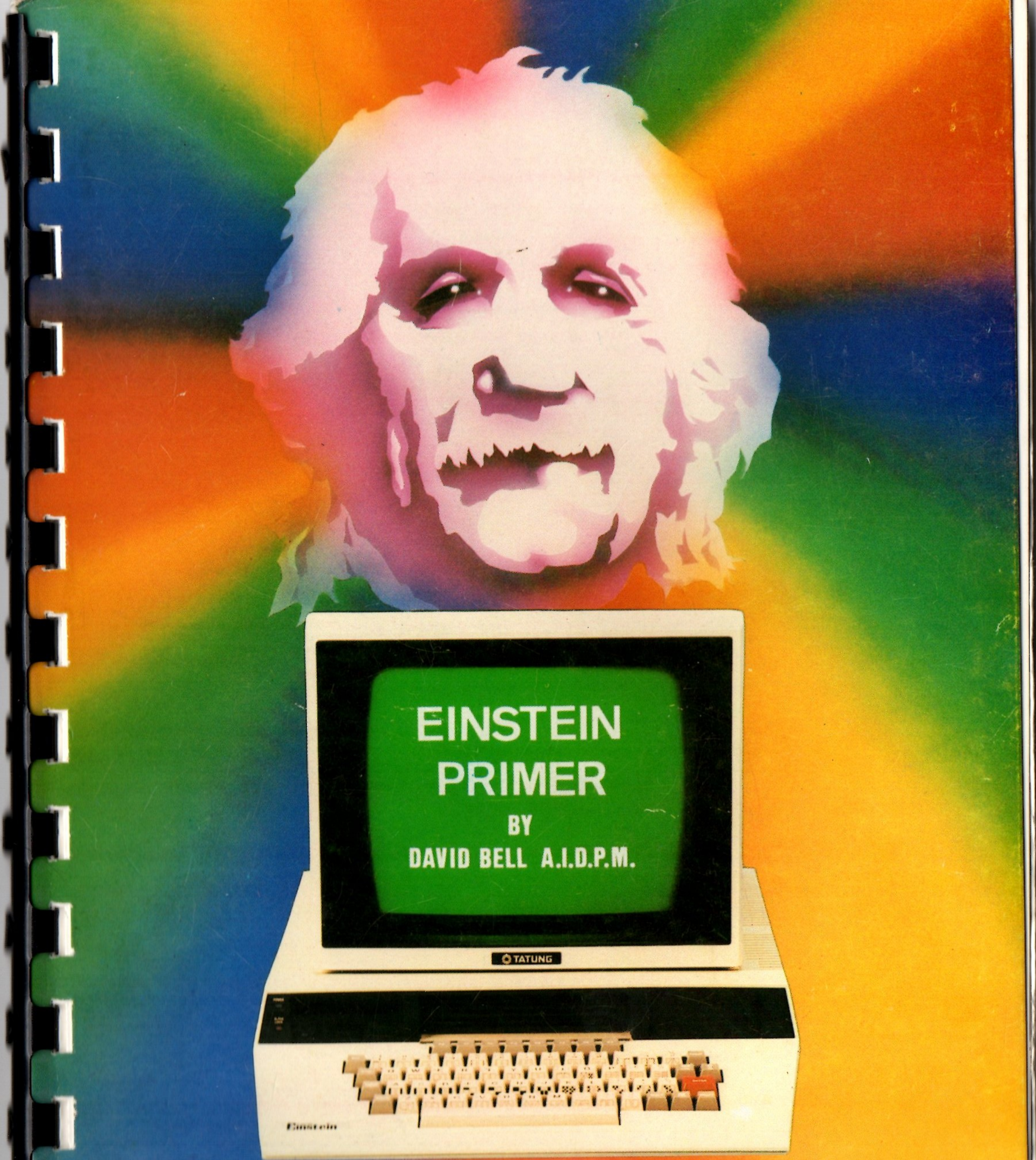


A BEGINNERS GUIDE TO THE

EINSTEIN MICROCOMPUTER

EINSTEIN PRIMER

CONTENTS

INTRODUCTION 3

PART ONE

THE EINSTEIN COMPUTER 4

THE BUSINESS COMPUTER 9

OPTIONAL EXTRA HARDWARE 14

PART TWO

POPULAR SOFTWARE 19

PART THREE

PROGRAMMING IN XBASIC 27

EINSTEIN PRIMER

Introduction

The purpose of this book is to introduce a new user of computers to the Einstein micro computer and to computing in general. It is not intended to teach you how to program, there are many such books available on the market, and some products such as RELATIVELY BASIC, EINSTEIN NEWS, THE BRAIN MAGAZINE, BASIC TUTORIAL and EINSTEIN USER are dedicated to the Einstein computer.

This book is divided into three sections.
PART ONE: Introduces you to the concept of computing and the Einstein in particular.

PART TWO: Introduces some of the more popular pieces of software and acts as guide to their purpose.

PART THREE: Serves as an introduction to computer programming and the use of various programming languages.

I hope that you find this book useful and that, after reading it, you are able to put your Einstein to full use and get many hours of pleasure out of it.

DAVID BELL A.I.D.P.M

PART ONE

CHAPTER 1
THE EINSTIEN COMPUTER

Before we start it is as well to understand some of the basic concepts of the subject as this section sets out to explain.

A micro computer can be divided into three main units:-
a) the central processing Unit (CPU)
b) the memory
c) the Input/Output (I/O) units

The input/output units can consist of such items as:-
1) The visual display unit (VDU)
2) The storage device (floppy disk, hard disk, tape etc)
3) Printer
4) keyboard

The CPU or central processing unit is the heart of the system and contains the chips that carry out the processing of instructions which are fed into the CPU from the I/O devices e.g. the keyboard or disk drives.

The main processing (or memory) chip in the CPU is the Z80A and this enables the Einstein to handle programmes written in CP/M (one of the main languages in which Business and Application programmess are written). Other chips in the CPU, handle such things as the Video Display, Parallel Interface, Serial Interface and Sound generator. It is not necessary for the average user to understand these, only to realise that further information is available in the books "AN INTRODUCTION TO TATUNG EINSTEIN", "THE HARDWARE MANUAL" also Volume 1 Number 1 of the EINSTEIN USER magazine.

The Input devices, as the name implies, accept information from the outside world and pass it on to the computer. This information is commonly in the form of a program or instructions which are entered from the keyboard. The Einstein is able to accept information from a wealth of sources, such as Internal and External disk drives, Keyboard, RS232C serial port, Analogue inputs and the Pipe. Information can be sent to the VDU, sound generator, parallel printer port, serial port, disk drives, a UHF receiver and the Pipe.

The Output devices take information from the main processing unit and pass it on to the outside world, where it is either read or stored by such units and the floppy disk drives, video display unit or printer.

So much for the basics of the computer, let us set up a standard Einstein system and then add to it, so that it can be used for a wide range of business applications.

Installing the System

The heart of the Einstein is the TC01, this is what was referred to as the CPU in the section above. In the Einstein's case, this unit also includes one of the Input devices, a keyboard, and Input/Output device, a 3" floppy disk drive.

There are 4 sizes of floppy disk drives in common use today. These are the 8", 5.25", 3.5" and 3". The 8" and 5.25" are very susceptible to mis-handling, being contained in a flimsy protective cover which only really protects them from

fingerprints to 90% of the disk area. They are easily bent and the apperture where the disk head actually touches the magnetic material is left unprotected, whilst the disk is being inserted into the computer or returned to its protective envelope.

The 3.5" and 3" disks almost eliminate mis-handling of the disk. The magnetic material is housed in a hard protective covering and a metal shutter protects the area where the disk head actually touches the magnetic surface when it is not inside the computer. These disks are more expensive than the 5.25" and 8" counterparts but this is a small price to pay for the extra protection given; after all, what price can be put on the hours of data the computer has carefully stored on a disk which is suddenly made useless by a careless fingerprint, or the creasing of a floppy disk.

The TC01 has, as standard, a single 3" disk drive built in. This allows 188k of data to be stored on each of the two sides of a 3" disk. 1K is the equivalent of 1024 bytes, so 188k allows 192,512 bytes of data to be stored, ample for most domestic applications!

The TM01 or colour monitor, is normally housed on top of the CPU. This gives a much better quality of picture than a domestic UHF television receiver which can be connected into the Einstein. The monitor is connected to the CPU by means of a cable with a 6 pin DIN plug on each end. One end of this plug fits into the top-most of the two sockets on the back of the monitor and the other end fits into leftmost socket of the CPU when viewed from the rear. There is also a three way slider switch on the back of the monitor and for the present, this should be in the middle or colour position. The mains lead from the Monitor and computer should be connected to a suitable electrical supply and if plugs are not supplied, instructions for fitting them should be followed.

Switch on the computer and monitor following the instructions in the handbooks supplied and adjust the colour, brightness and contrast to give clear white letters on a blue background. Providing you have not inserted the system master disk, you should obtain a display on the screen similar to the following:-

```
*** E STEIN ***

Insert disc in drive 0 and
press Ctrl-BREAK to load

TATUNG/Xtal Mos 1.21                    (c) 1983 1984

Ready
>
```

Locate the SYSTEM MASTER DISK, this should have been in the box with the CPU and would have been enclosed in a cellophane bag with the copyright notice. A serial number will have been printed on the outside of the disk and at the present time, begins with the letter x. This is the number you should quote when contacting Tatung or Crystal Research for technical help. Side A is the correct side to insert into the drive and this should have the serial number printed on it.

There is a red WRITE PROTECT tab on the front left-hand side of each surface of a 3" disk. This is either a red slider switch or a white tab. With the red slider switch, you should push it across with the point of a ball point pen or similar instrument until you cannot see red through the hole underneath the slider. If you can see red, that means danger, you may write information to the disk or erase information from the disk. Locate the white tab, slide push this down, so that the tab is no longer flush with the edge. This is now similarly protected from accidental erasure. Information on this procedure is contained on the disk packing.

It is advisable to protect the disk in the above manner before inserting your system disk into the computer!

Having inserted your System Master Disk into the left-hand drive (also known as drive Ø:), follow the instructions given on the screen and hold the key marked CTRL down whilst pressing the BREAK key. You may have to hold the BREAK key down for a second or two, or alternatively try the procedure a second time but eventually you will be presented with the LOGON notice:-

```
                    ***  EINSTEIN  ***

TATUNG/XtalDOS 1.31                              (C) 1983 1984

Ø:
```

At this point, I should mention what is known as version numbers. The above version number is version 1 revision .31 of TATUNG/XtalDOS as is the current version at the time of writing this book. You may, for example, have the message TATUNG/XtalDOS 1.11 on the screen, in which case you may continue with the procedures mentioned in this book but should contact Tatung with a view to obtaining an upgrade. Upgrades are normally given free of charge to registered owners, providing their original disk is returned and contains the current version. Therefore, if you have version 1.11 you could obtain version 1.31 but not 2.Ø2, as this would be a new version. You are only entitled to upgrades of a revision to Dos.

Providing you have followed the procedures so far, you should have a white square flashing to the right of the drive indicator. This flashing square is known as the cursor and the drive indicator says that you are currently able to access drive Ø: (the left-hand drive). Anything you now type will appear where the cursor is flashing. Let's get a directory, or list, of the programmes and files which are stored on the system disk. Type DIR (short for DIRectory) and press the red ‹ENTER› key. Providing you inserted the disk the right way up, the light under the disk drive should glow green whilst the computer is reading the file names which are recorded on a track on the disk known as the DIRectory track. After a short while the light will go out and a list of the file names will appear on the screen. If there are more than 18 files on the disk, press the ‹ENTER› key to see the next 18 names. At the end of the list is printed the amount of space that these files have taken on the disk, the amount of remaining space and the total space that the disk had available when it was fresh.

You will notice that each of the file or program names is formed by a name to a maximum of eight characters, a period or full-stop and a three character file type. The file name can be any

character from the keyboard with the exception of ‹›.,;:=?*[]()/ or tab. A file name must start with a letter of the alphabet and not a number, therefore CHAPTER1 would be legitimate but you could not have 1STMAY.

The file type has caused some confusion with first-time users of the computer who are not programmers. Anything having a file type COM is a COMmand and this programme name can be typed into the keyboard from DOS (the state of the computer when the system disk was first loaded). Anything having the file type XBS is a program which was written using the language XBAS and this BASIC language, must be loaded into the computer before a programme with the file type XBS can be run. For example, suppose we wished to run the BASIC programme INTRO.XBS we would first need to load XBAS. We notice that XBAS has the file type .COM, as this is a COMmand programme, we can enter this direct from DOS, so we type in XBAS and press the ‹ENTER› key. After a short while the message:-

```
TATUNG/Xtal Basic 4.12                                   (C) 1983
1984
Size 43324
Ready
```

will appear on the screen (or something very like it)! You have now loaded XTAL BASIC and you are ready to run the programme INTRO.XBS To run this programme you must tell Xtal Basic to run it with the instruction:-

```
RUN "INTRO"
```

and press the ‹ENTER› key. Notice the speech marks.

The program will run and you will see the name EINSTEIN appear four times on the screen with the Tatung Logo, accompanied by sound. After the programme has run, the screen will repeat:-

```
Ready
```

waiting for the next instruction.

Let's return to DOS where we were, when the system disk was first loaded. The command from within XBAS to return to DOS is, believe it or not, DOS. Enter the letters DOS and press the ‹ENTER› key. You should be presented with the cursor flashing next to the indicator, showing that you are ready to access drive Ø:

From this you should realise that the indicaton that you have XBAS loaded into the computer, is the cursor flashing under the sign "Ready" and that the indication you are in DOS, is the cursor flashing next to the drive indicator. When you have XBAS loaded, you may run any programme with the file type XBS by entering RUN, then the file name enclosed within speech marks. To run a COMmand programme (one ending in .COM), all you need to do is enter the programme name when you are in DOS (i.e. the drive indicator is shown next to the cursor).

Type DIR to obtain the directory again. You will see another file type displayed. This is the file type .LOG These are programmes written using another language LOGO. (actually they are not called programmes but procedures, I will continue to call them programmes for simplicity). You may run a programme ending in LOG by first loading the language LOGO. This is exactly the same as loading the other language XBAS. Type LOGO from DOS and you will first see a copyright notice, then the cursor flashing next to a

?

We have now discovered a third indicator. When you have LOGO loaded, you are prompted with a ?

To load one of the LOGO programmes SKETCH.LOG, you use lower-case and enter the following:-

load "sketch <ENTER>

Notice you used lower-case and that you did not enter the closing speech-mark before pressing the <ENTER> key.

The computer will now load all the instructions it needs to process the programme SKETCH.LOG and when this is in the computer's memory, you will be presented with the ? prompt again. All that is now required to run this programme is to enter the programme name, so enter in lower-case:-

sketch <ENTER>

The screen will clear and a demonstration of some of the abilities of the language LOGO will appear.

The command to return to DOS is different in LOGO to XBAS. In LOGO you type bye and press the <ENTER> key after the ? prompt.

Type bye and press <ENTER> when the sketch demonstration has finished and this will return you to DOS. Once again type DIR to obtain a directory.

You will notice one other file type listed here. This is the file type .DAT These files are not programmes but DATa used by other programmes. The two files on the screen are DICTIONE.DAT and DICTIONH.DAT Both of these programmes are used by the XBAS programme HANGMAN.XBS to obtain words to play the game hangman. DICTIONE.DAT are the easy words and DICTIONH.DAT are the hard words. Trying to enter these file names from DOS will not work, but you could see the contents of the dictionaries (if you wish to cheat) by using one of the Built-In DOS commands. These are commands which are in the computers memory when DOS is loaded and are not commands stored on disk. Type:-

DISP DICTIONE.DAT

and you will be presented with a list of the words contained in the Dictionary of easy words used by hangman.

CHAPTER TWO
THE BUSINESS COMPUTER

The equipment discussed above would suit the average user but for those who require to conduct more serious business applications, this equipment will require supplementing.

The first additional piece of equipment the majority of business users will require is the TK 02 or 80-column monochrome card.

The Einstein is CP/M v 2.2 compatible, but to run the majority of the CP/M software requires a video display of 80 columns. The Einstein 80-column card is an economical piece of hardware which simply plugs into the PIPE at the back of the Einstein and allows the user to run either 40 column or 80 column software, by sending a simple control sequence from the keyboard.

A brief description of what is meant by CP/M would not come amiss but, if you require more in-depth knowledge, you may purchase the full CP/M package for the Einstein from your local dealer.

Some years ago, it was decided that there should be a standard operating system which would allow software producers to write software that could easily be transferred between 8 bit computers. This operating system became known as Control Program/Microcomputer or CP/M for short. CP/M was designed specifically around the floppy disk as a storage medium, therefore it will not work with cassette-tape systems but will work with a hard disk.

Because CP/M programmes have been available for some years, there are many thousands of programmes able to run under this system, though each requires some modification to run on specific computers. The most obvious is floppy disks that your computer accepts. This is not a major problem, as, if you cannot obtain the software you require "off the shelf", there are a number of companies specialising in transferring it to the Einstein format. XITAN LTD of Southampton is one such house and charges £8.00, plus the cost of the disk, at the time of writing this book. Another variation between computers are the codes that different computers require to handle the video display. Some computers require a different code, for example, to clear the screen than others. This procedure is known as configuring and, though this is not difficult for standard CP/M programmes, it is daunting for the first-time user and should be left until more programming experience is available. Software purchased for the Einstein specifically, will have already been configured for that system.

Another problem that the inexperienced person may overlook when choosing software, is the amount of storage space required on disk to run the system. This will be dealt with in more detail in the next section, but for now lets state that if you are running a full accounting suite with production control you would require a lot more than the standard Einstein. Let us now presume that you are a small businessman who wants to run a professional word processor and has chosen one of the better packages known as WORDSTAR PROFESSIONAL. You have correctly been advised that you require the Monochrome 80 column card and have purchased this to be added to your single drive Einstein and colour monitor.

Inside the 80 column card packaging you will notice an instruction booklet which, at first sight, may appear daunting,

but the main things to note are very simple.

Before you attach the card to the PIPE at the back of the computer you must decide one major thing. Do you require the computer to automatically power up in its original 40 column mode or in the new 80 column mode? If the majority of your use will be for business, then I suggest 80 column mode. If your machine is also used by children, then 40 column would be better. Whichever mode you set the card to, it is a simple matter to swap from one mode to the other, by use of a control sequence from the keyboard.

Holding the CTRL key down and pressing the key P then <ENTER> will place you in the 80 column mode from 40 columns!

Holding the CTRL key down and pressing the key N then <ENTER> will place you in the 40 column mode from 80 columns!

For the sake of illustration let us assume that you are mainly a business user and that WORDSTAR PROFESSIONAL will be the piece of software which you will be using the majority of the time. You will notice that Fig 4 of the 80 column card instruction book shows you how to alter "link positions" to alter this power up sequence. I stress at this point that the procedure is not difficult and requires no soldering.

Hold the 80 column card printed circuit board carefully with the ribbon connector coming out of the card towards you. You will notice 4 link connectors to the left of the ribbon cable. These are labelled M001, M002, M003 and M004 (see Fig 2 of the instruction booklet). The connector M003 is the one you need and you will see that it is made up of 3 pins and a piece of material that forms a bridge over two of them. If the bridge covers the bottom two, then the computer will power up in the 40 column mode. If the bridge covers the top two, then the display will be 80 columns at power-up. If you have just purchased the 80 column card, then it is probable that the bridge will have been set at the factory to power up in 40 columns. If this is so, then carefully pull the linking bridge from the bottom two pins and slide it down over the top two. When this is done, follow the instructions given in the handbook to attach the card to the pipe on the back of the computer. Again this is not difficult and should only take five minutes or so with a screwdriver.

We now come across the one snag with the system, which I have chosen as an example. The 80 column card was not designed to run with the 40 column colour display, but with the Tatung TM80 green screen monitor. If this is what you purchased, you can ignore the next paragraph or two and connect the cable that came with your 80 column card to the green screen monitor and the socket on the 80 column card.

The output from the Einstein computer is taken via a 6 pin din plug to the colour monitor. This provides the standard colour output when you are using the computer in 40 column mode and is the way the computer should be set up when running the computer in 40 columns <u>even with the 80 column card attached</u>!

The output from the 80 column card is taken from a phono socket, so obviously the 6 pin din cable is useless. The simplest way is to ask your dealer to supply you with a three-way switched cable allowing you to run both 40 columns or 80 columns by flicking a switch and pressing the control sequence from the keyboard. If you do not wish to pay for this, then you will have to make up a cable for yourself. This is accomplished by soldering Pin 2 of a

6 pin din plug to the centre of a phono plug and pin 5 to the outside of the phono plug. You may now attach the phono plug to the 80 column card and the din plug to the colour monitor, but, obviously, you will have to attach the other cable to the monitor from the computer when you wish to show 40 column colour. It is your choice whether you change the switch on the monitor to green screen when running in 80 columns, or watch in white on black with the switch set to colour. In either case there is some loss of quality over the recommended TM80 green screen monitor, which was designed for the purpose.

So much for the 80 column card. You may now run the word processing package WORDSTAR, but the limitations of having a single disk drive will quickly become apparant. When using more sophisticated pieces of software, the program utilises a considerable proportion of the disk space, leaving little facility for data storage.

Take the WORDSTAR PROFESSIONAL package. This has become one of the standard packages for word processing and it is possible to do simple word processing with a single disk drive. To make use of the other features, such as mail-merge and the spelling checker, additional drives become almost mandatory, especially if your text is of any great length.

How easy is it to fit an additional internal second disk drive? The answer is <u>very</u>. All that is required is a screwdriver and a little patience.

Obviously before you start opening the Einstein to fit the 2nd internal disk drive kit, the computer should be unplugged from the mains. Having done this the first job is to remove the two screws on the back of the computer (these are the ones that were removed to fit the 80 column card). The computer cover will now lift off, if you ease it to the rear as you lift the cover up.

You next have to slide out the blanking plate covering the space where the 2nd disk drive will sit on the right-hand side of the computer's fascia. You will notice a ribbon cable running from the original disk drive to a socket on the printed circuit board labelled M005. This cable should be carefully disconnected. Your next job is to check that the disk drive kit you have purchased, has been set up at the factory to be the second internal disk drive. There is a small ceramic box at the front of the drive and this has seven pairs of link connectors (similar to those on the 80 column card). The instruction booklet supplied with the kit should be consulted, but the links required are marked DS1 and M1. It is likely that these will have been correctly set up and no alteration should be needed.

Next, slide the disk unit into its sleeving acting as a shield and partly screw in the four retaining screws. The ribbon cable has to be folded twice in the shape of a flat 'U' to lie on the floor of the computer. One end of the cable goes into the socket marked M005A and the middle plug of the cable goes into the new (2nd) disk drive. The plug on the other end goes into the socket of the original disk drive, where you earlier removed the ribbon cable. There is a coloured stripe down one edge of the cable and this should be on the side of the power supply, where it plugs into socket M005A. Plug the supplied 2nd disk drive power lead into socket M009 and the other end into the drive itself and you are ready to reassemble the cover onto the Einstein. You may have to press quite hard on the cover to make it sit into place and don't forget to replace the two restaining screws and the 80 column card, if fitted. You may now connect your computer to the

mains power supply and commence use. The whole sequence sounds alot more complicated than it is and further instructions can be obtained from the instruction booklet provided with the 2nd disk drive or volume 1 number 2 of the Einstein User Magazine.

To check if the second drive works, place your system disk into drive Ø and another programmed disk into the right-hand drive (drive 1). Press the break key whilst holding down the CTRL key to load the operating system. If all is well at this stage type in :-

DIR 1: <ENTER>

The drive light for the right-hand drive should now illuminate and you should see the DIRectory of the disk in drive 1 on the screen. Should this not be the case, then retrace your steps or consult your local Einstein dealer for advice.

You now have the basis of a professional business computer, able to run a wide range of business software packages. There is still one product missing required for the majority of business applications. This is a printer, to obtain the output from the computer, whether this is in the form of letters, financial forecasting or sales invoices.

There are a number of printer types on the market, ranging from very cheap thermal printers to the ultra fast laser, but the average user need only concern himself with two types; the DOT MATRIX and the DAISY WHEEL. They can also be further subdivided into serial and parallel types. Parallel types are the easier to connect to the computer and these are the ones recommended to a beginner, though, once again seek your dealers advice. After all, if he is setting up a complete system, then he will deal with the problems of configuring your system for a serial printer.

Let us commence with the DOT MATRIX printer, normally costing less than the daisy wheel, capable of high speed and the capability of producing graphics (something for which the daisy wheel does not cater)

Print from a dot matrix printer is made up of tiny dots formed by a matrix of needles striking the ribbon and imprinting a character on the printers paper. The character is electromechanically controlled and formed into shape by the computer program and hardware. The more common matrix printers have a matrix of needles 7 x 8 or 8 x 9 which form the characters. The greater the number of needles, the more 'letter-quality' the print. Most good dot-matrix printers can print in both directions, thereby giving greater print speed to long documents.

The printer speed is measured in characters per second (CPS) and with the Tatung TP100 dot matrix printer, this is 100 cps., bidirectional.

A daisy wheel printer is another impact printer used in small businesses and at home. It is more commonly used in word processing applications, where true 'letter-quality' print is required. Daisy wheel printers tend to be slower in printing when compared with dot matrix printers of the same cost.

The type is carried on the tips of the spokes of a small metal or plastic wheel resembling the head of a daisy or similar flower. The typeface can easily be exchanged by interchanging the daisy wheel.

The choice of printer will depend on the application and buyer's pocket. The problems experienced by the user tend not to be with the choice between dot matrix or daisy wheel, but interfacing them with the software used.

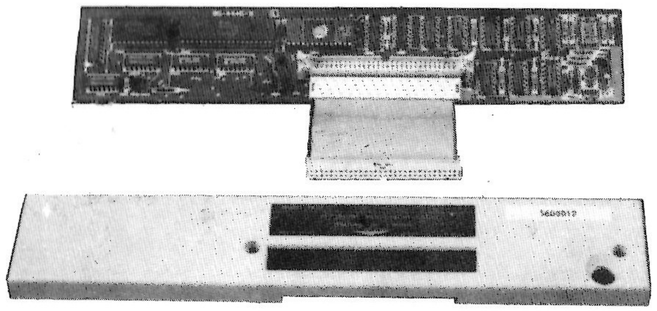
This is because different makes and models of printers require different codes to instruct them to perform certain control functions. The code for an underline instruction on the Epsom RX-8Ø, for example, would be different to that of, say, a Brother HR5. There is a standard code for data communication (ASCII), but this is only for 128 characters including upper and lower case letters, digits, symbols and certain control characters. Unfortunately, this does not include every eventuality, so different printers expect different codes to be sent. A different example would be the £ sign. There is no standard ASCII code for this, so pressing the £ key on the Einstein will produce different characters to be printed on the printer. most software and a number of printers go some way to allowing the user to customise their products, but, once again, this is not easy for the casual user.

One of the easier Word Processor packages to customise is TASWORD and the documentation that accompanies the product devotes six pages to detailing how this is done. Wordstar professional has a similar procedure but, in my opinion, much more complicated to understand. The best advice is to consult the computer manufacturer or dealer as to the recommended printer for your particular purpose. Alternatively, purchase the printer suggested by the computer manufacturer, as it is likely that the software will be compatible, without customisation.

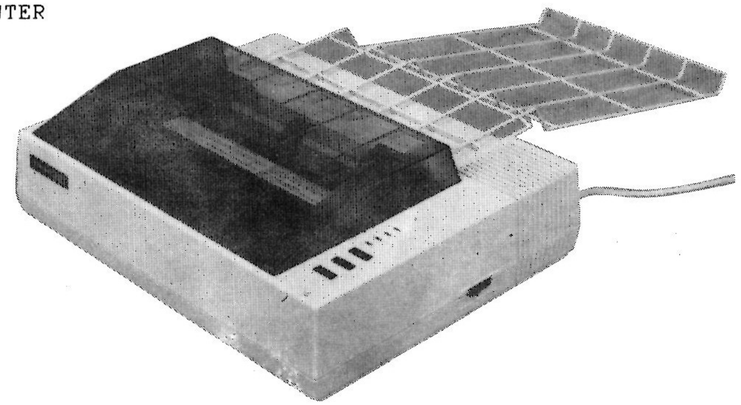
One final comment on the printer. When you purchase it, please remember you will need an Einstein compatible cable to connect the computer to the printer. In the case of a parallel printer, such as the TP1ØØ, Epson etc, this is catalogue number 22-8112-Ø in the Einstein product catalogue.

So that is the basic system complete; capable of running complex business packages as well as recreational software. In the next section we will go on to discuss optional extras you may wish to purchase when the need arises.

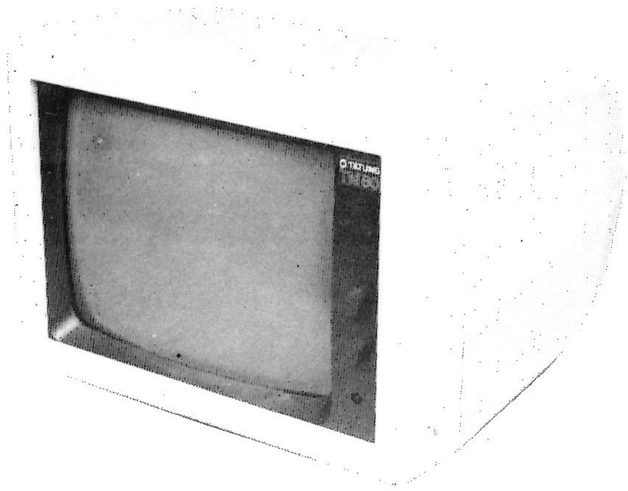
TATUNG TK02 80 COLUMN CARD



TATUNG TP100
DOT MATRIX PRINTER



TATUNG TM 80
MONOCHROME MONITOR



CHAPTER THREE

OPTIONAL EXTRA HARDWARE

DISK DRIVES

Probably the first item required by the normal business user, in addition to those previously described, is a third and fourth disk drive. This will allow greater storage of data and more complex programmes to be run.

The two external floppy disk drives can be obtained in three types:-

(a) 3" drives to match the existing internal drives.
(b) 3.5" drives
(c) 5.25" drives.

Probably the first choice would be the 3" drives. These have the advantage that all media is of the same type so that disks purchased or used in either of the internal drives will equally be compatible with either of the external disk drives.

3.5" disk drives could be chosen if you want to retain some extent of protection over mishandling, but find it difficult or expensive to obtain blank 3" disks. Personally, I see no advantage to connecting this type of drive to the Einstein.

5.25" drives have the advantage of using disks which are more economical to purchase than their 3" counterparts. They are, however, more susceptible to mis-handling.

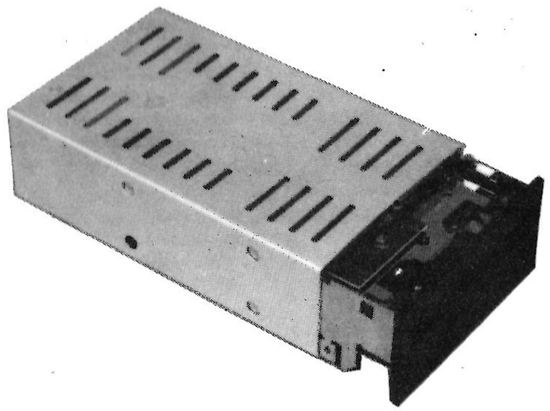
At this point it should be stressed that, just because you purchase a 5.25" or 3.5" disk drive you can obtain your programmes in this format. Software will still be available on 3" disk and it is the user's responsibilty to COPY it across to the other media. However, this process is very simple!

For normal use it is suggested that one or two external 3" disk drives be purchased. This will give the operator up to 752K of storage space available for programmes and data. For example, the operator could have their WORDSTAR programmes on drive 0, their WORDSTAR TEXT on drive 1 and the WORDSTAR SPELLING DICTIONARY on drive 2. Should they be sending out thousands of personalised letters using WORDSTAR MAILMERGE, then the data of customers' names and addresses could be on drive 3. This would mean that exchange of disks is not necessary.

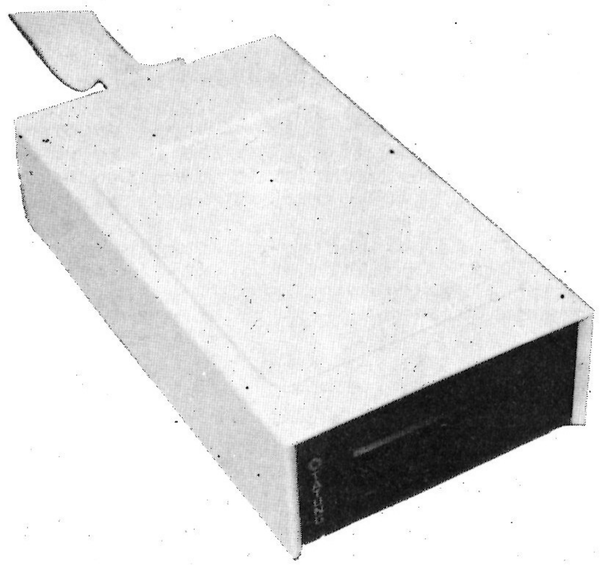
Tatung's external 3" disk drives (cat.no. TD301) has its own power supply and simply needs connecting to the external disk drive socket at the back of the Einstein and the mains lead to be connected to a convenient power point.

Sacrificing the standardisation of disks, a similar procedure may be undertaken by connecting either of the other two disk sizes to the Einstein but, when purchasing, please make sure that the supplier understands that they are to be connected to the Einstein computer and that the drives have their own power supply.

There is, currently, at least one supplier who can provide 5.25" disk drives for the Einstein allowing formatting of 80 tracks in double density. This requires an alternative operating system known as SYSTEM 5. Though not officially Einsoft approved, most



TATUNG
3" EXTERNAL DISK DRIVE WITH
POWER SUPPLY UNIT



TATUNG
3" INTERNAL DISK DRIVE

programmes will run using SYSTEM 5. The author has found both WORDSTAR PROFESSIONAL and THE CRACKER supplied for the Einstein are incompatible with SYSTEM 5, but other similar software such as SUPERWRITER and MULTIPLAN do operate. WORDSTAR will run under system 5 providing a user patch to the software is used.

System 5 has been reviewed in EINSTEIN USER vol 2 no. 1 and the first issue of the BRAIN. In the context discussed here, SYSTEM 5 allows the user to customise his or her operating system to use disk drives with greater storage capacity. For example, by altering one byte on the SYSTEM 5 disk (described in the manual) to HEX CC, you would have configured your new operative system to access two built-in 40 track single sided 3" drives and two 80 track 5.25" double sided drives as externals. This produces nearly 2 megabytes of on-line storage. Tatung are about to launch their own system for mixed/track drives, called system 80.

Is this storage capacity really necessary? Well, this depends on the specific purpose of your computer system!

Let us take the common application of stock control. A typical package may allow 8 characters to store a part number for the stock item, 30 characters for a description, 15 characters for the supplier of the part, 3 characters for the location or bin number, 5 characters for the current stock level, 5 characters for the reorder level, 5 characters for the reorder quantity, 6 characters for the year sold to date figure, 9 characters for the sale price, 9 characters for the purchase price and, say, another 25 characters for overheads. This means that 118 characters are required to be stored against each item of stock. Presuming you are using one 3" disk drive to hold the stock control programme and the other 3" to hold the data, you could store, theoretically, 1593 stock items on one disk. Invariably this is not the only factor; you may also wish to have sales ledger records and invoicing available. You would, therefore, require additional space for client details, discount codes etc. and this all eats up space; but the above theory still applies. The more stock items you have, the more disk space you require, the more applications you wish to run <u>at the same time</u>, the more disk space you will need.

Please remember that the above stock control programme was fictitious and you will have to ask your dealer or the software manufacturer, how much space is required for your application (ie. how many and what size drives are required for your application). It is always best to over-estimate than to get the message "disk full" in the middle of running your business programme on a Friday night.

Floppy disk drives are not the only drives available for your Einstein computer. There are also silicon disks and hard disk drives.

Silicon disks are unlike floppy disks in many ways. Firstly they do not store data onto a magnetic surface, but retain it in a silicon chip so that, unless there is a battery back-up, the data is lost when the power is switched off. Secondly, as there is no read/write head to scan the surface, data is stored and retrieved at a much faster rate. Thirdly, the units are not interchangable, therefore backups must be taken onto floppy disk drives.

The other type of drive is also of a fixed type and is known as a Hard Disk Drive. These differ from floppy disk drives in that they have a magnetic read/write surface of a much greater storage capacity and, therefore, can store many times more data (usually

around 10 megabytes). The disk drive also spins at a greater speed and the read/write head hovers over the surface; therefore data is transferred to and from the computer at a far greater speed allowing business programmes to function far more quickly. At the time of writing (1986) a typical 10 megabyte hard disk drive would cost around £1,000.00 and an 800K floppy around £200.00 so, hard disk drives are more economical if the initial outlay can be afforded.

As with silicon disks, hard disk drives are not interchangable and, therefore, another floppy disk drive must be utilised to take back-up copies. It would be very tedious to take a back-up copy of the whole of a hard disk drive onto the standard 3" floppy which only accepts 188k of data. It would take a great deal of time and around 53 disks to complete the tasks, therefore, a higher density 5.25" floppy is highly recommended.

Hard disk drives also make NETWORKING a viable proposition and, at present, one company provides a networking facility, with the Einstein. This makes use of either a 5, 10 or 16 megabyte hard disk drive system. With the networking facility, up to 31 Einsteins can share the same data and facilities; an ideal system for the larger office, where one terminal (Einstein) may be needed in the dispatch office, another in the accounts office and a third for use by the managing director. Files may be protected from unauthorised use and from illegal modifying. The system works like a ring-main, and the main computer system sends data around the offices by means of a cable and other Einsteins can plug into the cable at desired points. This means that only one hard disk drive and one printer is necessary.

MODEMS

After extra storage capacity, the other most popular accessory to the Einstein is a Modem.

Modem stands for modulator/demodulate, and there are two types of modem, the acoustic coupler and the hard-wired. Actually, only the hard-wired is really a complete modem and the acoustic coupler is only a hand set converter, but the term has become common to both types.

Modems can cost anything from around £50.00 to many hundreds of pounds, the most usual price for the Einstein being around £100.00 to £150.00.

Both types send information to and from other computers over a telephone line. Both computers must be connected to the telephone line via a modem. The digital signal created via a computer is turned into analogue noise by the modem and this passes over the telephone line. The signal is transmitted at a speed known as a baud rate and there are two main rates for domestic use. These are 75/1200 and 300/300. The first figure is the rate at which the modem will transmit data and the second is the rate at which data is received. It is not necessary, at this level, for the user to understand how this data is transmitted, just to understand that the modem chosen should be capable of transmitting and receiving data at the required rate for the services you wish to communicate.

The current standard baud rate for Prestel is 75/1200 and for User Group bulletin boards it is 300/300 though this can vary. Many magazines, such as Personal Computer World, publish such information and telephone numbers.

The modem plugs directly into the modern telephone point in the home and the related device, the acoustic coupler, links the computer to the telephone directly via the handset. This connection is normally using the older type handset and the new "trimphone" style does not fit the acoustic coupler.

Two of the most common areas in which the modem is used is for PRESTEL and the passing of ELECTRONIC MAIL.

Prestel is British Telecom's videotext service, available in your home or office over the British Telecom telephone line. You connect to Prestel's main computer which holds details of information such as rail timetables, Theatre agenda, computer software, news, sport and weather. It also enables users to leave messages commonly refered to as "logging on", to be accessed by other users when they in turn communicate with Prestel. Further details of the Prestel service can be obtained by dialing the Telephone Operator and asking for FREEPHONE PRESTEL SALES.

Electronic mail is available from a similar number of sources, the two most common being TELECOM GOLD and ONE-TO-ONE.

Both these companies provide a similar service, but I shall briefly describe One-to-One.

The One-to-One service is based on a central main frame computer which receives, sorts and despatches computer mail to the recipient whether or not they have a computer linked to the system, a telex machine or a Postal address.

Having subscribed to One-to-One, all you need is a modem and some suitable Communications software. There are packages available and your dealer will be able to advise you on the most suitable.

Having prepared your message on your Einstein, it can be sent direct from the Einstein to the One-to-One computer, via a simple telephone call. The sender specifies if the recipient is a One-to-One subscriber, has a telex or if it is to go by normal mail or courier.

Unlike ordinary mail, electronic mail can be circulated to a number of recipients by specifying this and only typing the letter into the computer once.

As with Prestel, One-to-One is Password Protected for security, so that unauthorised use is eliminated.

JOYSTICK

It is not compulsory to have a joystick to play games on the Einstein computer, in fact some computer games do not have the joystick option, but, in most cases it makes the game easier to operate and more fun to play.

There are a number of joysticks and you should make sure the one you receive is specifically for the Einstein. One way is to ask for EINSOFT APPROVED joysticks and the current ones with this approval are made by COOKRIDGE COMPUTERS.

A single joystick plugs into the analogue 1 port on the right-hand side of the computer when viewed from the front.

OTHER PERIPHERALS

Sound Box

EINSTEIN PRIMER

Penman Plotter
Dust Covers

PART TWO

POPULAR SOFTWARE

The purpose of this section is to introduce the Einstein User to some of the more popular pieces of software and briefly explain their capabilities. This section is not intended to replace the instruction manual of the software itself, nor the advice of your dealer, but to show the wide range of uses the Einstein can accommodate.

In the main, the software will require the 80 column card to be fitted to the computer and to be active when the software is run. Some software will also require more than one disk drive and where necessary, this will be mentioned.

WORDSTAR PROFESSIONAL

Commonly quoted as the most popular word processing package, Wordstar Professional requires the 80 column card to be fitted and it is strongly recommended that at least two drives are used.

The price of this package tends to vary and there are frequent special promotions offered by your dealer.

You are advised to ask for the version of Wordstar Professional which has been "pre-configured" for the Einstein, unless you are familiar with the screen handling of your computer or you have a dealer willing to do this for you.

The supplied Einstein configured WORDSTAR PROFESSIONAL comes installed for the Tatung TP100 dot matrix printer, but by running the WINSTALL.COM programme supplied, this can be altered. There has, however, been some confusion over running this install programme, as it should only be run on a copy you have made on the programme. Use the COPY programme on your system master disk to copy the following programmes onto another blank formatted disk:-

WS.COM
WSMSGS.OVR
WSOVLY1.OVR
WS.INS
WINSTALL.COM

Place this new disk into drive 0 and enter WINSTALL from the keyboard.

After a short while a copyright message will appear, press the <ENTER> key and Enter Y to the next question if you wish to proceed.

You are installing Wordstar, so enter WS to signify this at the question ? prompt.

After a few introductory statements you are now presented with the message "Enter the disk drive name (a letter followed by a colon....."

Providing you have inserted your copy Wordstar disk into the left hand drive you can enter A: here.
The next prompt asks you to enter the name of the wordstar programme. Enter WS.COM here and you will be asked to state the name of the programme to be called at the end of the procedure, I suggest that you still call it WS.COM.

You are given an option to change your mind, then you are presented with a list of installations you may carry out.

We will only concern ourselves with the simplest option, to install a printer that Wordstar already knows! Press option C to see these printers. If your printer is not listed here, press 2 to see the next list. We will assume that you are installing Wordstar for an Epson MX80. In this case press G and you are asked to confirm your selection.

Communications protocol is the next stage and this is not complicated. The computer sends data to the printer much faster than most printers can process, so it needs to know when to stop sending data and wait until the printer can take more. A number of printers including the TP100, sense this, via the printer cable and no Protocol is required, as it is handled outside the Wordstar package. You will need to consult your printer manual, but if in doubt <u>try</u> A for NONE. You cannot do any harm to the computer or printer.

In most cases the next option for the Printer Driver will be A or Operating System Primary list device.

You have now finished installing a different printer and you may, if you wish, select option X from the Installation Menu and return to DOS. I would suggest, however, that you examine option E where you can alter the default Top and Bottom margins, page number, left and right Margins etc.

Well, Wordstar professional is not the least expensive piece of software, so, is it worth it?

First, let's get one thing clear, it is not just a conventional word-processor. It includes a Spelling checker with 20,000 or so words already included and more can be added by the user. It also includes mailmerge, which enables the user to set up a list of, say, names and addresses, then create one letter and the computer and printer will personalise that letter to each of the people in the name and address list. Wordstar Professional also includes Starindex which creates reference aids for use in reports etc. Such things as an index or table of contents are reference aids.

Let us get started by creating a simple letter, but, please remember this is not a tutorial but a <u>small</u> taste of what is available in Wordstar Professional.

After inserting your working copy of Wordstar Professional into the left hand disk drive, (and a blank formatted disk into the right if you have one), type WS and press the <ENTER> key. After a short delay while Wordstar is loading the first part of the word processing programme, you are presented with the OPENING MENU or "NO-FILE MENU". These menus are shown at each stage (unless you opt not to show them) and prompt you to the most usual commands that may be used at each stage. If the words "NO-FILE" MENU seem strange it is because you have yet to chosse a document or FILE to work on.

In addition to the list of commands you can make at this stage, is a DIRectory of the files that are recorded on the disk in the left-hand drive. Lets see what is on the right-hand drive (if you did not put a formatted disk into the right-hand drive, DO NOT use this next command). You will notice that the first option on the OPENING MENU is "L Change logged disk drive". The logged disk drive is the drive that the computer is currently using to read or save data and this is currently the left-hand or drive A. (Note: drives are numbered A, B, C or D under CP/M and 0, 1, 2 or 3 under XTAL DOS, in this example you are running a CP/M programme.) If you now press B: you will log onto the right-hand disk drive or drive B and if there are any files on this disk, then these will be displayed on the screen in place of the earlier directory. (Note: if there is no drive B or no disk in drive B you will see:-

```
Not Ready 1:
Try Again (Y/N)?
```

displayed on the screen and you will have to re-load WS after pressing the N key).

If you are logged onto drive B press L again and log onto drive A by entering A: and press the <ENTER> key.

To create a sample letter, open a document file by entering the letter D (a Non-document file would be used to create a list of Names and Addresses or a programme written in Pascal or Dbase II).

After entering the command D you are asked for the name of the file you wish to edit. This could be the name of an existing file on the disk or a new one. We will create a file called LETTER so enter those characters and press the <ENTER> key. The programme will search to see if that file already exists on drive A and, if not, will display the words "NEW FILE" and then clear the bottom half of the screen and change the top part of the screen to show the MAIN MENU. This menu shows the main ways of moving the cursor around the screen, how to delete characters and words and other commands. On the right of the Main Menu you are shown further menus that may be called up for other tasks.

Underneath the MAIN MENU is a line showing you where the current paper margins would be and the positions of the tab stops. For the moment we will forget the margin and type in some text. Change the text from the computer to lower-case by pressing the ALPH LOCK key and type in this paragraph of text. You do not have to press the <ENTER> key when you get to the end of the line, only if you wish to start a new paragraph.

As you are typing you will notice that the word processor automatically fitted the text neatly onto the screen with only complete words at the end of the line. This is called justification, and differs from some other word processors, such as Kuma's WDPRO. Should you have wanted to start a new paragraph, you would have pressed the <ENTER> key and on the screen, the cursor would have moved down a line and waited for you to type in further text. I call this "on screen formatting" and shows you on the screen what you will get on the paper when the document is printed, often termed WYSIWYG-what you see is what you get. Obviously the screen cannot show all the print instructions exactly as they will appear in the finished document. The screen is incapable of showing letters underlined and these will have ^S either side of the letters to be underlined and letters to be produced in bold typeface will appear on the screen surrounded by

^B.

You have now typed in a short document and you now wish to save it to disk. There are a number of ways to save documents to disk, but the two main ones are <CTRL>KS and <CTRL>KD. (From now on we will symbolise holding the <CTRL> key down by the character ^). ^KD will save the file to disk under the file name entered when you began the session and then return you, to the OPENING-FILE MENU. ^KS will save the file to disk but you will remain in the text editing mode and the cursor will be at the beginning of the text.

Before you can perform any editing on your text, you need to know how to move the cursor around the document. If you look at the help messages displayed at the top of the screen, you are reminded that ^A moves the cursor a word to the left ^S moves the cursor character to the left etc. You will soon remember these simple controls and not need to look at the top of the screen, though it is always there to remind you.

Deleting characters - words and lines are equally easily performed. Just place the cursor over the character you wish to delete and press ^G and the character will not only disappear, but the rest of the text to the right of the cursor will close up to fill the gap. To delete a whole word, you may either repeat the above procedure for each letter or place the cursor on the first letter and press ^T. Deleting a line is accomplished with a similar procedure, but pressing the ^Y keys instead. When you have finished deleting in a paragraph, position the cursor anywhere prior to the first deletion and press ^B, this will re-format the paragraph to have a neat right-hand justified margin.

So much for the common delete procedures, inserting is just as simple. Position the cursor where you want to insert the character or words and then press ^V. You will notice the words INSERT ON appear at the top right of the main menu; if this does not appear press ^V again. Any characters you now type in, will be inserted at the position of the cursor and the rest of the characters to the right will be moved across. When you have finished inserting text, ^V will turn the insert mode off, but the line of the text you have typed may not have a neat right hand margin. Use ^B to re-aligne the right margin and this will act on all text from the cursor position up to the end of the current paragraph.

You now know enough to write a document, edit and save your document or file to disk. What you now need to do, is to obtain a printout of your text from your printer.

Save your document to disk and return to the OPENING MENU by pressing ^KD. A document, once saved to disk, is known as a file and you will notice that one of the options you can perform on your files is to PRINT a file.

When you are in the OPENING MENU you do not need to hold the <CTRL> key down so, making sure that your printer is connected, turned on and on line and that paper is loaded, press the P key to load the PRINT a file option. You will first be asked to enter the name of the file you wish to print. This could be the one that you were just typing in or any listed in the directory. For now, type in the name of the document you have just saved to disk (LETTER) and press the <ENTER> key. The programme will check to see if that file exists and if it does will ask you a series of questions. For this tutorial I will ignore the meaning of the

questions and just ask you to press the <ENTER> key to each question or the <ESC> key to ignore all subsequent questions. Providing your printer is correctly set-up, you should now see your document being printed out with neat margins just as it appeared on the screen as you were typing it in. Once the document has been completely printed you are returned to the OPENING MENU for further operations, or you may press the X key to end the session and return to DOS.

There are a great number of extra features in WORDSTAR PROFESSIONAL and I will only try to point out some of them, leaving the technical operations to achieve these features to the manuals that accompany the programme.

You may define any part of your typed text as a BLOCK and perform a number of tasks on that block of text. For example you may:-

MOVE the whole block of text to another part of the document.
COPY the block to another part of the document leaving the original in place.
WRITE the block of text as a separate file to disk for later use.
READ in a previously stored block of text.
DELETE the whole block of text.

You may produce characters which are underlined, printed in bold text or centered on the screen, or a combination of all three. You may search for and replace words automatically or, just find specified words in the text.

Using the extra features of WORDSTAR <u>PROFESSIONAL</u> you may produce personalised letters to a number of people by just typing one letter and READING IN a list of names and addresses you have previously created using the NON-DOCUMENT feature of WORDSTAR.

Spelling checking is relatively simple, but the dictionary included with the package is American and you may wish to delete references to, say, COLOR, and replace these with the English spelling. This is done via the Dictionary Maintenance option when running the spelling checker.

Slightly more complicated to use is the INDEXING feature and this is best left for a later date when you are more familiar with the WORDSTAR programme.

<u>DBASE II</u>

Dbase II is a relational database able to run on a single drive Einstein with an 80 column card, but a further disk drive would be required for any large databases. It is one of the best selling and more sophisticated databases available on any micro computer.

Before I describe this product, what is a database? A database is simply a list of information which is stored in a row and column format. An address book is an example of a database. One column would have the surname, another the forenames, a third the address and a fourth the telephone number. In a database, these columns are known as <u>FIELDS</u> and each entry in the address book would be known as a <u>RECORD</u>. To find the telephone number of, say, Joe Smith, you would look up the name Smith in the first "Surname" field. If there was more than one Smith, you would continue searching until the second or "forename" field gave Joe, you would then be able to read the last or "phone number" field to give his telephone number.

This is a very simple example, but even this gets more difficult if the names are not in alphabetical order and there are hundreds of entries. This is where a database excells. It can perform the task of searching for a record at very fast speeds.

You insert English words to create and command your database. As with the Wordstar section, I will give an easy example.

The disk containing dBase II is inserted into drive Ø and the system loaded. To commence using dBase II you enter the command dBase ‹ENTER› into the computer. After a short while you are prompted for the current date, which, if the programme was purchased from a Tatung Dealer should be in British format of DD/MM/YY. You can enter the date here and the programme will remember it, but, alternatively, if you will not be using the date within your programming, you may just press the ‹ENTER› key.

At this point you are presented with a copyright notice, the line:-

Type 'HELP','HELP dBASE', or a command

and a period on the next line. The period is the dbase program way of telling you that it is waiting for you to insert text.

Entering HELP or HELP dBASE will bring onto the screen a number of quick 'Help' messages, giving information on the demonstration programs, the install programmes plus other useful information, but this is no substitute for the rather hefty dBase manual supplied with the software.

A command is a simple word that dBase understands such as SORT, FIND or LIST. Should you forget the actual way of using these commands, you may remind yourself by entering HELP and the command word such as HELP LIST.

Before you can begin to use your database for your own purposes, you must create the structure of the records. This is to store the names of the fields the data is to be stored under. To take our earlier address book example we would need to create the fields SURNAME, FIRST, ADDRESS and PHONE. This is a simple process and is created by entering the command CREATE ‹ENTER› after the period prompt.

CREATE‹ENTER›

DBase II is capable of having more than one database stored on disk at any one time. In fact, as we shall see later, this is one of the powerful features of dBase II. The next prompt asks:-

ENTER FILENAME:

You must now give a unique name to your database, let us use the name ADDRESS and this is what you would enter here.

After pressing the ‹ENTER› key, you are asked for the structure of the database. This is the FIELD names, whether they can contain characters or just numbers or even just T or F for True or False and the maximum number of characters the field could contain. If the entry is only numeric, you would also indicate the number of decimal places required in the field.

You may store up to 32 FIELDS in a single database. (But, as I stated earlier, you could have more than one database stored on disk). Each record can be made up to a maximum of 1,000 characters and you may store up to 65,535 records (ie if you have

enough disk space) in each database.

After entering the name of your database, you are asked for the name of your first FIELD. As well as naming the first field (ie SURNAME) you are asked to state whether it contains Characters, Numbers, or a Logical T or F (True or False). You are now prompted to enter the width. This is the number of characters this particular field may contain (when entering the width of a numeric field you must make allowances for the decimal point taking up the space of a character). Once you have entered this, you will be asked for the second FIELD name and so on. Just hitting the ‹ENTER› key instead of a field name will terminate this section and you will be asked if you wish to enter your data at this point. If you wish to do so, simply press the Y key and then ‹ENTER›. You may now enter the data against each of your specified fields and press the ‹ENTER› key at the end of each field entry. Once all the entries have been made, to record one field the programme automatically displays the blank fields for you to enter record two, and so on. Once you have entered all your records, you just have to press the ‹ENTER› key instead of entering data in field one you then return to the period prompt.

In its simplest form, this is how you CREATE the structure for your dBase II database and enter the data. The other dBase II commands are equally clear;

LIST will list to the screen all the records in the database.
BROWSE will list a screen-full of records at a time.
EDIT will allow you to modify a specified record.
APPEND will allow you to add records to the database.
DISPLAY STRUCTURE will allow you to view the structure of the database which you have CREATEd.

This is only a very few of the simple "English" commands which this program uses, but, I feel that the programs full power is not in its vast and helpful range of commands, but in its built-in COMMAND LANGUAGE.

The COMMAND LANGUAGE allows you to construct small programmes of dBase II commands and then execute these commands automatically under given circumstances. An example of simple command programmes could be as follows-

```
ERASE
USE ADDRESS
LIST FOR SURNAME = "SMITH"
```

This would clear the video screen; use the database called ADDRESS and list to the screen all the records where the surname is Smith.

Such a simple command program would hardly ever be used and actual command files normally ask for input from the user and act accordingly. By using such command programs the ultimate user of dBase II need never know they are using a database, to them they could be running s STOCK CONTROL, PAYROLL or similar program. There is almost no limit to its possible uses.

THE CRACKER

Another very well used programme in the world of computers is THE SPREADSHEET. This type of programme allows the user to create 'what-if?' situations. A typical example of this would be a financial forecast. There are many such programmes available for the Einstein and range in price from the very simple KUMA

SPREADSHEET to the much more expensive SUPERCALC2. I have chosen an economical package called THE CRACKER. Though modest in price, it does carry a great number of the features of the more expensive packages.

The Cracker will work in either 40 or 80 column mode and only requires a single drive Einstein. The current version will not run under the SYSTEM 5 operating system, its use being restricted to the DOS supplied with your computer.

THE CRACKER allows your computer screen to act like a large sheet of graph paper which has a built-in calculator. You may enter numbers or formulas anywhere you wish on the paper and link them to other locations. These locations are identified by giving the columns a letter and numbered rows. So, a spreadsheet of 5 columns and 7 rows would have columns letter A,B,C,D and E and the rows 1,2,3,4,5,6 and 7. To identify the cell in the centre you would address location C4.

The first task, once you have loaded THE CRACKER into your computer, is to set-up the area you wish to work. This will normally be larger than can be displayed on the computer screen, but it WILL exist in the computers memory and you may scroll around the area of your work-space at will.

The commands are reasonably logical and you are prompted at the top of the screen as to which commands are acceptable at every stage. The prompts are normally limited to a single letter or symbol, but are very soon learned.

Text, numbers or formulas may be entered into the cells and as stated earlier, cells may refer to one another. For example, cell A1 may contain the number 12.25 and cell B1 may contain the formula A1+7. Whenever the contents of cell A1 are altered, cell B1 will show the sum of A1 plus 7. This is a very simple example, but if you imagine a very large cash projection for a small business, over a period of a year, calculating percentage seasonal changes in sales, or the overall cost of producing an item if certain raw materials were purchased at a 7.25% increase in cost, then you will see the power of the spreadsheet. Calculating the same thing by hand would take a long time and probably a great deal of erasing. Using the spreadsheet, different figures may be tested with almost instantaneous results.

There are a number of extra tasks that may be be done with THE CRACKER that are unusual to some other spreadsheets. Firstly, there is a facility for producing mailing labels, so, if your spreadsheet was being used to keep the accounts of your local club, you may be able to produce labels giving the name and address of each member, without using a separate database programme.

Another unusual function is the DO and WHILE function. This would allow you, for example, to set up a loop to test a certain formula, until a previously defined result was obtained.

The spreadsheet is much more difficult to describe than it is to use. The logic of its use is very simple, the expertise of its use comes with thinking out the problem logically.

PART THREE

PROGRAMMING IN XBASIC

The idea behind this section is to introduce the reader to the basics of programming in TATUNG/Xtal BASIC 4.2.

To use this section of the EINSTEIN PRIMER, you will require the Einstein colour monitor or a domestic TV connected to your computer. An 80 column card may be left attached to the computer, but the video signal must not come from the 80 column card.

This section will not be an in-depth programming course, there are other books available on this subject, but, as the title infers, this book is meant as an introduction or primer.

The Einstein computer, unlike most other computers in the lower price bracket, does not have a BASIC language resident in its memory all the time. This has the advantage of not using up valuable memory when it is not required for BASIC programming. It also has the advantage of allowing revisions or upgrades to be used when available. A further advantage is that you are not limited to TATUNG/Xtal BASIC, you could, if you so wished, use MICROSOFT BASIC or Digital Research CBASIC.

The computer language BASIC is the world's most commonly used computer language, because it is fairly simple to understand. However, like the natural languages, it has grammatical rules which must be followed.

BASIC stands for Beginners All-purpose Symbolic Instruction Code and was invented in the United States of America in 1964 and is one of the programming languages for modern micro computers.

Programmes instruct the computer what to do and in what order sequences should be carried out. BASIC is also known as a High-Level language; that is to say the BASIC instructions we write are converted into Machine-code (a Low-Level language) that the computer understands whilst the programme is running. This is what makes BASIC programmes slower to execute than some other languages such as Machine Code.

To commence programming in TATUNG/Xtal BASIC (from now on referred to as BASIC), place your system master disk, or one of your legal back-up copies, into the left-hand drive of your Einstein computer and reset the machine. When you obtain the 0: prompt, enter the command:-

XBAS<ENTER>

and wait for a short while whilst the program loads into the computers memory. When this is accomplished, you will see the following message displayed:-

```
TATUNG/Xtal BASIC 4.2                          (C) 1983 1984
Size:  43324
Ready
```

You may see a different BASIC version number, if this is so, do not worry, continue with this tutorial, but contact your dealer or Tatung when it is convenient for you to do so, to upgrade your System Master.

The numbers displayed after the Size: message, is the amount of bytes available for your BASIC programme. These are 1024 bytes

per 1K and if you compare this with some other machines, you will see the Einstein has a great deal of memory avaiable for the BASIC programmer.

A BASIC program is simply a list of instructions to the computer in a specified order. The following is a simple BASIC programme:-

```
10 CLS 40
20 INPUT A
30 INPUT B
40 PRINT A+B
```

You will notice that there are a series of numbers at the beginning of each line of the program. It is standard convention to commence your program by numbering each line in increment's of 10. This is to enable you to insert further lines into the programme at a later date. If the program had been written:-

```
1 CLS 40
2 INPUT A
3 INPUT B
4 PRINT A+B
```

it would have executed just as well, but we would have had a problem if, at a later date, we had wanted to insert a line between 2 and 3 to say PRINT "NOW ENTER A SECOND NUMBER", but in the first example we could have added a line at 25 thus:-

```
10 CLS 40
20 INPUT A
30 INPUT B
40 PRINT A+B
25 PRINT "NOW ENTER A SECOND NUMBER"
```

It does not matter, when programming, if you have entered the extra or additional line at the end of the program. The computer would run the program in line-number sequence and if the program was listed or saved to disk, it would be in the correct numerical line order. Try the above. Enter the 5 lines as shown above, ending each line with the <ENTER> key and then type in the command RUN. Do not put a line number in front of the word RUN as this is a direct statement and not part of the program.

As a result you will be prompted with a ? mark and the flashing cursor. At this point, you should have entered a number such as 7. The next line on the screen should have shown your PRINT statement NOW ENTER A SECOND NUMBER and you should have seen the ? on the next line waiting for another number to be entered. If you had entered 3 here, the next line on the screen should have shown the sum of the two numbers 10. The final line shows the word Ready indicating that the computer has run your BASIC program and is waiting for you to enter another command. The program is, however, still in the computers memory and you could run it again by entering the command RUN. This is something to be born in mind as, if you wish to write another program you should clear the first program from the computers memory by inserting the command NEW and then press the <ENTER> key.

This program was just an example to show the convention of using line numbers in increment's of 10 and we shall now go on to teach you something about programming.

Each line or statement of the above program instructs the computer to do something. Line 10 above instructs the computer toCLEAR THE SCREEN and put the computer into 40 column display.

We could have quite easily have stated CLS 32 here and the program would have run just as well though the text would have been larger on the screen being only 32 columns accross.

Line 20 asks the computer to wait until the user has entered a number from the keyboard. The computer will continue to wait until you signify that you have finished entering the sequence of numbers by pressing the <ENTER> key. You may enter any number or a decimal point. You must NOT enter a comma or any other character. These numbers will now be stored in a part of the computers memory. The computer locate's these numbers again by assigning them a label known as a VARIABLE, in our example of line 20 this variable is A.

Line 25 just asks the computer to print on the screen all the characters within the quotation marks so the words NOW ENTER A SECOND NUMBER will appear on the screen.

Line 30 is similar to line 20 but we have had to assign a different VARIABLE so as not to confuse the two. (If you had used the same variable 'A' line 30 would have destroyed your input in line 20).

Line 40 asks the computer to print out on the screen the numbers you had stored under variable 'A' plus the number you had stored under VARIABLE 'B'. As the words A+B were not enclosed in quotation marks, the computer knew not to print the characters A+B but printed the contents of the variable 'A' plus the contents of variable 'B'.

A VARIABLE can be made unique by a combination of letters and numbers but it must always start with a LETTER. The variables we have shown so far are known as numeric variables as they hold a numeric value. The other type of variable is a 'STRING' variable and this is distinguished from a numeric variable by having the $ sign entered after the variable. A string variable could be A$ or AB$ or even name$. The one thing that you must watch out for is not to have a variable name that is the same as a BASIC command. You could not, therefore, have the variable name PRINT for example. These commands are known as the BASIC RESERVED WORDS and a full list is given, together with their meanings, in the TATUNG "BASIC REFERENCE MANUAL" commencing on page 31.

We know how to store a number into a numeric variable but how do you store a string of characters into a STRING VARIABLE?

Easy! You just surround the characters by quotation marks. For example:-

```
25 C$ = "HELLO DAVID!"
27 PRINT C$
```

This short program would store the character string HELLO DAVID! into string variable C$ and at the next line of the program, print the contents of the string variable C$ on the screen.

Before we leave variables it should be mentioned that there are two types of numeric variables. FLOATING POINT which holds numbers and INTEGER which only holds whole numbers. Floating point is the default state and is the numeric variable that we have discussed so far. Integer variables have a % suffixing their variable name.

```
A$ = STRING VARIABLE                    eg "TOM BROWN"
A  = NUMERIC (FLOATING POINT)           eg 100.25
```

A% = NUMERIC (INTEGER) eg 100

Line 20 of our program waited for us to INPUT a number from the keyboard. We could, if we had wished, instructed the computer to wait for two seperate sets of numbers. We could have done this with the following statement.

```
20 INPUT A,B
```

Line 20 would have caused the computer to wait until we had input our first number and signified this by pressing the <ENTER> key. The program would have stored this in the variable 'A' and when the second number had been entered, this would have been stored in the variable 'B'.

We are not limited to two variable inputs but the variables must be seperated by commas.

When the INPUT statement is encountered, the computer indicates that it is waiting for data to be entered via the keyboard by displaying a ? on the screen, but this is not very explicit. The person using the program would not know if they were to enter numbers or a name etc., and entering a name instead of numbers would prove disasterous to the program efficiency.

This lack of any sensible input prompt can be corrected by inserting a prompt string in front of the variable in the INPUT line. To give an example, this is how line 20 of our programme could have been written.

```
20 INPUT "ENTER YOUR FIRST NUMBER ";A
```

When this is run you will see the prompt ENTER YOUR FIRST NUMBER on the screen whilst the computer is waiting your input. Please note that the string is surrounded by the quotation marks and is followed by the ; symbol.

Strings of characters are also displayed on the screen with the PRINT statement. Our example, in the earlier program, was at line 25 with the statement PRINT "NOW ENTER A SECOND NUMBER". Again, anything enclosed within the quotation marks will be printed onto the screen whether it is letters, numbers or symbols. If we had used the statement PRINT "A+B" we would have seen A+B printed on the screen as these are the letters and symbols enclosed within quotation marks.

In BASIC we use special symbols, called operators, to indicate the arithmetical operations of addition, subtraction, multiplication, division and exponentiation. We have already used one of these, the + meaning addition. The full list is as follows:-

Addition	+	(plus sign)
Subtraction	-	(minus sign)
Multiplication	*	(asterisk)
Division	/	(back slash)
Exponentiation	↑	(upward arrow)

These operators are used to connect numbers and numeric variables, thereby forming formulae or expressions.

It is important to understand the hierarchy of the operations, as the wrong answer can easily be given. Take the simple example of 7 minus 4 multiplied by 2. To calculate this in your head would give the answer 7 minus 4 gives 3, multiply 3 by 2 gives the

answer 6. The computer, however would calculate this entirely differently. It would first multiply 4 by 2 to give 8 and then subtract 8 from 7 to give -1 a very different answer.

The order of preference given to operators discussed so far, are as follows:-

HIGHEST PRIORITY

↑	exponentation
*	multiply
/	divide
+	addition
-	subtraction

LOWEST PRIORITY

So you will see from our example, the computer gave the highest priority to the multiplication and calculated this first and then performed the subtraction using the result from the first calculation.

In addition to the above, here are two further operators () or parenthesis and MOD menaing MODulus arithmetic or remainder.

The parenthesis overcomes this problem as they are given the highest priority. Anything enclosed within parenthesis are calculated first, so, in our previous example, if the formula had been written:-

```
(7-4)*2
```

the computer would have performed the calculation 7-4 first and found the answer 3, it would have then multiplied 3 by 2 to give the correct answer 6.

If there is a duplication of operators ie. two sets of parenthesis or two multiplications then the calculations are carried out from left to right in order of precedence.

The MOD operator gives the integer value that is the remainder when an integer division is performed. Therefore 7 MOD 4 gives the answer 3 as 7 can be divided into 4 once and gives the remainder 3.

The full order of preference is:-

HIGHEST PRIORITY

()	parenthesis
↑	exponentation
*	multiply
/	divide
MOD	remainder
+	addition
-	subtraction

LOWEST PRIORITY

Before we experiment with another small program, here is one further matter to discuss concerning the PRINT statement.

The PRINT statement is used to display numerical and string output from the computer as we have seen in previous examples. However, in these examples we have only displayed one variable or string at a time. We could, if we wished, display a number of variables with a statement such as:-

```
10 PRINT A,B,C$,D
```

This statement would print the contents of numeric variables A, B and D together with string variable C$ on one line. The comma will set each variable at a preset TAB stop, which is at 10 space intervals across the screen. Providing the variables do not contain more than 10 characters or numbers, we can place 4 variables across the screen in the 40 column mode.

We are not limited to variables, we could mix strings and variables as in the following example:-

```
10 PRINT "THE NUMBER IS",X
```

In the above example, if we do not wish the output to be placed at set tab stops across the screen, all we have to do is remove the comma after the string.

```
10 PRINT "THE NUMBER IS"X
```

Obviously, we could not do this with numeric variables, as there would be ambiguity over where the end of one variable was placed and the beginning of the next. We could, however, use string variables as the $ sign would signify.

Let's produce another simple program to put together what we have learned so far. Remember to enter the command NEW to clear any exsisting program from the computer's memory before typing in the next program.

```
10 CLS 40
20 INPUT"WHAT IS YOUR FORENAME? ";A$
30 INPUT"HOW OLD ARE YOU IN YEARS? ";B
40 PRINT"WELL "A$" YOU ARE APPROXIMATELY"B*365" DAYS OLD"
```

The first thing we notice as we type this program into the computer is that line 40 will not fit on one line. Don't worry, just keep typing, you are allowed to type 126 characters before you need to start a new program line. More than enough for our examples!

Another thing we will notice, is that, we do not know how many letters will be in the forename, we cannot therefore, be certain what the finished display will look like. There are ways to overcome this with reserved words such as LEN, but that is for later explanation.

Let us analyse what our program will do.

Line 10 is simple, it clears the screen of any text and sets the computer to 40 column mode.

Line 20 prompts you to enter your forename and stores this in the string variable A$.

Line 30 asks you for your age and stores this in the numeric variable B. As we did not state that the numeric variable is an integer (%), we could have entered the number with a floating point.

Line 40 displays the calculation in readable form with the words WELL followed by the contents of string variable a$ (your forename), this is followed by the words YOU ARE APPROXIMATELY as these are enclosed in quotation marks. The program then displays the result of the simple calculation, taking the contents of numeric variable B and multiplying this by 365 (age multiplied by

365 days in a year). The program finally states that you are DAYS OLD.

As the demonstration programmes become more complicated you are advised to study section 7 (page 20) in the TATUNG EINSTEIN BASIC REFERENCE MANUAL. This covers the Editor and will allow you to correct any errors in the program without re-typing the whole line again.

RELATIONAL OPERATORS

So far, the programmes the computer has executed have not been very remarkable. All they have done is to do simple arithmetic and to display words we have given it, on the screen.

However, the Einstein computer is much more sophisticated and has the ability to make logical decisions and carry out specified instructions based on those decisions.

Decisions are made by the computer, based on Relational Operators and they are normally used when LOOPING within a programme or controlling a JUMP to another line in a programme. Let's take another simple example:-

```
10 CLS 40
20 REM counting to 100
30 X = 1
40 PRINT "STARTING"
50 IF X = 100 GOTO 80
60 X = X + 1
70 GOTO 50
80 PRINT "FINISHED"
90 END
```

One or two new things are introduced here. In line 20 we have the reserved word REM. This is short for REMark and, anything written after this statement is ignored by the computer. It is only treated as a comment or reminder to the programmer.

Line 30 could have read LET X = 1 (the LET being optional) and all this does is stores the number 1 to the numeric variable. This is similar to our earlier INPUT A, except for the fact that the computer is storing the number instead of us inputing this from the keyboard again.

Line 50 uses one of the relational operators. It tests the numeric variable X and, if it equals 100 then the program executes the remainder of line 50. If it does not equal to 100, then the programme continues with the next programme line, which is 60.

Line 60 merely adds 1 to whatever the total contents of numeric variable X.

Line 70 forces the programme to go to programme line 50 and continue from there.

All these lines will be carried out until line 60 has increased X to a value of 100. When this happens, line 50 will sense that X equals 100 and forces the program to jump to line 80 where the word FINISHED will appear on the screen and the program will end.

It is not strictly necessary to have the reserved word END at the end of a programme, but it is good practice, especially as the end of a program may not be the last line of the programme!

This short program showed just one Relational Operator, there are, in fact, six.

=	Equal to
<>	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Line 50 in the above program could easily have had > 100 or >= 100 and it would have worked equally well.

Relational Operators do not act only on numbers or numeric variables. You can compare strings with a similar command, but you must remember that you are acting on the ASCII value of the letters making up the string. The comparison:-

```
10 INPUT "ENTER SURNAME "; N$
20 IF N$ = "SMITH"
```

will seem obvious, but this would not see "Smith" as the same string, as some of the letters are in lower case. Similarly "A" is < "a" because the ASCII code for "A" is 65 and the ASCII code for "a" is 97. If you want to see a full list of the ASCII values, they are printed in the TATUNG EINSTEIN BASIC REFERENCE MANUAL on page 3 and 4.

BRANCHING

In our earlier example you will notice that line 70 has an unconditional instruction GOTO 50. If the programme encounters line 70, it has no choice but to GOTO line 50. However, the GOTO in line 50 is conditional to the value of X being 100 and the GOTO will only apply if that condition is true, ie X being equal to 100.

We can, however, expand this statement and force the program to compare two values before making its branching decision. Observe the following:-

```
IF X = 20 OR X = 21 THEN GOTO 100
```

Here the computer must act on the GOTO statement, if X is equal to the value 20 OR if X is equal to the value 21. If EITHER of these statements are True, then the branching to line 100 would take place.

Here is another example of a double decision:-

```
IF X = 20 AND Y < 21 THEN GOTO 100
```

With this example, the branching to line 100 will only take place if the value of X is equal to the value 20 AND the value of Y is less than 21.

There is a third variation of the IF statement we have not yet seen and that is the ELSE choice in branching. In our earlier counting program we could have written line 50 as follows:-

```
50 IF X = 100 GOTO 80 ELSE X = X + 1
70 GOTO 50
```

If we had taken this course we would have reduced the program

length, which, at this stage, would not have seemed important. But, as the programmes get longer and more sophisticated, program length and processing speed play a major part in producing an acceptable and faster program and the more compressed the programme, the faster it will run.

Before we leave Branching, there is one very important variation of the GOTO statement we have yet to discuss. This is Multiple Branching, which allows your program to branch to one of a number of lines from a single statement. An integer is used to direct the program to the desired line, for example:-

```
10 INPUT "Enter a number from 1 - 6 "; X%
20 IF X% > 6 THEN GOTO 10
30 ON X% GOTO 100,200,300,400,500,600
40 END
100 PRINT "This is line 100":END
200 PRINT "This is line 200":END
300 PRINT "This is line 100":END
400 PRINT "This is line 400":END
500 PRINT "This is line 500":END
600 PRINT "This is line 600":END
```

When the operator enters a number between 1 and 6, the program branches to the appropiate line. For example, if the operator entered 5 <ENTER> then the program would branch to line 500 (being the fifth number in the GOTO statement) and the message "This is line 500" would be displayed on the screen, before the program was terminated by the END statement. The numbers in the GOTO statement do not have to be in numerical sequence, it is only their relative position which is important.

In the above example, I have introduced the concept of multiple statement lines. You will notice that the lines from 100 to 600 have the PRINT and the END statement on each line. This is accomplished by placing a colon (:) where the line would normally have been broken. This gives the program a more efficient use of memeory, but it must not sacrifice legibility!

LOOPING

You may wish to repeat a part of a program a number of times and this could be accomplished by placing a GOTO statement at the end of a section of program. Thus, in our counting program we could change line 90 to read:-

```
90 GOTO 20
```

This would make our program count to 100, display "FINISHED" on the screen then loop back to line 20 to commence counting over again.

There is, however, no control over the program and this would be very desirable in normal applications. We may, for example, wish to execute a mathmatical calculation ten times and the FOR - TO and NEXT statements would control this.

The FOR - TO statement controls the number of times the loop is executed and must always be the first line of the loop. The NEXT statement, indicates the end of the loop as in the following example.

```
10 CLS
20 PRINT "SQUARED"
30 Z = 2
```

```
40 FOR X = 1 TO 10
50 PRINT "THE SQUARE OF "Z; "IS "Z * Z
60 Z = Z + 1
70 NEXT X
80 END
```

In this example, the variable x specifies how many times the loop is executed and could quite easily have been FOR X = 10 TO 20 as the starting point does not matter. The variable will always increase by 1 unless otherwise directed by the STEP clause to the FOR - TO statement.

```
FOR X = 1 TO 20 STEP 2
```

This statement would also execute the loop 10 times.

Negative and fractional numbers are also permitted as are variables, eg.

```
FOR X = 10 TO 1 STEP -2
FOR X = 20 TO 100 STEP 2.5
FOR X = 90 TO 1000 STEP Y
FOR X = N * 2 TO P - 4 STEP Z
```

You may use a GOTO statement within the loop to branch out of the program, but you cannot branch into the middle of a loop as the FOR - TO statement would not have been set.

One loop can be imbedded or nested within another loop, in fact there can be several levels of nesting, but you must observe the following rules.

1) Each of the nested loops must begin with its own FOR - TO statement and end with its own NEXT statement.
2) Each loop must have its own unique variable.
3) Each inner loop must be completely imbedded within the outer loop.

```
FOR X = 1 TO 10
FOR Y = 1 TO 4
NEXT Y
NEXT X
```

In the above loop, the Y loop is completely embedded within the X loop and you could not have had the NEXT X before the NEXT Y.

CHAPTER 5

ADDITIONAL FEATURES OF XTAL BASIC

ARRAYS

Up until now our example programmes have used different letters and numbers to define our variables, but there is another, more versatile, method of holding program data. This is known as a subscripted variable. With this method, we use a letter, or letters, followed by a number enclosed in parentheses. For example:-

```
10 B(1) = 5:B(2) = 10:B(3) = 15:B(4) = 20
20 PRINT B(1),B(2),B(3),B(4)
```

The numbers in the parenthesis are the subscript's and the data assigned to the subscripted variables is collectively known as an array. In the above example we have an array called B with Data 5,10,15 and 20.

The advantage of forming an array is the speed and methods of manipulating the data. Take the following example, which produces results similar to the one above:-

```
10 FOR X = 1 TO 4
20 B(X) = X * 5
30 NEXT X
40 PRINT B(1),B(2),B(3),B(4)
```

A subscripted variable can contain either numeric or string data and the array can contain data in subscripts numbered 0 to 10 ie eleven elements. If we require a larger array than this, we have to DIMension the variable with a DIM statement in the program, before we use the variable. A DIM statement can define an array to have a size from 0 to 65,535.

In our above example, if we had wanted an array of 20 elements, the program would have been altered to read as follows:-

```
 5 DIM B(20)
10 FOR X = 1 TO 20
20 B(X) = X * 5
30 NEXT X
40 PRINT B(1),B(2),B(3),B(4)
```

Here, line 5 defines the variable B to have 21 elements from 0 to 20 though we are only using 20 of them. (In normal practice element 0 is not used).

The examples so far have shown ONE-DIMENSIONAL arrays, but much more use can be made of multi-dimensional arrays.

You may visualise this as being a series of rows and columns. Let us take a simple example of a TWO-DIMENSIONAL ARRAY containing the following data:-

```
TATUNG (UK) LTD                 0952-613111
SCREENS MICRO.                  09274-20664
SYNTAXSOFT                      0282-698849
MICRO EXPRESS                   0827-51480
```

This is a 4 x 2 array having 4 rows and 2 columns. We would DIMension this array as, for example, Z$(4,2) and variable Z$(1,1) would contain the string "TATUNG (UK) LTD" and variable

Z$(1,2) would contain the string variable "0952-613111". Variable Z$(3,2) would contain string "0282-698849" and so on.

We could, if we wish, enter this data as follows:-

```
10 DIM Z$(4,2)
20 Z$(1,1) = "TATUNG (UK) LTD":Z$(1,2) = "0952-613111"
30 Z$(2,1) = "SCREENS MICRO.":Z$(2,2) = "09274-20664"
40 Z$(3,1) = "SYNTAXSOFT":Z$(3,2) = "0282-698849"
50 Z$(4,1) = "MICRO EXPRESS":Z$(4,2) = "0827-51480"
60 FOR X = 1 TO 4
70 FOR Y = 1 TO 2
80 PRINT Z$(X,Y),
90 NEXT Y
100 NEXT X
```

(note: The comma at the end of line 80 causes the telephone numbers to be printed at the next tab stop).

Even for such a small array, this method of entering data is rather tedious and large amounts of data can be more economically entered into variables with the READ and DATA statements.

The READ statement specifies the variable or variables whose values are to be READ into the computer. These variables can be ordinary or subscripted and represent string or numeric data. If there are two or more variables, they must be seperated by commas.

The DATA statement assigns the appropriate values to the variables listed in the READ statement. The values in the READ statement are seperated by commas and are read into the variables sequentially. There must be a data entry for every variable which is read. An example should make this all clear:-

```
40 READ A$,B$,C$,D,E,F
70 DATA RED,BLUE,GREEN,5,10,15
```

Here the variables A$,B$ and C$ have the string values RED, BLUE and GREEN respectively, and the variables D, E and F have the numeric values 5, 10 and 15 respectively.

We have seen, therefore, that this is a much more economic way of entering data into variables.

```
10 DIM Z$(4,2)
20 FOR X = 1 TO 4
30 FOR Y = 1 TO 2
40 READ Z$(X,Y)
50 PRINT Z$(X,Y),
60 NEXT Y:NEXT X
70 DATA TATUNG (UK) LTD,0952-613111,SCREENS
   MICRO.,09274-20664,SYNTAXSOFT,
0282-698849,MICRO
   EXPRESS,0827-51480
80 END
```

The same data may be used again in another READ statement by using the RESTORE statement. This resets the data pointer to the first item in the DATA statement. Try these two simple examples to see what will happen if the RESTORE statement is not used, before attempting to re-read the same data:-

(not correct)


```
10 READ A,B,C
20 PRINT A,B,C
40 GOTO 10
50 DATA 1,2,3
```

(correct)

```
10 READ A,B,C
20 PRINT A,B,C
30 RESTORE
40 GOTO 10
50 DATA 1,2,3
```

RANDOM NUMBER GENERATING

Often we wish to generate a random number. This could be to assist with a program for a game of chance, or to enter data to test a business program.

The statement generating a random integer number between 2 and 65535 or a random floating point number between 0 and 1 is RND.

The statement RND(1) would produce a random floating point number between 0 and 1

The statement RND(100) would produce a random integer number between 0 and 99

The statement RND(65535) would produce a random integer number between 0 and 65534

To create a floating point number greater than 1, we would just combine the two functions, such as:- A=RND(1) + RND(100).

A simple game will give a good example.

```
10 CLS
20 PRINT "              TOSSING A COIN"
30 INPUT "HEADS OR TAILS ";A$
40 B = 0
50 IF A$ = "HEADS" THEN B = 1
60 C = RND(2)
70 PRINT "I TOSSED ";
80 IF C = 1 THEN PRINT "HEADS"
90 IF C = 0 THEN PRINT "TAILS"
100 IF B = C THEN PRINT "YOU WIN"
110 GOTO 30
```

GO SUB

As your programmes become more complex, you will find you are using the same routines on many occasions. You could, if you wished, write these routines over and over again, but this would not only become tedious, but would waste the memory of the computer. A much more economical method is to write the set of instructions once, then go to the routine, every time it is needed, returning to the original place in the program when the routine is finished. The end of a Sub Routine is marked by the statement RETURN at the end of the routine. The statement to instruct the program to go to this routine is GOSUB and is used on many occasions in professionally written programmes.

```
10 REM * EXAMPLE OF GOSUB
20 INPUT "ENTER A NUMBER ";A
30 IF A = 0 THEN END
```

```
  40 GOSUB 1000
  50 PRINT "THE SUARE OF ";A;" IS ";B
  60 GOTO 10
  70 REM END OF MAIN PROGRAMME
1000 REM START OF SUBROUTINE
1010 B = A * A
1020 RETURN
1030 END
```

The use of GOSUB and GOTO makes structured programming possible. This is the method of breaking up your program into blocks of code and calling them, when necessary, from the main program flow. This method makes following listings of the program easier for yourself and others to read. Particularly if you return to your programme at a later time.

Another useful routine in professionally written programmes, is the statement ON ERR GOTO or ON ERR GOSUB which stands for ON ERROR GOTO and ON ERROR GOSUB respectively.

If you write a program for someone else, chances are there maybe some eventuality for which you have not catered. During use, the program might suddenly stop and display an error message on the screen. This may be that the user has forgotten to put a disk in the right-hand drive, or that the data disk is full. There are many errors over which the writer of the program has no control. To leave the program without any error detection would cause the program writer to receive a number of telephone calls reporting error messages, that could be easily resolved by the user himself, and the way to detect these errors is to use the ON ERR statement. For example type in the following program:-

```
10 CLS
20 INPUT "ENTER A NUMBER ";A
30 INPUT "ENTER A NUMBER TO DIVIDE IT BY ";B
50 PRINT " THE ANSWER IS ";A/B
60 INPUT "DO YOU WANT TO TRY AGAIN (Y/N) "Z$
70 IF Z$="Y" GOTO 20 ELSE END
```

The program seems fairly fool-proof, if the user types in 20 as the first number and 5 as the second number, the program prints out THE ANSWER IS 4 on the screen and asks if you want to try again. But, supposing the user types in 20 for the first number and either enters 0 or presses the <ENTER> key with no number for the second number, then the program crashes and we get the error message "Division Error in 50" displayed on the screen. This means the computer has tried to divide one number by 0 which it is unable to do, the message also indicates in which line the error has occured. In this case, it is in line 50.

As well as displaying error messages, the computer generates an error number. To see the number, just enter PRINT ERR <ENTER> (do not give it a line number), straight after the Ready prompt on the line after the 'Division Error in 50' message. The error number for Division Error should have been displayed as decimal 11, one of 35 errors which the Einstein detects.

We can make use of this facility by instructing the computer to detect certain errors and take specific actions when these occur. Modify the above program by adding the extra lines:-

```
10 CLS
15 ON ERR GOTO 1000
20 INPUT "ENTER A NUMBER ";A
30 INPUT "ENTER A NUMBER TO DIVIDE IT BY ";B
```



```
  50 PRINT " THE ANSWER IS ";A/B
  60 INPUT "DO YOU WANT TO TRY AGAIN (Y/N) "Z$
  70 IF Z$="Y" GOTO 20 ELSE END
1000 IF ERR = 11 PRINT "YOU CANNOT DIVIDE BY 0 ": GOTO
     15
```

Now, if you run the program and enter 0 as the second number, you will be given the message THE ANSWER IS YOU CANNOT DIVIDE BY 0 and you will be prompted to enter the numbers again.

You will notice that the program returns to line 15 on encountering an error, because you must re-action the ON ERR statement after it has been used. We could have changed line 1000 to read:-

```
1000 IF ERR=11 PRINT "YOU CANNOT DIVIDE BY 0":ON ERR GOTO
1000:GOTO 20
```

The former takes up less program space and is tidier.

You could have further lines after line 1000, for example:-

```
1010 IF ERR=23 PRINT "THAT DRIVE IS NOT AVAILABLE": GOTO 15
1020 IF ERR=30 PRINT "DISK FULL, PUT BLANK FORMATTED DISK IN
DRIVE":GOTO 15
1030 IF ERR=28 PRINT "YOUR DISK IS WRITE/PROTECTED" GOTO 15
```

These are, of course, only examples, and do not attempt to write data to a disk. A full list of the error codes (use decimal numbers) can be found as APPENDIX B in the EINSTEIN BASIC REFERENCE MANUAL.

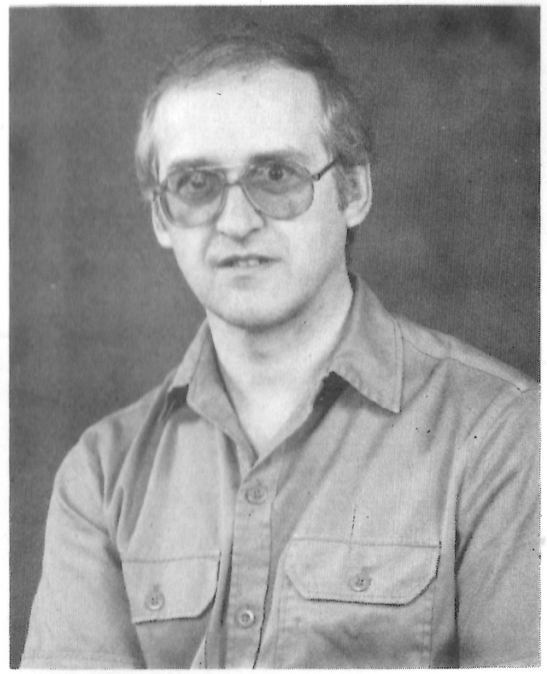
These are just a few of the many facilities available under XTAL Basic. I have not touched upon the important task of writing data to disk in either sequential or random access mode, neither have I touched upon computer graphics adequately covered by available software.

In conclusion, always remember that specialist help is available from the many Einstein Dealers and User Groups, so avail yourself to this service. However, your dealer also has a living to make, so be fair and buy your products through him.

SUGGESTED FURTHER TUTORIALS

Relatively Basic	Solo	Further XBasic Programming
Basic Tutorial	Solo	A Disk Tutorial of XBas
Apollo 11 Graphics Tutorial	Orion	Graphics Tutorial
Assembly Language Programming	Glentop	Assembly Language Tutorial
Using DR Logo on the Einstein	Glentop	Logo Language Tutorial

ABOUT THE AUTHOR



DAVID BELL A.I.D.P.M.

DAVID BELL WAS BORN IN LANCASHIRE, ENGLAND AT THE END OF THE SECOND WORLD WAR AND SPENT THE FIRST 12 YEARS OR SO WORKING FOR A STOCKBROKING FIRM IN LONDON.

HE FIRST BECAME INTERESTED IN COMPUTING WHEN HE INVESTED IN A TANDY MODEL 1 MICROCOMPUTER AFTER BEING MADE REDUNDANT. THIS WAS DONE IN AN EFFORT TO CREATE JOB SECURITY IN FUTURE EMPLOYMENT.

SINCE THAT TIME, DAVID HAS USED THE ALTOS, ALPHA AND FORTUNE MULTI USER AS WELL AS THE SANYO, APRICOT AND I.B.M P C's . HE IS CURRENTLY SOFTWARE PROJECTS MANAGER FOR TATUNG (UK) LTD. PART OF HIS CURRENT RESPONSIBILITIES INCLUDES THE AQUISITION OF SOFTWARE FOR THE EINSTEIN MICROCOMPUTER. THIS RESPONSIBILITY BROUGHT ABOUT THE PUBLISHING OF THIS BOOK.

DAVID IS AN ASSOCIATE MEMBER OF THE INSTITUTE OF DATA PROCESSING MANAGEMENT AND HAS QUALIFICATIONS IN THE AREAS OF COMPANY SECRETARIAL AND CREDIT MANAGEMENT.

HE LIVES IN TELFORD WITH HIS WIFE, JANE, WHO IS THE PROPRIETOR OF A BALLET SCHOOL.

EINSTEIN PRIMER

AN EASY TO READ AND UNDERSTAND BOOK DESIGNED FOR THE BEGINNER TO COMPUTING WITH THE EINSTEIN.
COMPUTER JARGON HAS BEEN ELIMINATED AND THE AUTHOR HAS ASSUMED THE READER TO HAVE LITTLE OR NO PREVIOUS KNOWLEDGE OF COMPUTING.
NUMEROUS TIPICS AND EXPLANATIONS HAVE BEEN GIVEN AND IN ADDITION A USEFUL DESCRIPTION OF PRODUCTS AND PERIPHERALS FOR THE EINSTEIN.

ESTABLISHED EINSTEIN OWNERS WILL FIND THE AUTHORS WORKING DESCRIPTIONS ON VARIOUS SOFTWARE PACKAGES VERY USEFUL AND AN INTRODUCTION TO THE BASIC PROGRAMMING LANGUAGE HAS ALSO BEEN PROVIDED.

ScreenSoft

£9.95