A useful editing facility with the aid of which the word processor can search for occurrences of a specified word or phrase and, optionally replace them by another one.

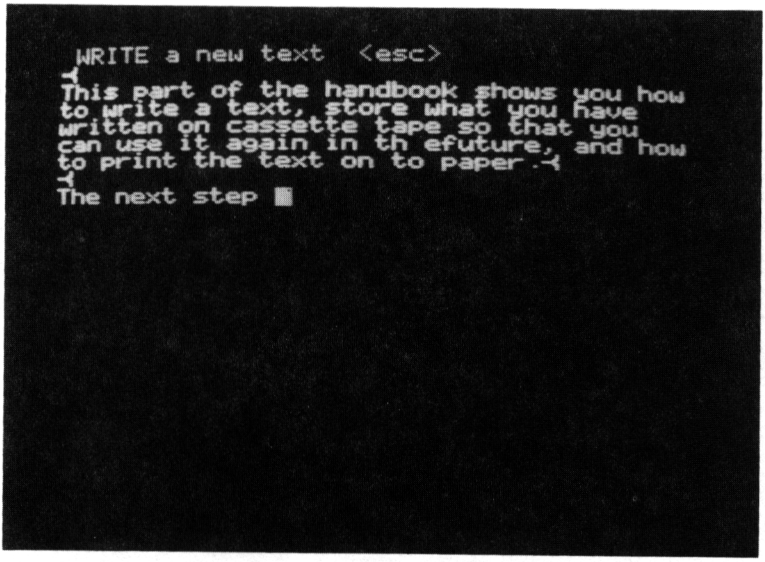
Tabulation. Moving along a line to a fixed position (the tab stop). It is useful for setting out tables or in any work where vertical alignment is needed.

Word wrap. Placing words that will not fit at the end of one line at the beginning of the next one.

Word processing with Author

We have already illustrated the display that Author produces as soon as it is loaded in Figure 5.2. This is its command menu. All the commands are listed in this menu and they can only be issued when this display is being shown. So to write a document or to edit, store or print it we must first return to this display and then issue the appropriate command, which is done by keying the first letter of the command.

By pressing W, then, we give the Write command and can start to type text into Author exactly as we have already described. The line at the top of the screen shows 'WRITE a new text' to remind us that this is the state the word processor is in. It also contains < esc> to remind us that by pressing the ESC key we can return to the command menu to issue another command. After any command, AUTHOR always shows us that command and the keys that then have special uses with it on the top line of the screen. Figure 5.3 shows the screen after some text has been entered.



```
WRITE a new text  <esc>
┤
This part of the handbook shows you how
to write a text, store what you have
written on cassette tape so that you
can use it again in th efuture, and how
to print the text on to paper.┤
┤
The next step █
```

Figure 5.3 Entering text with Author.

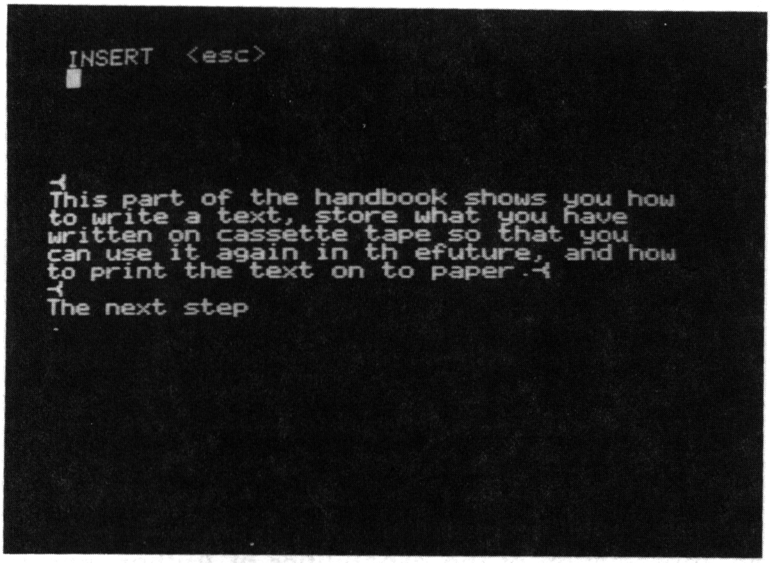
If a typing error is noticed as soon as it is made it can be erased with the DEL key, as can any text on the current line. But it is not possible to delete anything further back than the current line. That is as far as DEL can go. Other mistakes must be corrected later after giving the Edit command.

When your text has all been typed, you will probably want to edit it. Pressing ESC brings back the command menu from where the Editing command is given by pressing E. Pressing a single key at this stage will select one of the editing functions. Some of the keys and their functions are shown in Table 5. Before pressing one of these, we must make sure that the position at which we want to do the editing is displayed on the screen. If it is not, the screen must be scrolled using the 8,9 and Ø keys to bring it into view. Then the cursor movement keys can be used to place the cursor at the position where the editing is to occur and, finally, we can press a key to select the editing function that we want. Some of these keys and their purposes are given in the following table. As soon as one of them is pressed, the top line of the screen changes to show the function selected. Figure 5.4 shows the screen after I (for insert) has been pressed. Notice that a gap has automatically appeared in the text to make room for the insertion.

Table 5 Some keys for selecting editing functions

Key	Function
DEL	Deletion. Pressing DEL again deletes a character, pressing the space bar deletes a word, pressing the full stop key deletes a sentence and pressing RETURN deletes a paragraph.
I	Insertion. Text is inserted in the gap that opens up after the cursor just by typing it.

M	To move a block of text. After the block to be moved has been identified, a 'T' command must be given to show where it is to be moved to.
O	For Overwriting, so that any text that is typed subsequently replaces the text that it overwrites.
F	To find a given phrase or word in the text and, optionally, to replace it with another.



```
INSERT  <esc>


This part of the handbook shows you how
to write a text, store what you have
written on cassette tape so that you
can use it again in th efuture, and how
to print the text on to paper.

The next step
```

Figure 5.4 Author's screen after selecting the editing function 'insert'.

After the text has been corrected, it should be saved. After pressing ESC to end the editing session and return to the command menu, the text can be saved on cassette by giving the S (for Store) command. Author returns to its command menu when all the text is stored. The document can then be printed by pressing P (for Print).

This covers the most important commands that can be given to Author. The others are listed below.

Continue. This finds the end of the text stored in the word processor and places the cursor there so that you can continue to enter text, thereby extending the document.

Retrieve. To retrieve text from tape and load it into the computer.

Append. To append text stored on tape to the text already held in the computer.

Tape speed. To change the speed at which text is transferred between computer and tape from fast to slow or vice versa.

Install printer. To change the format in which a document in printed by a printer.

Z — word count. To give a count of the words in a document.

Summary

This section provides a general introduction to word processing and a specific introduction to word processing with Author. Although there is a good deal more to word processing than has been covered here, it should provide sufficient information for the reader to acquire an appreciation of the capabilities of Author, and to proceed with confidence. A really detailed account of Author is, of course, provided by the manual that is supplied with it.

Databases and 'Oricbase'

A database is an organised and integrated collection of data. It is also rather more than this, for a collection of data has no value unless we can make some use of it. This means that the data in a database must be organised in such a way as to allow us to do the things that we want with it. Operations typical of those that we might want to carry out with a collection of data are to select particular items of data from it according to some criterion, to search it for all the items that meet a given condition, to update items, and to sort the items into some special order. If a database if organised in this way, then from a collection of raw data we can extract useful, and even valuable information.

The data in a database is organised, essentially, by ensuring that all the items are stored in the same structured fashion. If possible, this should take advantage of any relationships between different items of data. A database becomes integrated by ensuring that all items are stored in the same way and also by avoiding any duplication of items. This allows access to any item of data in the database in a natural way by using its relations to other items.

Anyone can benefit from a database program, for we all have the need to store information and to retrieve it, whether in keeping records, running a business or looking after our finances. Typical personal uses are to keep the details of a record collection, a stamp collection or a collection of programs for the computer. Keeping records of all the financial transactions relating to one's annual tax return would also be possible. For educational purposes it could be used to keep an account of all the work that is due to be handed in, and of all the topics taught on a particular syllabus. In a small business, or for that matter a large one, it could be used for stock control by recording the quantities of each item held in stock and updating them as necessary.

The ways in which the items in the database may need to be accessed will be broadly the same in all these cases, although in some instances particular operations may be needed more frequently than others. When using a database to hold the details of a record collection, a typical requirement might be to display all the details of a particular record, or to find the shelf location of a particular record. When used for stock control, the updating of the quantities held will be a common requirement, but if the levels at which goods should be re-orderd are also held in the database, then re-ordering can take place without fail if a stock level can be compared with the re-order level. A businessman who keeps the names and addresses of his customers in a database can print this information on envelopes to address them automatically. If other information is recorded about each customer, then it is posssible to select a particular group. By including their ages, all those under 30 can be selected for a special offer. By including the balance of their account it could be ensured that no more goods were supplied to any customers in the red.

From this discussion we can see that a database program has two distinct parts. The first must allow the data to be entered. The second must allow the user to examine and retrieve the data in any way that may be required. In the entry phase the database program should allow its users to structure their data in a way that is suited to the application. For the second phase the program should supply the facilities to enable the user to carry out all the necessary operations on the stored data. These operations include selection, searching and sorting as we have seen and commands for these activities should be provided.

As the final point in this section, we can illustrate how a database gives considerable advantages over a collection of information recorded in a conventional fashion. Almost every house in the country contains a database, although it is printed on paper, in the form of the details of the week's television and radioprogrammes. This is a database in the sense that it is an organised and integrated collection of data about radio and television programmes. From the printed record it is easy to find the programme that will be broadcast at a particular time, not quite so easy to find at what time a given programme will start, and less easy still to find all the programmes of a particular kind that will be broadcast during the week and when they begin. But if the programme information were all stored in a database program, rather than being printed on paper, each of these selections would be equally easy to find. By giving the appropriate command to the database program it would retrieve the required information straight away.

Filing cards and a database

We begin by examining the way that information is recorded on filing cards and then recovered from a card box full of filing cards. This is because the way that a database program is used is entirely analagous to using a filing card system. Writing the information on the cards corresponds to storing the data in the database program. Retrieving information by examining the cards corresponds to recovering information from the database program. The idea is that by using the database program on a computer, the recovery of the information is made easier and quicker, particularly if there is a great deal of it.

To deal with a concrete example, suppose that we decide to create a file containing the week's television programmes, with the details of each individual programme recorded on a file card. We must first decide how to record the information about a programme on a file card. The details that we are likely to want are its title, its type, the day it is shown, the starting time and the channel on which it is being shown. We can now design a file card as shown in Figure 5.5. When we have filled in a card in this way for every programme, we shall have a box full of cards that records the week's programmes.

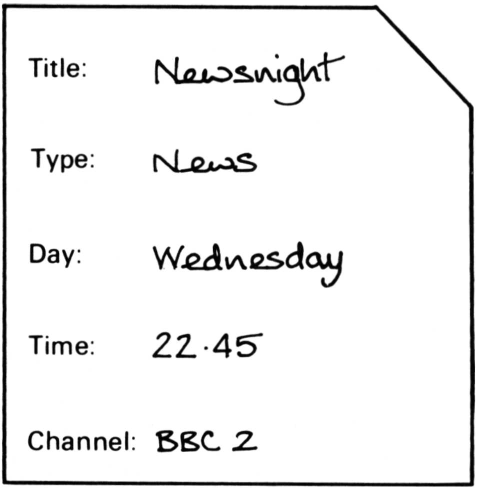


Figure 5.5 A file card.

The information to be stored has been structured by deciding carefully which details to record. It is clear that other items can be placed on each card just as easily as the ones we have chosen, and that an item on each card can be crossed out if we are no longer interested in it. But the time spent on getting the design for the cards right in the first place is well spent. It is also obvious that a file card on which the details of a stamp in a stamp collection were recorded would not be stored in the same box as the cards containing television programme details. A separate box would be kept for these so that it could be filled with similar cards for each stamp and eventually record the entire stamp collection.

When a card for every television programme has been filled in and our box for them is full we have completed the data entry stage and completed our database. Now it can be examined to recover any information we want about the week's television programmes. We can find what is being shown at 8 o'clock on Tuesday evening by flicking through all the cards and noting those that give a match to our requirements under DAY and TIME. We shall find more than one card like this because there is more than one channel broadcasting programmes. We can find the time at which the news is shown by flicking through the cards to find 'news' under TYPE and then reading the entry under TIME. We can find what is on now by looking for an entry under TIME that matches the time now and reading the entry under TITLE. We can find how many sports programmes there are by counting the cards with 'short' under TYPE. We could even decide to sort the cards into an order that suited us, perhaps with the programme titles in alphabetical order.

These operations, and particularly the sorting, will take a considerable length of time. At the least, we must rifle through every card, while re-ordering a large pack of cards to sort them into a new order could be a very lengthy affair. Although a computer database will operate in essentially the same way when it carries out the same operations, the computer's speed of operation ensures that everything is done much more quickly. In addition, once the data is entered, it is the computer that does all the work rather than us.

When using a database program, it first allows us to design, as the equivalent of a file card, a *record*. After this, the details of each record can be entered as the equivalent of filling in file cards. Each item of information within a record is known as a *field*. It corresponds to a single entry on a file card. When a record is entered, it is stored in the computer's memory. A collection of records stored in this way is called a *file*. A file is equivalent to a box of completed cards. A file can be stored permanently on cassette or disc. A database program can be used to create any number of files, so that we could create one file for television programmes, another for stamps and as many others as we needed.

Using Oricbase

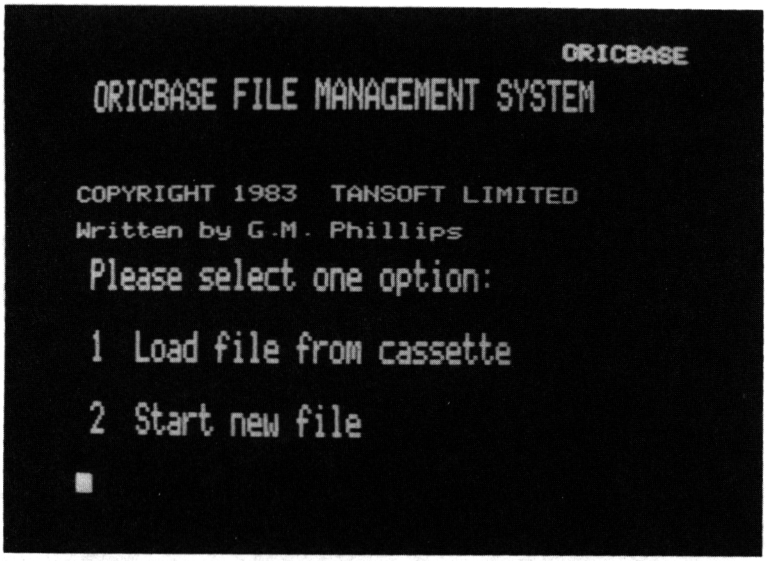
Loading Oricbase from its cassette gives us the initial display shown in Figure 5.6. This invites us to load data previously stored on cassette or to enter data from scratch. As we are just starting, we shall press 2 to select the latter. Having done this, we have to tell the computer how many records we shall put in the file we are creating. Unless we know how many, and we probably do not, a guess that is on the high side is better than a low one. So enter a high number, perhaps a hundred, giving

How many records? 100

Now we must say how many fields each record has. We shall use our television programme example, and so there are five:

How many fields? 5

Next we are asked how many macros we want. Using a macro, we can save the commands that are needed frequently to examine the database to save retyping them. The property of a macro is that a long command can be saved under a name and subsequently, when we type the name it is replaced by the command saved under



```
                                    ORICBASE
    ORICBASE FILE MANAGEMENT SYSTEM


   COPYRIGHT 1983  TANSOFT LIMITED
   Written by G.M. Phillips
    Please select one option:

    1 Load file from cassette

    2 Start new file
   ■
```

Figure 5.6 The initial display of Oricbase.

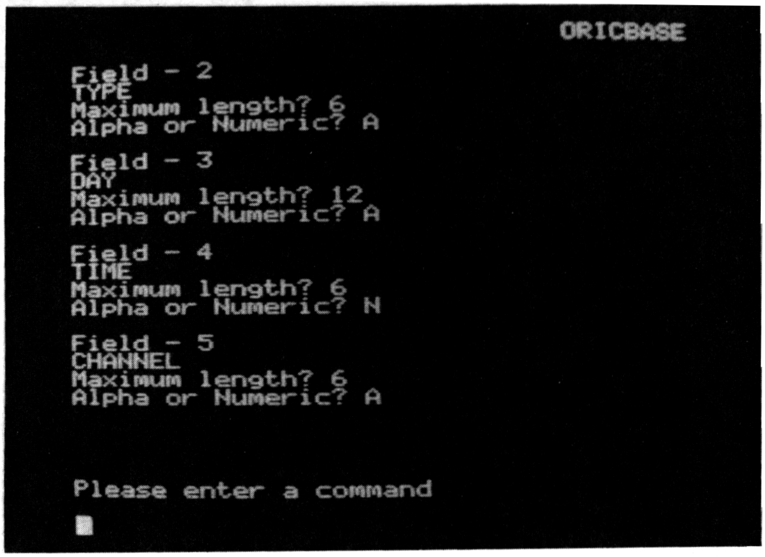
that name. To keep Oricbase happy, we will claim to want five macros.

How many macros? 5

The details of all the fields in a record must be entered next. For each, we must give its name, the maximum number of characters that it will ever contain, and whether the characters are alphanumeric or numeric. The names we shall enter are shown in Figure 5.5. For the maximum number of characters in a field it is advisable to enter something rather more than the largest number you will ever type, as this aids the presentation of records. If a field is to contain just a number, then respond to 'alpha or numeric?' with numeric (which can be abbreviated to N). Otherwise enter alpha (A). For our first field, we could enter

```
Field – 1
TITLE
Maximum length? 12
Alpha or Numeric? A
```

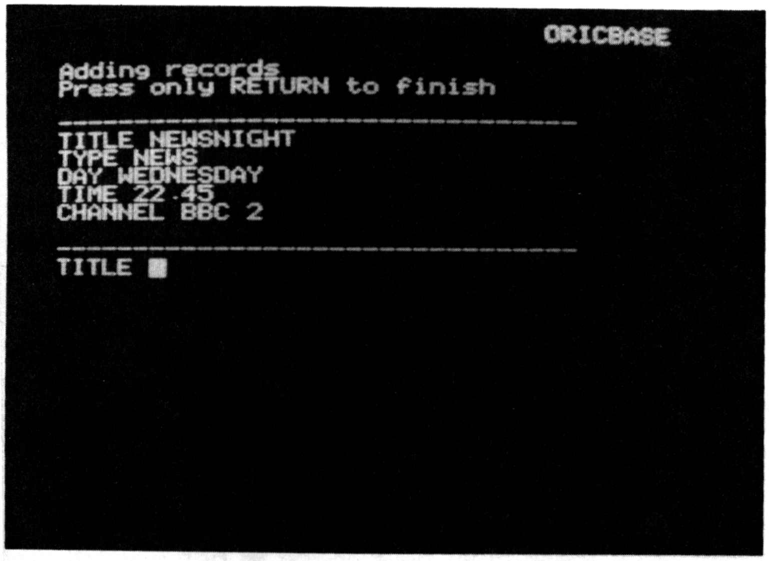
When all five fields have been described, the screen appears as in Figure 5.7, and Oricbase is inviting us to give it a command.



```
                                    ORICBASE

   Field - 2
   TYPE
   Maximum length? 6
   Alpha or Numeric? A

   Field - 3
   DAY
   Maximum length? 12
   Alpha or Numeric? A

   Field - 4
   TIME
   Maximum length? 6
   Alpha or Numeric? N

   Field - 5
   CHANNEL
   Maximum length? 6
   Alpha or Numeric? A


   Please enter a command
   █
```

Figure 5.7 Oricbase field descriptions.

At this stage the only thing to do is to put some records in the database. The command for this is ADD. In response to this command the equivalent of a blank file card is displayed with the cursor flashing against the first field name. The entry against this field is made by typing it and pressing RETURN. The cursor then moves alongside the next field name. When the data of Figure 5.5 is entered, the screen appears as shown in Figure 5.8. Further records can be entered continuously. Their entry can be terminated at any time by pressing RETURN before typing any other character.



```
                                   ORICBASE

   Adding records
   Press only RETURN to finish

   ------------------------------------
   TITLE NEWSNIGHT
   TYPE NEWS
   DAY WEDNESDAY
   TIME 22.45
   CHANNEL BBC 2

   ------------------------------------
   TITLE █
```

Figure 5.8 A record entered in Oricbase.

A command can only be given when 'please enter a command' and the flashing cursor are displayed. When a number of records have been entered, they can all be displayed by the DUMP command. With a printer attached, the PON command causes all output to be directed to the printer, and DUMP, for example, will then list all the records on the printer rather than the screen. POFF cancels PON, routeing output to the screen again. The file that has been entered can be saved on cassette with the SAVE command. A file saved on cassette is loaded again by selecting the 1 option when Oricbase is showing its initial display of Figure 5.6.

Examining the data

Once a file is stored in the computer, whether by typing it in or by loading it from a cassette, we can examine it. The command for selecting records is FIND. If this is followed by a test then all the records meeting the condition in the test will be found. We could find the record for the Newsnight programme by

```
FIND TITLE = Newsnight
```

The programmes for Tuesday could be found by

```
FIND DAY = Tuesday
```

and those starting after 9 o'clock in the evening on Tuesday by

```
FIND DAY = Tuesday AND TIME > 21.00
```

Note that there must be a space in between all the words and numbers, and that a space must be placed before and after the equals and greater than signs. But these commands will only find the specified records. They do nothing else, and there is no outward sign of what they have done. If we are searching for Newsnight, it is presumably because we want to know when it is being shown. So when the computer has found that record, we want it to display the DAY and TIME for it. We do this with

```
FIND TITLE = Newsnight ATRECORD PRINT DAY
             TIME
```

To see the title and type of evening program starting after 9 o'clock on Tuesday we type

```
FIND DAY = TUESDAY AND TIME > 21.00
           ATRECORD PRINT TITLE TYPE
```

There is a DELETE command for deleting records, and we could delete the programme called 'Dallas' with

```
FIND TITLE = Dallas ATRECORD DELETE
```

A move command allows us to move data. It can be used for correcting entries, and if we remembered that we had entered ABC2 for the CHANNEL on one record, but were not sure which record it was, we could correct it with no further fuss by

```
FIND CHANNEL = ABC2 ATRECORD MOVE
               BBC2 TO CHANNEL
```

Arithmetic can be performed on records using ADD, SUBTRACT, MULTIPLY AND DIVIDE. By using ADD, we could amend all the records that were affected if all the programmes shown after 9 o'clock on Tuesday were delayed by half an hour with

```
FIND DAY = TUESDAY AND TIME > 21.00
           ATRECORD ADD 0.30 TO TIME
```

There are two commands for sorting records into order. With SORA they are sorted into ascending order, and with SORD into descending order. The command

```
SORA TITLE
```

causes the records to be sorted into alphabetical order by titles. After

```
SORD TIME
```

they will be sorted into order with the programmes shown latest first and those shown earlier last.

By using BEGIN and END we can give commands that include commands to be done once at the beginning and once at the end, as opposed to being done at every record with ATRECORD. Using these, we can count the number of sporting programmes in our file, or since it amounts to the same thing, the number of records with 'sport' under TYPE, by

```
FIND TYPE = Sport BEGIN MOVE 0 TO #1
            ATRECORD ADD 1 to #1 END PRINT
            #1
```

Summary of Oricbase commands

Because we have introduced rather a lot of commands in a short space, this section collects them all together for reference purposes. Oricbase possesses still more commands, but the ones we have met are:

For creating, displaying and saving files.

ADD	to add a new record.
DUMP	to display a file.
PON	to activate the printer.
POFF	to cancel PON.
SAVE	to save a file on cassette.

For selecting records.

FIND	to find a record.
AND	for use in a test.
ATRECORD	to introduce the action to be taken at every selected record.
PRINT	to display data.
BEGIN	to introduce an action performed once at the beginning of a search.
END	to introduce an action performed once at the end of a search.

For processing records.

DELETE	to delete a record.
MOVE	to move data.
ADD	
SUBTRACT	to do arithmetic on data.
MULTIPLY	
DIVIDE	

For sorting files.

SORA	to sort into ascending order.
SORD	to sort into descending order.

Summary

This section has introduced the idea of a database and described the uses to which it can be put. An introduction to Oricbase is then provided by showing how it can be used to create, examine and modify a database. Oricbase has facilities beside those described here, and the user is referred to the manual for a description of them. All in all, this section provides a general introduction to databases and a specific introduction to Oricbase which shows in both general and specific terms how an individual's needs for storing and retrieving information can be met.

Spreadsheets and 'Oric-calc'

By running a spreadsheet program, a computer provides its users with the electronic equivalent of pencil, sheets of paper and a calculator. A spreadsheet can be used for preparing tables of numbers, and for performing calculations on them, so that the effects of changing one value or another in the tables can be investigated. In this sense, a spreadsheet is an 'electronic worksheet'. The advantages it brings when compared to pencil and paper methods are speed, convenience and the ability to handle large amounts of data with ease. By allowing the rapid and direct investigation of many alternatives that can occur in a given situation, particularly in a complex one, the spreadsheet becomes an ideal tool for planning and forecasting.

The user of a spreadsheet program is initially presented with a blank sheet. The sheet provides a number of positions at which entries can be made, and these are arranged in rows and columns. Each position in this tabular arrangement is known as a 'cell', and our entry is made in a cell by moving the cursor onto that cell, typing the entry and pressing RETURN to show that it is complete. The entries can be numbers or text, and there is one other possibility, as we shall see. But by placing numbers or text in the appropriate cells a table such as the following can be prepared. Text provides headings and labels for the table and numbers the entries in the table itself.

A small table

Quarterly profits

	Sales	Costs	Profit
First quarter	1500	1000	500
Second quarter	1600	800	800
Third quarter	1400	800	600
Fourth quarter	2000	1100	900

Although a spreadsheet allows tables like this to be prepared rapidly and conveniently, this only begins to hint at its capabilities. What makes it much more useful is that a formula can be associated with any cell. When this is done the spreadsheet does not display the formula: it calculates a value from the formula and displays that. This ability helps even in preparing a table as small as ours, for the entries in the third column are the differences of those in columns one and two. So in this case the table could also be prepared by entering the formula 'cell in column 1 — cell in column 2' each time the cursor is positioned on the cell in the third column. We could also display the annual profit in a cell somewhere by placing the formula for the sum of the entries in column three in that cell.

When our table is prepared by placing numbers in the third column, we have a once-and-for-all table that serves to display the figures but can do nothing more. But if the entries in column three are formulae we have a much more useful table for when the number in one of the cells in columns one and two is changed, the entry in column three having this cell in its formula changes correspondingly. So will the total profit cell. Whatever figures are placed in the table for sales and costs, the spreadsheet will automatically calculate and update the display of the quarterly profit, and of the total profit.

All spreadsheets have this property of re-calculating and updating the values of formulae whenever a change is made to a cell that is involved in a formula. This means not only that calculations are left to the spreadsheet, which can do them more reliably than us, but also that the spreadsheet can rapidly display any and all changes in a complex situation, whereas we might neglect some of them.

When our table is prepared by using formulae, it can be used to present the quarterly figures of any company in any year. All that is necessary is to enter the necessary figures in columns one and two. In this way it is a general model for a quarterly table. It can also be used to show the effects that result from having different figures in the columns where data must be entered. This illustrates, in a small way, that a spreadsheet is ideal for investigating the effects of changing data values. It will provide immediate answers to 'what if ?' questions about the data that are of the kind that need to be answered when planning ahead to prepare budgets, make forecasts and plan investment.

In fact, by using a spreadsheet, models of a wide range of activities can be created, including a company's operations, a proposed enterprise and an investment portfolio. The ways in which future activities may proceed can then be examined quantitatively, and coherent plans can be prepared based on an analysis of the possible occurrences. Creating a model requires a knowledge of the initial data items that are needed, but its essence lies in deriving the formulae to represent the situation. As an illustration, when modelling a business we might expect the fixed costs of the business and its income from sales to be provided as data items and, correspondingly, to be entered as data items on a spreadsheet. The overall costs of the business may be given by the costs associated with making the items that have been sold plus the fixed costs. The overall costs can then be calculated from a formula, and can be entered on the spreadsheet in that way. The formula would give the overall costs as some fraction of the income from sales plus the fixed costs. The fixed costs, despite being calculated from a formula, can in turn be incorporated in a further formula for the profit.

When modelling any complex situation, the spreadsheet will have to support a very large table. It is impossible to display a large sheet on the screen in its entirety and in this circumstance the screen acts as a 'window' through which a part of the sheet can be seen. By moving the window, any part of the sheet can be seen.

An example of spreadsheet usage

In this section, we give an example to show how a spreadsheet can be used. It is quite a small scale example because it is not practical to develop a large one in a book. A spreadsheet really comes into its own with a large sheet, but a small example can illustrate the basic ideas behind constructing and using one.

The example deals with a personal investment portfolio. Anyone who has shares in, say, three companies can create the following work sheet to show their holding in each company, its current price and the current value of the holding. The holding is a number of shares, the price is expressed in pence and the value of the holding is in pounds.

This table can be entered as numbers and text only, but since column three depends on columns one and two it is preferable to enter formulae in column three. There is no alternative to entering the numbers in columns one and two, for the holdings and the prices represent the basic data.

Company	Holding	Price	Value
Sainsbury	200	276	552
ICI	500	602	3010
Amersham	250	268	670
		Total	4232

When a spreadsheet program presents its initial blank sheet, the columns are labelled with numbers and the rows with letters as shown for Oric-calc in Figure 5.10. The rows are labelled, from the top downwards, by AA to AZ, then BA to BZ and so on for as long as necessary towards ZZ. Similarly, the columns are labelled from left to right by 1, 2, 3 and so on. The user can fix the size of the sheet to be used by giving the number of rows and columns for it. Any cell can be identified by giving its row letters and column number. The cell at the top left is cell AA1, and the cell to its right is AA2. We can position our table in the following way:

	1	**2**	**3**	**4**
AA	**Company**	**Holding**	**Price**	**Value**
AB	Sainsbury	200	276	552
AC	ICI	500	602	3010
AD	Amersham	250	268	670
AE				
AF			Total	4232

To do this, we position the cursor on each cell in turn and type the appropriate text, number or formula.

Since the value of a holding is obtained by multiplying the number of shares held by the price per share, to give it in pence, and then dividing by 100, to convert it to pounds, we have the formula we need for the column headed 'Value'. In cell AB4 we must put the formula AB2 * AB3/100, in AC4 the formula AC2 * AC3/100 and in AD4 the formula AD2 * AD3/100. Since these formulae are all very similar, and this occurrence is typical in spreadsheet usage, there is usually a facility for copying them, to save typing each individually. The total is obtained by placing the formula AB4 + AC4 + AD4 in cell AF4.

This worksheet can be extended to allow us to keep track of the gains and losses on our share holding. By adding in column 5 a new set of entries headed 'Cost' we can show the original cost of each share (in pence). This will have to be entered as data. But then we can use a formula to show the gain in column 6. The formulae we need are, for AB6, AB4 – (AB2 * AB5)/100, for AC6, AC4 – (AC2 * AC5)/100, and for AD6, AD4 – (AD2 * AD5)/100. Again, the formulae show a pattern. We can show the total gain in AF6 with the formula AB6 + AC6 + AD6. The resulting table is:

	1	2	3	4	5	6
AA	**Company**	**Hold-ing**	**Price**	**Value**	**Cost**	**Gain**
AB	Sainsbury	200	276	552	251	50
AC	ICI	500	602	3010	599	15
AD	Amersham	250	268	670	242	65
AE						
AF			Total	4232		130

Now, as the share price changes or as our holding in a company changes, all we have to do is change the relevant data, and the spreadsheet automatically recalculates the other values in the table that alter in consequence. Of course, a completed sheet can be saved to be reloaded, into the computer later, saving us the trouble of entering it all over again.

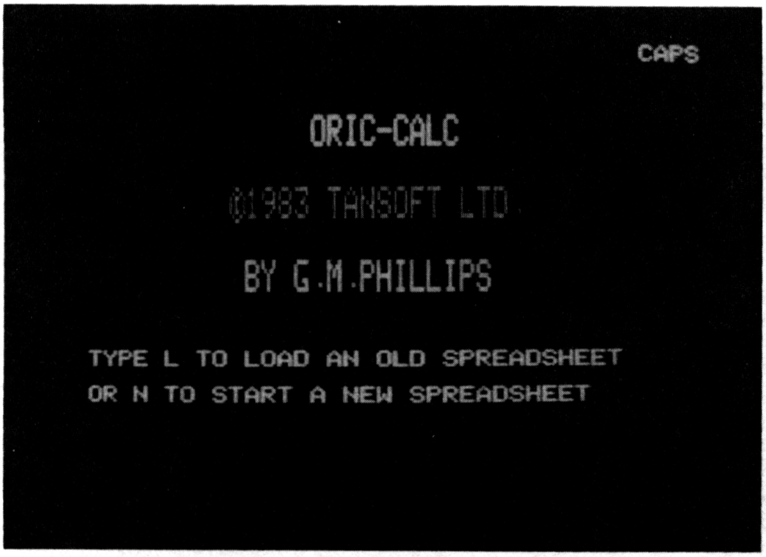
Using Oric-calc

Oric-calc is the Atmos' spreadsheet program. It can be used for the purposes indicated above, and it possesses the features that this discussion has led us to expect, plus a few others. When it is loaded from its cassette it gives the display shown in Figure 5.9. From here we can load an existing spreadsheet or create a new one and, since we are just starting, we will press N for the latter. Before we start to make entries, the program needs to know how big the spreadsheet is to be and how it is to display numbers. We can respond to its queries with:

```
HOW MANY ROWS (DOWN) 20 – 99? 20
HOW MANY COLUMNS (ACROSS) 4 – 54? 10
HOW MANY DECIMAL PLACES – 0 – 9 OR V IF
VARIABLE FORMAT 0
```

The last entry causes the spreadsheet to display numbers with no places after the decimal point, that is, as whole numbers. After this, Oric-calc displays a blank sheet ready to be filled in, as shown in Figure 5.10. It is also ready to accept commands, and they can all be given by pressing a single key.

The cursor can be moved around the sheet with the cursor movement keys. It can also be moved by giving the G (for go) command followed by the identification of the cell it is to go to. By these means we can position the



CAPS

ORIC-CALC

©1983 TANSOFT LTD

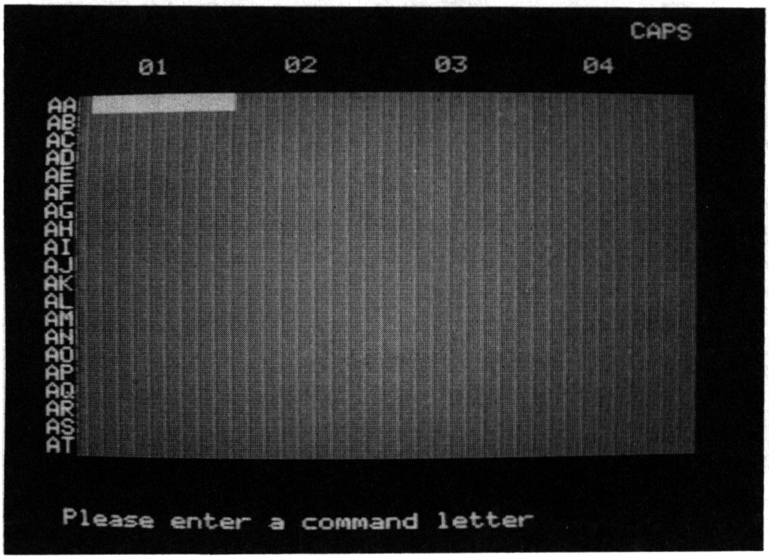
BY G.M.PHILLIPS

TYPE L TO LOAD AN OLD SPREADSHEET
OR N TO START A NEW SPREADSHEET

Figure 5.9 The initial display of Oric-calc.

cursor where we want it. We are now in a position to enter the spreadsheet described towards the end of the previous section.

We will start at the top left corner, so if the cursor is not already at cell AA1, either move it there using the cursor movement keys or give the command G AA1. We want to place the text 'COMPANY' here, so first we give the command for entering text by pressing the quotation mark key ('), then we type the text and when it is typed press RETURN. As soon as we press RETURN, the text appears in this cell. All the other items of text can be entered in exactly the same way by moving to the cell, giving the text entry command, typing the text and


CAPS

```
     01         02         03         04
AA
AB
AC
AD
AE
AF
AG
AH
AI
AJ
AK
AL
AM
AN
AO
AP
AQ
AR
AS
AT

Please enter a command letter
```

Figure 5.10 The blank sheet provided by Oric-calc.

pressing RETURN. When all the text is entered, we can tackle numbers. We shall enter the numbers under 'HOLDING' first, so move the cursor to cell AB2, give the command N for entering numbers, type 200 and press RETURN. After pressing the cursor down key, we can enter the number below it in exactly the same way and by continually doing this, the column can be filled. There is a slightly easier way to enter numbers which we can illustrate by filling the 'PRICE' column, so go to cell AB3 and give the A command. This is for entering a number and then automatically moving to an adjacent cell ready to enter the next number. As we want to proceed down the column, press D. Now the number 276 can be entered by typing it and pressing RETURN, but after this the cursor moves to cell AC3 and the spreadsheet is ready to accept a number for this cell. When all the numbers in the column are entered, just pressing RETURN by itself causes the automatic entry of numbers to cease and the spreadsheet is again ready to accept any command. When all the numbers are entered it only remains to enter the formulae. We shall begin by entering the formulae for the 'VALUE' in column 4, so move to cell AB4 and give the F command for entering a formula. The formula AB2 * AB3/100, as given in the previous section, can be typed and as soon as RETURN is pressed the spreadsheet pauses to calculate its value and then places the value in AB4. The rest of this column can be filled in the same way by moving to a cell, giving the F command, typing the formula and pressing RETURN. But, again, there is an easier way to fill a column, and we can illustrate this with the formulae for 'GAIN' in column 6. First move to cell AB6, type F and then the formula AB4 – (AB2 * AB5)/100. When this formula is entered it can be copied, with the appropriate modifications made automatically down the column. Give the C command to copy the formula in the current cell. The spreadsheet will then ask where we want it copied. It can be copied into a rectangular region of the spreadsheet which we identify by giving the top left and bottom right corners. In this case we want to copy it from the top of a column at AB6 to the bottom at AD6, so we enter AB6 followed by AD6. As soon as this is done, the spreadsheet copies the formulae, completes their values and displays the values in the appropriate cells. To see the formula associated with any cell, you need only place the cursor on that cell. The formula is then displayed at the bottom of the screen. In this way you can confirm that the formulae created by the copy command are the ones you want.

To finish the sheet we must add the totals for the GAIN and VALUE columns. To deal with VALUE first, move the cell AF4 and enter the formula AB4 + AC4 + AD4. If the column had more entries, this formula would be correspondingly lengthy and, as usual, there is a short cut. We will illustrate it with the formula for the total gain. After moving to cell AF6 and typing F, we could give the formula AB6 + AC6 + AD6, but it can be abbreviated to [AB6 AD6, that is, a square bracket followed by the first cell in the addition and the last cell in the addition. This completes the spreadsheet, and it is shown in Figures 5.11 and 5.12. We need two figures because the window on the spreadsheet is only four columns wide and so two windows must be shown to give the whole spreadsheet.

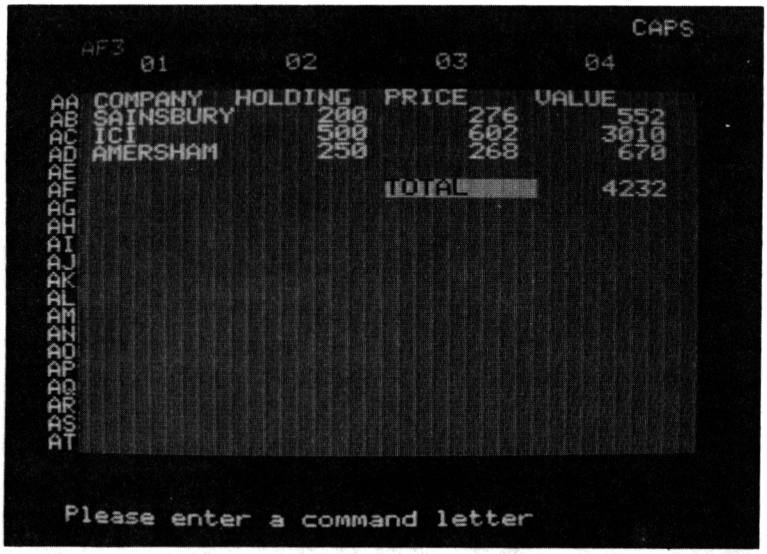
But what happens when the prices of the shares change? All we have to do is enter the new price and tell the spreadsheet to re-calculate all the formulae. To illustrate, suppose shares in Sainsbury rise to 284 pence. We can enter this by:

G AB3
N 284

and then pressing W causes the re-calculation of the value, gain and totals that are a result of the change in data. Oric-calc does *not* do re-calculation automatically, it must be commanded to do it. Finally, Oricbase provides the Q command to suit the spreadsheet and returns to the display of Figure 5.9, the S command for saving a spreadsheet on cassette, and P for printing it on a printer.

Summary of Oric-calc commands

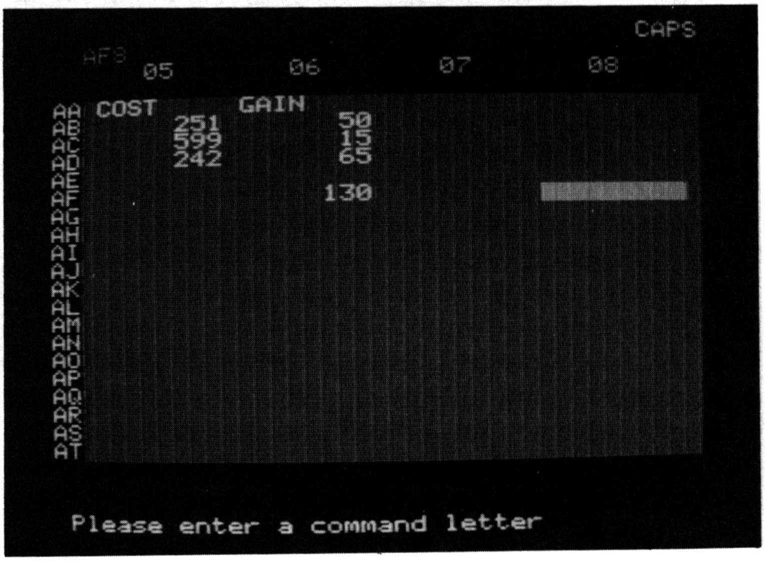
The Oric-calc commands introduced in the previous section are listed below. There are others, and they are given in the Oric-calc manual.



```
                                              CAPS
  AF3
       01          02          03          04

AA COMPANY   HOLDING   PRICE       VALUE
AB SAINSBURY     200         276         552
AC ICI           500         602        3010
AD AMERSHAM      250         268         670
AE
AF                     TOTAL                4232
AG
AH
AI
AJ
AK
AL
AM
AN
AO
AP
AQ
AR
AS
AT

  Please enter a command letter
```

Figure 5.11 Window on part of a spreadsheet.

A	Accept a number and move to the next cell ready to accept another number.
C	Copy a formula to other cells, adjusting it for them as necessary.
F	Accept a formula.
G	Go to a specified cell.
N	Accept a number.
Q	Quit the spreadsheet.
S	Save the spreadsheet or cassette.
' (quote)	Accent text.
P	Print the spreadsheet on a printer.



```
                                               CAPS
   AF8  05          06          07          08
AA COST        GAIN
AB      251              50
AC      599              15
AD      242              65
AE
AF                      130
AG
AH
AI
AJ
AK
AL
AM
AN
AO
AP
AQ
AR
AS
AT

Please enter a command letter
```

Figure 5.12 The remainder of the spreadsheet of Figure 5.11.

Summary

This section provides an introduction to spreadsheets and their applications in general, and to Oric-calc and its use in particular. A spreadsheet is an 'electronic worksheet' that can be used to display tables, and to investigate the consequences of changing any entries in the table. By allowing the investigation of various alternatives in a given situation, it is an ideal tool for planning and forecasting. Although Oric-calc does not possess all the refinements of other spreadsheets it does provide the basic spreadsheet functions.

Appendix I

The keys used with CTRL and ESC

The following table shows the keys that can be used with CTRL and the effects they have. In each case holding down CTRL and presing the key in question gives the effect. Holding down CTRL and pressing L, for example, clears the screen.

The keys used with CTRL and their effects

Key	Effect
A	Copies the character under the cursor.
C	Halts the program being run and returns the ready message.
D	Causes double line printing, so that everything printed is repeated on the line below.
F	Turns the click produced when a key is pressed on and off.
G	Rings the bell.
H	Moves the cursor to the left.
I	Moves the cursor to the right.
J	Moves the cursor down.
K	Moves the cursor up.
L	Clears the screen.
M	Gives a RETURN.
N	Clears the line occupied by the cursor.
O	Disables the screen so that what is typed is not displayed.

Q	Turns the cursor on and off.
S	Freezed and releases the screen display.
T	Switches between the display of capital and lower case letters when a key is pressed.
X	Cancels the line being typed.

Pressing ESC and another key *simultaneously*, which is none to easy to do, gives a space on the screen and affects the part of the line to the right of that position in various ways, some of them quite spectactular. The keys used with ESC and the resulting effects are listed in the next table.

The keys used with ESC and their effects

Key	Effect
@	Gives black ink.
A	Gives red ink.
B	Gives green ink.
C	Gives yellow ink.
D	Gives blue ink.
E	Gives magenta ink.
F	Gives cyan ink.
G	Gives white ink.
H	Gives the standard characters.
I	Gives different characters including block symbols for graphics.
J	Gives double height characters with their top and bottom halves or alternate lines.
K	Gives the alternative characters in double height.
L	Gives the standard characters and makes item flash on and off.
M	Gives the alternative characters flashing on and off.

N	Gives the standard characters in double height and flashing.
O	Gives the alternative characters in double height and flashing.
P	Gives black paper.
Q	Gives red paper.
R	Gives green paper.
S	Gives yellow paper.
T	Gives blue paper.
U	Gives magenta paper.
V	Gives cyan paper.
W	Gives white paper.

INDEX

NOTES

NOTES

NOTES

NOTES

NOTES

USING YOUR ORIC ATMOS
by Garry Marshall

The aim of this book is to introduce the newcomer to the use of the Oric Atmos computer. It is divided logically into three main areas: first, an introduction to the hardware; second an introduction to BASIC; and third an introduction to the software. The reader can learn how to use the Oric Atmos, quickly and easily and at the same time discover its unique capabilities.

Features of the hardware

Inside the computer

Expanding the computer

Features of the Oric Atmos

Graphics

Sound

Colour

Word processing on the Oric Atmos

Databases on the Oric Atmos

Spreadsheets on the Oric Atmos

£3.95

N A Newtech Production

ISBN 0-09-938790-5

00395

9 780099 387909

HCP

MADE EASY–USING YOUR ORIC ATMOS Marshall ARROW