



EINSTEIN MAGAZINE

TENTH  
BIRTHDAY  
COMPENDIUM

No. 77

NOTICED ANY CHANGES YET?

This issue marks the 10th birthday of the United Kingdom Einstein User Group. As the group has always been primarily magazine-based, it also marks ten years of continuous publication of the user group's EINSTEIN MAGAZINE -- which is a considerably better record than the "official" EINSTEIN USER MAGAZINE, which B&H have failed to publish for 3 years.

This is all very well for B&H. They've got the subscription money and are sending out neither magazine nor refunds. Our problem is that people tend to assume that EINSTEIN USER is published by the user group -- and WE get the blame !!!

B&H are good at excuses -- "Our printer is a bit poorly / has retired / is dead and buried / and no-one else knows how to run the printing machine" -- but it just isn't good enough to rip people off like this, when they've handed over money in good faith, and someone else gets the blame.

We know all about printing problems -- we've had quite enough of our own lately! Nevertheless, despite our own lack of resources, all too few Einstein users, & hectic cash-flow problems, we've still managed to produce a dozen magazine issues - plus this special one - since taking over from Graham at Ipswich. All that the poor old B&H magazine subscribers have received in that time is lame excuses.

Several times we've offered to resolve the problem by taking over B&H's commitments and honouring them ourselves, but so far we have only been met with a rather paranoid "You're only trying to steal our database." !!!

COME ON B&H -- RIPPING OFF THE PUNTERS IS JUST NOT ON.

Either send them what they've paid for - your "official" magazine - or else refund the money they've paid you! If you magazine is no longer viable, and you can't/won't refund the subscription money (maybe you've long since spent it all?) then simply transfer your magazine title & subscription list to us -- no payment either way -- as Graham at Ipswich did with OUR magazine when it was no longer a viable proposition for him to publish it -- so WE can fulfil YOUR commitments!

MEANWHILE, BACK AT THE RANCH .....

There was really only one way to celebrate 10 yrs of continuous publication -- by producing this issue as Compendium 2. It brings together in one volume all those articles that you know are in the back numbers somewhere, but can never find them. Of course, it's still only a minute proportion of the total content, and you will certainly need a full set of mags if you are a new Einstein owner, or maybe just to fill in the gaps. We are about to reprint EM 1/1 to 1/4, and the rest are still in print -- see back cover.

CONTENTS

- The Portable Einstein ..... 7  
(Ron Morris)
- TPC-2000: The Einstein's Big Brother..... 7
- So, you want to own an IBM-compatible?..... 8
- Now I've bought an Einstein, how do I switch it on?.... 9  
(Terry Madicott)
- What is a Spreadsheet? How do I use it?..... 11  
(Peter Gosden)
- FIND.COM: a very useful little utility..... 13  
(George Longden)
- Delving into Disks, Dos and CP/M..... 15  
(Vin Davies)
- The Operating System, Life and Everything..... 22  
(Phillipe Henrie)
- Interrupts and polling explained simply..... 29  
(Philip Eley)
- Add a "Dump screen to printer" routine to Dos 1.31.... 32  
(Chris Giles)
- Disk labels: upside down is the right way up!.... 33  
(Stuart Marshall)
- For the 256: Demonstration of Colour Scrolling... 37  
(MRV Peterson)
- XBAS: Produce bar charts, plot graphs..... 38  
(David Williams)
- Blend in more colour on your TC-01..... 40  
(Shaun Jeffery)
- XBAS: Educational: Find the town on the map.... 42  
(David Williams)
- WDRRO files: Naughty, but you can make it nice... 44  
(A B. Wilmot)
- XBAS programming: Do it like the professionals... 45  
(Vin Davies)
- Joysticks: how to adapt them to your Einstein... 48  
(Stuart Wilson)
- Build your own Burglar Alarm..... 49  
(Dave Arts)
- DAC Project: Polyphonic Hi-Fi on Albert..... 56  
(Josef Karthäuser)
- Build a Low Pass Filter for even better music.... 61  
(Colin Coker)
- Composite Video: improve your output..... 64  
(Stuart Marshall)
- Using your MOS: what the user manual doesn't tell you... 66
- Another for the 256: your own auto-boot utility... 70  
(Peter Mansell)
- The Scratchpad: what you didn't know you didn't know!... 71  
(Josef Karthäuser)
- More 256: disk sector skew & modified sector gaps... 76  
(MRV Peterson)

## THE PORTABLE EINSTEIN

Allow me to give you a problem and then I will explain our solution.

You are a consultant working for a partnership that has been tasked by a client to provide a computer which will assist their consultants to sell corporate pension and investment plans. They have no real computer knowledge but they have drawn up a list of essential criteria;

1. Portable or trans-portable.
2. Full colour graphics.
3. Uses television and large screen video and not a monitor
4. Storage facilities for client's data.
5. 80 column printer (silent if possible).
6. Prestel/viewdata facility.
7. Costs less than £1000-00.

As our company was to develop the software we added a few more requirements, namely;

1. CPM or MSDOS compatibilty.
2. Good 'off the shelf' programs available.
3. Compiler available for BASIC
4. If possible 16 bit as large number crunching requirement for actuarial data.
5. Must be extremely reliable as systems are for 15 offices in both the UK and Germany.

We were presented with this problem in November 1984 and my first reaction was 'you have got to be joking at £1000-00.' But I was tasked with investigating the current market for a possible solution. 32 portable and trans-portable systems later I knew exactly what I knew at the start of the exercise - no such system available. Not one of the systems I investigated could even drive a colour television set. The only system that came close cost £3650-00.

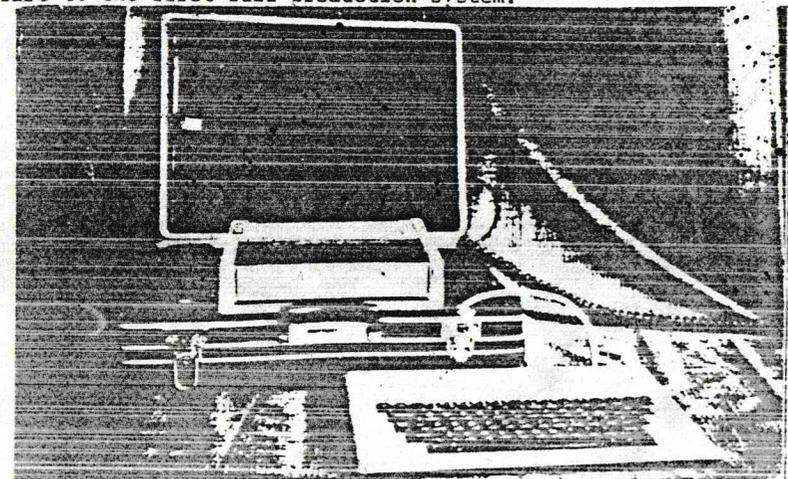
The next suggestion was that I designed a completely new computer but with an order for only 150 systems, potential of 1000 if trial went OK, this was out of the question. We completed a limited market research survey to see if there was a slot in the industry for such a portable system. We decided that there was no market in the home/small business sector but there was a slight possibility as a sales aid within large companies. The cost of launching into such a market added to the research and development cost, in both time and materials, would prove impractical to a small company like ours.

The next suggestion was that we considered taking an existing computer, modify and re-package it so that it became portable. Good idea but which system? I was tasked with looking at the 'home market' for a possible solution - not very impressed. I then, by chance, meet representatives of Tatung in a car park on the M1 (thats another story). They convinced me that I should look at the Einstein computer. Its specification looked good and it had a major advantage over all the others, namely British designed and British built. This meant that, should I have any problems, I could have face to face meeting with both hard and software engineers

It was obvious that there is no way the Einstein can be classed as a portable computer, particularly as we intended to add an 80 column printer. So we decided to design our own briefcase in which to house all our hardware systems.

Plastics and forming was a whole new world that was as full of jargon as the computer industry. The only difference being that the plastics world deliver parts even later than the computer industry. The case and insert are vacuum formed from black ABS which has a very good strength to weight ratio. The case measures 20" by 14" by 6" and when ready for use it weights 18lbs. The lid is detachable and has space for documents and manuals etc. It took months of meetings and discussions to produce the final product and allow our engineers to try and manufacture the first transportable Einstein.

The picture below is the final product which took 14 months from start to the first full production system.



The computer specification is as for the Einstein and the printer above is the HP Thinkjet, we can fit the cheaper Brother HRS Thermal.

What have we learnt about the Einstein that could be of advantage to you?

\*\*\*\*\*

Hardware.

1. The computer is extremely reliable even when consultants throw it into cars.
2. The power unit is capable of driving 4 disk drives and a printer as well as the computer.
3. The Einstein has more standard features than any other computer we studied. This means that we have only really scratched the surface of the potential development and utilisation.

Software.

1. Programs using large areas of RAM must incorporate 'garbage collection' routines or Memory full errors occur
2. Full account must be taken of the variable register when large number crunching programs are developed. I reduced the running time of a program from 7 mins 21 secs to 1 min 11 secs by addressing this problem.
3. Compiler is good but the edition I have has 3 limitations, namely;
  - a. Will not compile Definable functions.
  - b. Will not compile Print #0 without a space between the word Print and the hash.
  - c. Will not compile the SIZE command.

These are extremely minor limitation and have not caused any real problems.

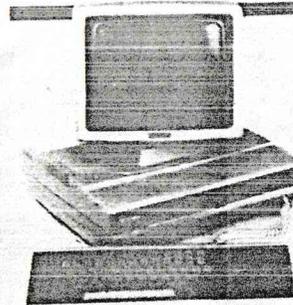
We now have manufactured 150 systems and they are deployed throughout the UK and Germany. We have developed a couple of other Einstein based systems, namely

1. 4 disk Einstein. We have modified the front of the Einstein to allow 4 disk drives to be fitted. The speaker has been removed to the back of the computer. Apart from the advantages of 4 drives we have developed a program that will backup 3 disks at a time. We use this for the mass copying of actuarial data disks for our client.
2. Telephone Logging System. We are developing a computer that will log all telephone calls, by extension and department.

Ron Morris,  
Technical Director  
Alternatives of Lincoln Limited

\*\*\*\*\*

THE EINSTEIN'S BIG BROTHER



DON'T MISS THE CPM Deal OF the CENTURY

The FABULOUS CPM TATUNG PC2000 Professional Business System

A cancelled export order and months of negotiation enables us to offer this professional PC CPM system, recently on sale at OVER £1400, at a SCOOP price just over the cost of the two internal disk drives!! Or less than the price of a dumb terminal!!

Not a toy, the BIG BROTHER of the EINSTEIN computer, the DUAL PROCESSOR PC2000 comprises a modern stylish three piece system with ALL the necessities for the SMALL BUSINESS, INDUSTRIAL, EDUCATIONAL or HOBBYIST USER. Used with the THOUSANDS of proven, tested and available CPM software packages such as WORDSTAR, FAST, DBASE2 etc, the PC2000 specification, at our prices, CANNOT BE BEATEN!!

The central processor plinth contains the 64K, Z80A processor, DUAL TEAC 55F Double sided 40/80 track disk drives (1Mb per drive), PSU, 2K of memory mapped screen RAM, disk controller, RS232, CENTRONICS and system expansion ports, and if that's not enough a ready to plug into STANDARD 8" DRIVE port for up to FOUR 8" disk drives, either in double density or IBM format. The ultra slim 92 key, detachable keyboard features 32 user definable keys, numeric keypad and text editing keys, even its own integral microprocessor which allows the main Z80A to devote ALL its time to USER programs, eliminating "lost character" problems found on other machines. The attractive detachable 12" monitor combines a green, anti-glare etched screen, with full swivel and tilt movement for maximum user comfort. Supplied BRAND NEW with CPM 2.2, user manuals and full 90 day guarantee. Full data sheet and info on request.

PC2000 System with CPM Etc. COST OVER £1400

PC2000 Business System with CPM and 'Ready to Run' FAST Sales and Purchase ledger, supports up to 9000 Accounts, VAT etc. COST OVER £1700

PC2000 Wordprocessor System with CPM and TEC FP25 daisywheel printer

NOW only £399

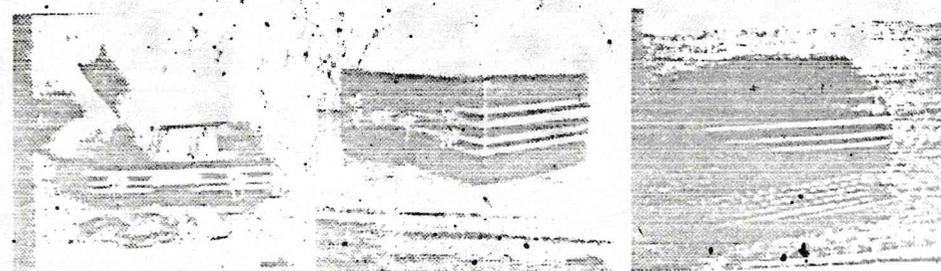
NOW only £499  
Carrage & Insurance £1200

NOW only £799

Abandoned at birth due to the government-funded research consortium back home in Taiwan successfully cloning a legal IBM BIOS by launch date, this was intended as a top-of-the-range CP/M 2.2 serious business computer. The initial production batch were remaindered off later by dealers.

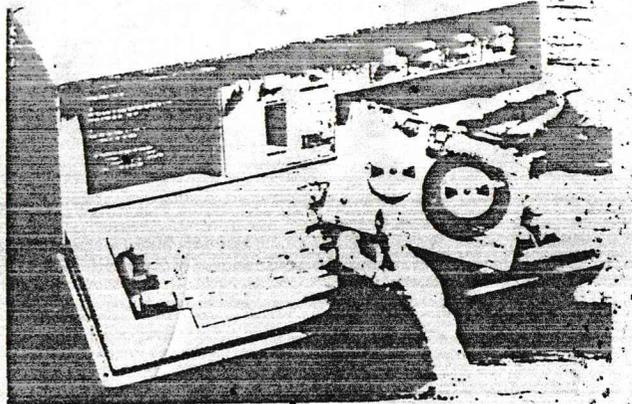
We'd intended to reproduce the full data sheet and the only computer magazine full review that we've seen on the machine, but both have been misplaced on our files, and this advert is all we have to hand at press time. Apologies!

\*\*\*\*\*



\*\*\*\*\*

WHY ON EARTH WOULD YOU WANT AN IBM-COMPATIBLE?



With the announcement of the IBM 5100 system in a press release dated Sept. 9, 1975, personal computing gains an entry from the industry's production and service giant, IBM. The IBM 5100 is being marketed primarily as a *problem solver* for industrial, commercial and professional people with the result that it is a very professional package at a premium price. But you will get a lot of function when you buy one of these computers and you'll be able to call upon IBM's longstanding reputation for good service and customer handholding, the points which have led to the commendable success of IBM as a computer company.

What IBM engineers have done is to design a 50 lb. package of interactive personal computing which includes the following major features as standard items:

- System software is built-in, with access to BASIC and/or APL depending upon options purchased. These languages and the necessary monitor programs are hardwired into a read-only memory.
- A video screen is built-in with up to 1024 characters displayed in a 16-line by 64-character format.

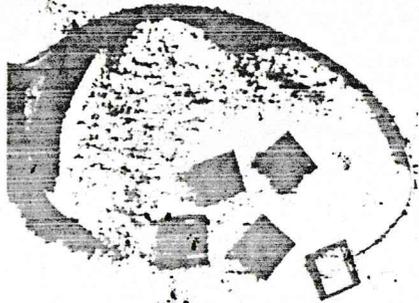
- An interactive keyboard is standard, including the usual text entry section as well as a separate calculator style keypad. The keyboard has special function coding for all the APL and BASIC syntax elements.
- User memory starts at 16K bytes in the minimum configuration and can be expanded to 64K bytes (65,536).
- A magnetic tape cartridge storage device is standard. This is built into the unit, and becomes the primary method of storing user data and programs. It is also used to load IBM supplied programming packages. The cartridges for this device hold up to 204,000 characters of information.

- IBM 5100 processor ... \$8975 to \$19,975, depending upon user memory (16K, 32K, 48K or 64K bytes) and language (APL or BASIC or both) options.
- IBM 5103 printer ... \$3,675 purchases an 80 cps 132-column dot matrix line printer.

You get all this function and professionalism from IBM by paying a high price. This machine is not intended to be a toy, although it would make an excellent one. It is intended as a production tool for people who presently use time sharing terminals, programmable calculators or other personal computers in daily work. Prices mentioned in the press release are:

Miscellaneous features also available for the machine include a TV monitor output, the external I/O adaptor used with the 5103 and 5106 devices, a communications adaptor which makes the 5100 emulate an IBM 2741 communications terminal, and a carrying case.

As an IBM engineered product, you can expect a solidly built computer. If you are a business or professional person needing a high quality calculational and programming tool, then you should investigate the 5100 as an item of capital equipment which you can incrementally use to program numerous BASIC games when you are not using it for business. But if your sole interest in the machine is as a luxury toy, you have to be moderately well off to purchase the IBM 5100 at its present price.



\*\*\*\*\*

FOR THE FIRST TIME COMPUTER USER

Beginners start here as Terry Madicott takes you on your first driving lesson.

After plugging in your Einstein to the mains and either connecting it to a television or monitor, switching on produces an opening screen asking you to insert a disc to drive 0/A and press CTRL+BREAK. Although you can insert the disc before switching on Albert it is better practise not to, as there is always a risk of corrupting a disc if it is in the machine at power on/off. (Here come another 500 letters telling me it has never happened to them!) In computer jargon the symbol for the control (CTRL) key is ^ and is sometimes referred to as a 'top hat' or 'carrat'. So if you see ^A this means hold down the CTRL key and the A key at the same time. If the machine is switched on without a disc in drive 0 it will power up in what is known as MOS, the Machine Operating System. To the less experienced user the computer is of little use at this stage, although if you progress with your own programs you will find the MOS of great value.

So having inserted the master disc and pressed CTRL+BREAK this puts Albert into the Disc Operating System, which is known as DOS for short. Pressing the above keys should cause the disc drive indicator to come on for a few seconds and the screen should change to show that DOS has loaded. Under the heading of which revision of DOS has loaded there should be on the far left hand side a 0: this is known as the DOS prompt and tells us that we are logged onto drive zero. The DOS is waiting for a command and all disc commands will be directed to drive

zero. The Einstein can access upto 4 disc drives, if you have a second internal drive you can log on to this drive by placing a disc into it and typing 1: followed by the enter key. The DOS prompt will change to 1: telling us that all disc commands will be directed to drive 1.

Now type DIR followed by the red enter key. On the display should be showing a list of all the files and programs from the disc we are logged into. DIR stands for Directory, it is just a list of what is on the disc. A filename can be upto 8 characters long, there is a full stop after the eighth character and then 3 more characters known as the extension. We can generally tell from the extension what type of file is contained on the disc. Names ending in .COM are Command files, these are machine code files that can be run directly from DOS, after the prompt just type the filename without the extension and the program should run. The DOS will only run programs ending in .COM, see what happens if you try to run a program with a different extension. The DOS/MOS manual pages 24-26 refer. On the master disc XBAS is an example of a .COM file. Filenames ending in .XBS, .OBJ, and .ASC are usually used with XBAS, any other extensions will usually refer to what are known as data files.

DATA FILES

Data files are used to store text and numeric information, an example is that of the telephone book.

There are several ways we can look at the contents of a file, if you want to cheat at hangman and look at the data file containing the words you can use the 'DISP' command from DOS: If you do a DIR on the master disc you should see two files, DICTIONE.DAT and DICTIONH.DAT, to DISPLAY their contents, after the DOS prompt type;

DISP DICTIONE.DAT&lt;E&gt;

holding down the BREAK key will stop the screen scrolling or you can use ^S to stop/start the display. This method will only work for files that contain text, if you try it on a .COM file it will produce funny results and probably crash the computer needing the reset to be pressed.

To look at what a .COM file contains we can LOAD it into memory from DOS by doing

LOAD filename.ext&lt;E&gt;

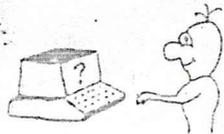
This loads the program or file into memory starting from location 0100 Hex. By entering MOS<E> we can use the T (for tabulate) to examine memory locations 0100 onwards. By entering

T 0100 FFFF&lt;E&gt;

all locations from 0100 would be displayed on the screen, the display will scroll, again use the break key to hold the display stationary. Generally this information will not mean a lot unless you have a knowledge of Z80 machine code, although there may be some text embedded in the code which can be read. Often at the beginning of a program there is a copyright notice or the name of the author.

This principle can be used for any filename.ext and after loading a file the size in blocks of 256 bytes will be displayed. We can now save the file by using this value and a filename.ext in the form SAVE XX filename.ext<E> where XX is the number of blocks. This is one method of copying a file.

Certain programs define the extensions used by their datafiles, e.g. WPRO .UFT, SCREENPLUS .MEM, DBASE II .DBF etc. So if you are familiar with which extensions go with which programs you have a good guide to their contents. Any files ending with an extension of .DOC .GEN .TXT .INF etc. are likely to be ASCII information files and should be capable of being DISPLAYed. These type of files often accompany P.D. software containing instructions for the actual running of the program.



IF YOU DON'T KNOW WHAT A  
BUTTON DOES... PRESS IT!



BUT DON'T BLAME THE  
RESULT ON ME.

## SPREADSHEETS ? EASY !

A LOOK AT THE CRACKER BY PETER GOSDEN

There is no doubt that many people go in awe of Spreadsheets. Like most things, however, familiarity and use soon overcome the initial fears. This is certainly my experience with Spreadsheets.

I came to them by chance. Before my Einstein I had a twin drive Oric Atmos. In what is still my favourite Basic I had written a sophisticated home Finances program. The program recorded all my incomings and outgoings over a full year with cash flow projections on a monthly basis, full details of each month by item, or item by month and a graph of the cash flow over the 12 months. When I began to find the Machine limited and occasionally unreliable (probably a voltage regulator problem) I moved to the Einstein and soon started to rewrite my home finances program in XTAL Basic. What a headache - I was having to Experiment with new commands and syntax and file handling. I soon realised that I did not have the time to both educate myself and convert the program so looked for a home finance package. I ordered Home Budget from B & H but it was far too limited. B & H exchanged it for the CRACKER Spreadsheet (and a further

cheque). I had no real idea what this would do but it seemed to be the big boys product and I was desperate. It has turned out to be the most practical program I have bought.

So what is a Spreadsheet? The usual answer is a large blank sheet of paper that you can scan by moving it vertically or horizontally past the screen. That is quite right, but it is also whatever you want it to be. It can be a set of accounts, a database, a mailing list, a parts inventory, etc. You see, rather than being just that large sheet of paper divided into boxes that you can record things in, you can also write things in that will do things.

Lets take an example. The Spreadsheet is ruled across and down forming boxes one line deep. These boxes can be lengthened or shortened to fit your data and are identified, like map co-ordinates, by numbers across and down. Spreadsheets have alphabetic characters across and numerics down starting at the top left corner, so the top left box (called a CELL) is called A1, and the one beneath it A2. The one to the right of that is B2 - get it? Lets say we want to list some figures and get a total. We will enter 5 in A1, 6 in A2 and 7 in A3. We want the total in the cell A4. In A4 we enter  $A1+A2+A3$ . The Spreadsheet automatically adds together the values in A1, A2, A3 and prints 18 on the screen in A4. Obviously a calculator could do that as well but now change A2 to 43 and A4 automatically changes to 55 ( $5+43+7$ ). That was a very simple example, but it is the basis of the power of a Spreadsheet. Lets make it a little more useful by adding a new line (ROW) at the top and a new COLUMN down the left. In the

new ROW CELLS A1, B1, C1, D1, E1 enter PARTS, BOX 1, BOX 2, box a, PART TOTAL, and in the new CELLS in COLUMN A - A2, A3, A4, A5 - enter 12345, 23456, 34567, BOX TOTAL

Put some numbers into the empty CELLS and you have something like this:-

PARTS	BOX 1	BOX 2	BOX 3	PART TOTAL
12345	5	9	16	
23456	43	11	13	
34567	7	3	12	
BOX TOTAL				

To make it work fill in the right and bottom CELLS with the formula to add the CELLS across and down, i.e. in E2 enter B2+C2+D2. The Spreadsheet will then do the totalling for you. Change any of the CELL contents and the totals in

both directions will also change. Extend this to 14 COLUMNS across and call these Months, etc, and as many ROWS Down as you need and you have a simple set of Monthly Accounts.

When you save a Spreadsheet to disk you lift the data you have entered off the paper and store it as you would the contents of a Wordprocessor or Database.

A Spreadsheet can be quite useful, can't it? But what else can it do? Want to print down in condensed print? Easy! Want to produce your data as a Bar Chart, Line Graph or Pie Chart? No problem. But what about getting the Spreadsheet to tell itself what to do? This is when Spreadsheets get really clever. Using commands called MACROS which are lists of the first letters of individual COMMANDS, you can enter a string of such letters, call up the CELL they are in, and automatically process those commands just as if you were entering them from the Keyboard one at a time. Why not use this to create a screen sized Menu in the top left corner. Write a Menu with your preferred options: Printing, Printing Graph, Printer Commands, Save, Load, Sub-menu, Quit. Give each a number. In a new COLUMN A enter the strings of letters that make up each operation, i.e. Menu item 2 in A2. Have one item in A as a return, say A9. Structure this to set your screen over the Menu and finish it with \*A. When you load your Spreadsheet enter \*A9 and you will be presented with your Menu. The last \*A will prompt you for the Menu number, which when entered will automatically action that choice. Tail any Menu choices in the same manner that require another MACRO command choice following their own action.

I havent tried to detail how data or changes are actually put into the Spreadsheet, but it is very straight forward and logical and all Spreadsheets prompt the user with available options and sub-options. In the case of CRACKER the manual is Substantial and well laid out. It pays to have a good read of it then follow through the first section that takes the reader step by step through the process of setting up a spreadsheet and entering different types of data and performing actions on that data.

I hope I have whetted your appetite for Spreadsheets. There are many more options to explore, including:

- If Then, Else Functions
- Table Handling Functions
- Statistical Functions
- Do and While Functions
- Searching
- Sorting
- Mailing Labels
- Looping
- Sub-routines, etc, etc.

## USING FIND.COM WITH TEXT FILES GEO LONGDEN TAKES A LOOK AT A P.D. PROGRAM.

I have for some time now been employing DBASE II in various roles from processing my Amateur Radio log and contacts to simply keeping a list of addresses/phone-numbers which I may require.

Having given a friend a copy of the PD programme FIND.COM off P.D. disc 173, I was somewhat miffed when he turned up and showed me how to use it. For simple one-off queries (whats old Freds phone number?) loading DBASE or EASIDATA is a long-winded process and it will often prove to be quicker to refer to a written list. Well why not use a written list - only in this case a text file. Remember, with FIND.COM, you do not have to load the file into memory. The programme dives into the disc and pulls out only the information you require and it does all this straight from DOS.

Using your favourite text editor write the file using key words in association with whatever data you consider you will need. One of my files contains friends (I only ever need their phone numbers) and also trade names for which I may also require the full address. In the latter case, a reference is included which will permit the address to be presented on screen in a tidy manner. Here is a sample:

\*\*\*\*\*

JOHN BARBARA	792481
PETE BROWN	69148
FRED SYLVIA GREEN SNOOKER	62481
AUTORAC COMPUTING STEPPER (A1)	0123 456789
A1 1 Homestead Cottages,	
A1 Fox Rd.,	
A1 MASHBURY,	
A1 Essex.	
PETE CRAY SEC WESTMORE ARC	0124 59595
etc. etc.	

You can fix it so that any-one can use this system by a little re-naming of the files. FIND.COM becomes INFO.COM and the text file is named simply ON. The disc contains both these files on one side (perhaps also your word processor for convenience). Stuff it into the drive and from DOS call 'INFO ON FRED' <ENTER>. If you have more than one FRED all will be displayed. In the case of AUTORAC, you may find it easier to remember STEPPER than the name. There on the screen you will see the leading line (including phone number just in case this is all you require) and a reference (A1). If you require the full address then simply call INFO ON A1.

Perhaps you are a small supplier of electrical fittings. The moron you employ on the counter doesn't know the first thing about computers but even he can take a disc pigeon-holed, PLUGS, stuff it into the drive and type 'WHERE IS INLINE' to find the location of a 15A INLINE PLUG. In this case, FIND.COM has been renamed WHERE.COM and your text file is called IS. From the following entries he will know exactly where to look.

5	PIN DIN	BIN A15
9	PIN DIN	BIN A13
15A	STANDARD	BIN B12
15A	INLINE	BIN C14
2	PIN ROUND	BIN D8

It matters not whether you list them in some sort of intelligent order or put 'em in as you find 'em. If the file gets too bulky, you may notice a reduction in speed but you would have to get a real monster to justify loading a Database manager. Separate sides could be used for other stocks (sockets, fuses, switches etc.), each one bearing the title IS and residing with WHERE.COM on an appropriately tagged disc. You could even programme one of the function keys with 'WHERE IS' to make life easier for him.

\*\*\*\*\*

This will not keep account of your stocks or issue a warning to re-order when an item reaches a minimum stock level etc. For this you must use a more comprehensive program but for a quick way of finding something conveniently, it's a lulu. I have a large library of books and reference material which occupies a number of shelves and folders. A title such as 'Solid State Power Supplies and Regulated Voltage Sources' would be entered as:

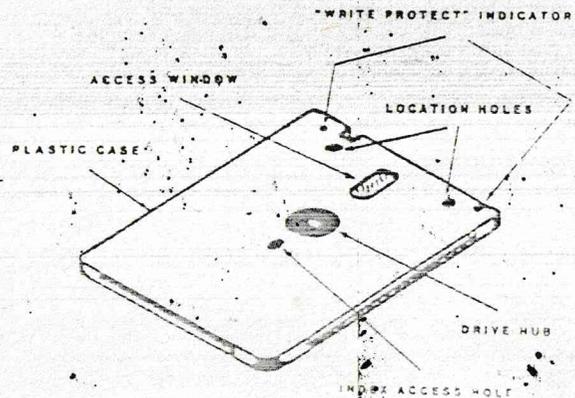
POWER SUPPLIES REGULATED VOLTAGE (P3)  
 P3 Solid state Power Supplies and Regulated  
 P3 Voltage sources  
 P3 Low Voltage Regulated Power Supplies\*  
 P3 Switch-mode Power Supplies  
 P3 Using Neons as variable-voltage Regulators  
 P3 Push-pull PSU (Radio World June/84 page 137)

SHELF A  
 SHELF C  
 SHELF C  
 FOLDER K  
 BOX 3

The first title is broken up to avoid confusing wrap-round but clearly I do not need to remember the complete title. Any of the keywords shown would find the reference (P3) and this would of course give me the complete list of related subjects, including the volume I was looking for. stick to a sensible convention - use upper case throughout for reference - and the limit is only as broad as your imagination and ingenuity.

DISCS, DOS and CP/M on the EINSTEIN (Vin Davies)

The standard 3" disc drive as fitted to the Einstein is a single sided 40 track drive with 10 sectors per track.



A single sided disc drive can only access one side of the disc media due to having only one read/write head, although the media may be used on both sides.

A sector is 512 bytes and this is the unit of transfer to/from the disc. It is not possible to transfer smaller units, larger units are transferred as an integral multiple of 1 sector.

A DOS or CP/M record is 128 bytes. DOS (or CP/M) function 20, (see the DOS/MOS reference manual pages 52 onwards), is provided to 'read sequential' one record and function 21 is to 'write sequential' one record. Having just read the above paragraph which states 1 sector, (512 bytes); is the unit of transfer this sounds like an anomaly. However when DOS receives the first instruction to read 1 record it actually reads 1 sector, (4 records), into its own buffer (which is at FE00 - FFFF Hex). A buffer is just an area of memory which is used to accept the incoming data. It then provides the 128 byte record from this buffer. The next three requests to read 1 record (sequentially) result in DOS providing the required record from the buffer. The next disc read is made only when a read is requested and the buffer has been emptied. A similar process takes place when writing sequentially. DOS builds the first four records to be written into the buffer and only when the buffer is full, is the buffer, (4 records), written to disc. This is why it is essential to close a file when writing to disc has finished, because it is very likely that there are 1, 2 or 3 records in the buffer. DOS will only write a sector if it has received less than 4 records when a file is closed. If the file is not closed the 1, 2 or 3 records in the buffer will NOT have been written to disc.

#### DISC ORGANISATION

The first two tracks are said to be reserved for the DOS (CP/M) operating system. This is not strictly true. Track 0 sector 0 and in some versions Track 0 sector 1 are for the BOOT program. This program is not part of DOS (CP/M), its purpose is to read in the operating system and pass control to the CCP, (Console Command Processor). This is the part of the operating system that analyses the user commands as typed in and process them. Once a user program is executing, the CCP has no work to do and can therefore be overwritten by the user program and its data.

This explains why some programs will terminate quietly whereas others cause the default drive to activate. When the drive activates it is re-loading the operating system because the CCP may have been overwritten. DOS occupies 4 or 5 sectors in track 0 (usually sectors 5, 6, 7, 8, and 9 depending on the version of the operating system.) and all sectors in track 1. Thus there are a few sectors in track 0 that are not used by the operating system.

Part of a copy protection scheme is to place some data into one or more of these unused sectors. The 'protected program' reads the sector(s) and if the expected data is not found then it assumes that the program is an illegal copy. This part of the protection scheme will cause a copy of a program, produced by COPY.COM (or BIP.COM), to fail because only the file has been copied and not the sectors in which the data has been planted. This scheme will not work when a disc is copied using a sector copier such as BACKUP.COM so an additional feature is required to prevent sector copiers from working. More details on this later.

#### THE DIRECTORY

This is an area of the disc that contains information relating to specific details of files or programs on the disc. This data is on track 2, remember we number from track 0, and uses the first 4 sectors (2k bytes).

To read the directory from a disc in drive 0, enter MOS and type:  
R 4000 4800 0002<E>  
To view the data use the tabulate command T:  
T 4000 4080<E>  
Each directory entry occupies 32 bytes which are as follows:

BYTE NO.	DESCRIPTION
00	User number or E5 if the file has been erased. On the Einstein, user areas are not implemented.
01 - 08	The file name.
09 - 0B	The file extension (e.g. .XBS)
0C	The extent number. An extent is 16k bytes. A file of 40k will have 3 directory entries. The first will have the extent no. 0; the second will have extent no. 1 etc
0D	Not Used.
0E	Used in CP/M to record the number of extents in the file.
0F	The number of 128 byte records in this extent. If the file is less than 16k this is the number of records in the file.
10 - 1F	Contains a list of the block numbers allocated to this extent The Einstein blocks are 2k bytes. (1k is more common) and 2 bytes specify the block number (1 is more common)
So a block number 1 appears as 01 00 (note reverse order) block number 2 appears as 02 00 etc.	

Since there are 16 bytes available for the block numbers there can be a maximum of 8 two-byte block numbers. This represents  $8 * 2k = 16k$ . This is why one directory entry represents a maximum of 16k i.e. one extent. The directory is a directory of extents and not files. Block 0 is used by the directory (2k bytes) and refers to Track 2 starting at sector 1. Block 1 is the start of the user data area and refers to track 2 starting at sector 4.

To calculate the track/sector number which corresponds to the start of a block:

1. Multiply the block number by 4, (4 sectors in 1 block).
2. Divide by 10, (10 sectors in 1 track).
3. Add 2 to the quotient to get the track no. (Block 0 starts at track 2).
4. The remainder is the sector number corresponding to the start of the block.

Conversely to find the block no. corresponding to any track/sector no.:

- 1: Subtract 2 from the track number, (block 0 starts at track 2).
- 2: Multiply the result of 1: above by 10, (10 sectors per track).
- 3: Add sector number.
- 4: Divide by 4, (4 sectors per block).
- 5: The quotient is the block number.

The program SCTED.COM, (part of EMSOFT DSCLONE disc), displays all the features described so far and includes a calculator for converting track/sector to block number and vice versa.

THE SECTOR NUMBERING SYSTEM and COPY PROTECTION

Early versions of DOS numbered the logical sectors in sequence. One track of 10 sectors was numbered:

05 06 07 08 09 00 01 02 03 04

The disadvantage of this is that when several sectors are to be read one after another by the time sector 00 has been read and saved in the buffer sector 01 has passed the read/write heads of the disc drive and so a long wait occurred before sector 01 was available.

The 'speed up disc' programs and later versions of DOS numbered the sectors:

00 05 01 06 02 07 03 08 04 09

So when sector 00 was written the minimum delay occurred before sector 01 was available. This sector numbering scheme is often referred to as sector interleaving or sector skew.

DOS expects sector numbers to be in the range 0 through 9.

BUT THE FLOPPY DISC CONTROLLER DOES NOT CARE WHAT NUMBERS ARE USED.

So the second part of a copy protection scheme is to format a disc so that periodically a sector number greater than 9 is used. Standard sector copying routines such as BACKUP.COM will fail to copy a disc once they encounter a non standard sector number. Thus by using some planted data in the 'unused' sectors in track 0 and incorporating some sector numbers greater than 9 in the formatting process we produce a copy protection mechanism that will cause any standard CP/M type copying programs to fail.

Cloning programs such as DSCLONE.COM, (see EMSOFT advert), analyse the sector numbers in the tracks of a disc to be copied and then format the receiving disc in exactly the same way. Only when the track of the receiving disc has been given an identical format to the source disc is the sector data copied.

However there are some more sophisticated methods available to 'copy protect' programs and these can be more difficult to detect. There are two approaches:

- 1: To make use of an undocumented feature of the disc controller.
- 2: To make a small change to the 'standard' pattern of bytes that exists between the data sectors.

SCTED.COM has a feature which is rarely found on other sector editors. It will give a display of the bytes which exist between the data sectors. There are 2 problems found when viewing this display and both are unavoidable due to the characteristics of the disc controller.
 

- a. The actual data bytes, which are displayed, are corrupted.
- b. Many of the bytes which exist between sectors are corrupted.

These 2 problems are of no great significance because the IMPORTANT bytes between the data sectors ARE CORRECTLY DISPLAYED and it is in these bytes that it is possible to make very small changes. The 'protected' program will analyse these bytes and if it fails to detect the small changes then it is assumed that it is an illegally copied program.

DISK FORMAT INFORMATION

The Floppy Disk Controller (FDC) on the Einstein is the WD 1770 series chip. When a disk is formatted the data sectors are filled with a marker byte, usually 0ESH. But what concerns us here is the information which goes between the data sectors. These areas are referred to as the inter-block gaps (IBG's). Here is the set of bytes which, it is recommended, should be written in the IBG.

No of Bytes	Hex Value	Comment
12	00	
3	F5	FDC writes 3 * A1
1	FE	ID address marker
1		Track No
1		Side No
1		Sector No * (0 - 9 is standard)
1		Sector length * 02 is standard = 2 * 256
1	FC	FDC writes 2 CRC chars
22	4E	
12	00	
3	F5	FDC writes 3 * A1
1	FB	Data address mark
512		DATA
1	F7	FDC writes 2 CRC chars
24	4E	

The CRC above is the Cyclic Redundancy Checksum. When data is written to disk a type of checksum is calculated AND written to disc. Subsequently, when the disk is read, the checksum is again calculated and compared with the checksum value read. If they match then all is well. If not then the message- 'BAD DATA' will be seen.

The 2 items marked with \*

1: SECTOR NUMBER as previously stated, the standard sector numbers are 0 through 9. However any single byte value can be used, but if this is done then any standard DOS (CP/M) program that attempts to read the file which contains the non standard sector numbers will fail with the 'NO SECTOR' message ( see the previous article on 'copy protection' schemes).

2: SECTOR LENGTH the standard value is 2, which means 2 units of 256 byte (giving a sector length of 512 bytes). But it is possible to include some non standard sector lengths in the formatting of a track. Once again any attempt to access a file containing a non standard sector length, when using normal DOS or CP/M programs will fail.

It must be said that a purpose built program can be written to read files which contain such irregularities.

Incidentally, it is stated that when a block of 3 \* F5 is sent to the FDC then 3 \* A1 will be written. This is not my experience, I have found that 2 \* A1 are written with another value to make up the 3 bytes.

\*\*\*\*\*

Another area for experiment is the 12 \* 00 's. It is certainly possible to alter this number slightly and this might be another candidate for using in copy protection schemes.

The format information can be examined using the SCTEDP program (select the T option). (see EMSOFT range for SCTEDP).

EINSTEIN CAN READ ALIEN DISKS.

The Amstrad 6128 uses 2 different sector numbering systems:

For DATA format disks the sector numbers are C1 through C9 ( a data format disk has no tracks for the operating system, thus it has a greater capacity). For SYSTEM format disks the sector numbers are 41 through 49. As you can see there are only 9 sectors per track on the Amstrad disks

To read a sector from an Amstrad DATA format disk which is in drive 0 Enter MOS and type

R 1000 11FF C100

this will read 1 sector (number C1 from track 0) into location 1000. If you have a 5 1/4" drive you can also read sectors from an IBM disc (40 track ).

DOUBLE SIDED DISKS ON THE EINSTEIN

Effectively track 1 is regarded as :

side 0, sectors 0 through 9 and  
side 1, sectors 0A through 13

and, of course MOS can be used to read any of these sectors.

How does the operating system know which drives are single sided, double sided or 40 track or 80 track? This information is kept at FBB1. The information is loaded in when a cold start is made. The 8 bits must be regarded as two quartets. The least significant 4 bits identify which drives are single or double sided:-

0000 all 4 drives are single sided  
0010 drive 1 is double sided, the rest are single sided  
0100 drive 2 is double sided, the rest are single sided

etc.

The four most significant bits are used to determine which drives are 40 or 80 track.

0000 all drives are 40 track  
0010 drive 1 is 80 track, the rest are 40 track  
0100 drive 2 is 80 track, the rest are 40 track  
1100 1000 drive 3 is 80 track, double sided and drive 2 is 40 track but double sided . Drives 0 and 1 are 40 track, single sided.

\*\*\*\*\*

THE DISK PARAMETER BLOCK (DPB) or how does DOS (CP/M) know how the disks are organised?

The DPB contains all the information relevant to the discs which can be used on the machine. The CP/M (DOS) function call number 31 can be used to find the start address of the DPB. E.g

LD C,1  
CALL 5

when these two instructions have been executed the HL register pair will contain the start address of the DPB. In DOS80 this address is at F8C0, where you will find 4 definitions for the 4 possible (different) drives which can be fitted. The details for each entry are:

Address	F8C0	Offset	Mnemonic	no. of byte	Explanation
		0	SPT	2	The no. of 128 byte units in 1 track [for single sided 10 sectors = 10*4 (128 byte units) = 40 = 24 00 HEX]
		2	B5H	1	Log(base 2) of no of 128 byte units in an allocation block. The block is 2k so there are 16 sets of 128 bytes in 1 block and log(base 2) of 16 = 4 (2*2*4 = 16)
		3	ASM	1	(No of 128 byte units in one block) - 1 [ 16 units in 1 block, 16 - 1 = 15 =0F]
		4	EXM	1	(No of extents which can be described in one FCB (i.e one dir entry)) - 1. As described in the first article only 1 extent can be described for each dir entry so 1- 1 = 0 for EXM]
		5	DSM	2	The largest block number. [s.s discs have 94 blocks = 5E 00 n.b LO - HI order d.s 40 track 194 blocks = C2 00 d.s 80 track 394 blocks = 8A 01 ]
		7	DRM	2	(Max no. of dir entries)-1 [dir entry is 32 bytes, 2k for ss discs so 2048/32 =127 =HEX 3F 00 4k for ds discs so 4096/32 = 127 = HEX 7F 00]
		9	ALO,1	2	Bitmap of dir block allocation [ss discs ALO ALI 1000 000 0000 0000 HEX 80 00 ds disks 1100 000 0000 0000 HEX C0 00 ]
		11	CKS	2	No of dir entries per sector [512 bytes per sector, 32 bytes per dir entry, so 512/32 = 16 = HEX 10 00 ]
		13	OFF	2	No. of reserved tracks ( for Op. Sys.) [s.s disks : 2 tracks are reserved so the value is 02 00 d.s. disks : 1 track (2 sides) is 01 00 ]

Well I know some of this has been heavy going but I have received a number of requests for this information. Furthermore, if you found it heavy going try casting your mind back. Have you not had several experiences when something looked impossibly difficult which you now find easy?

Finally I must pay tribute to the originators of CP/M. It is a very simple yet powerful operating system suitable for any Micro with a 8080 or 280 processor. Perhaps the most important aspect (which we now realise is obvious) is that the machine dependent parts of the operating system are isolated in the one program unit called the BIOS. Actually this is not quite the case for DOS 1.3 or DOS80 which do not hold their DPB's in the BIOS. In this respect these versions are more akin to the earlier versions of CP/M before 2.2. Fortunately other aspects of these versions of the operating system are superior to CP/M 2.2. The greater freedom to change disks makes things much easier.

For further reading on this topic I strongly recommend:

Mastering CP/M by Alan R. Miller  
published by SYBEX

CP/M The Software Bus by A. Clarke et al.  
published by SIGMA Technical Press

## INTRODUCTION TO THE OPERATING SYSTEM BY PHILLIPE HENRIE.

### 1. Introduction

When you switch on Albert, all this electronic machine receives power, and the heart of the system, a Z80A central processing unit (cpu), reads memory chips from location 0000H in ROM, begins to set up relations between the cpu and the hardware attached to the system. At last, you are allowed to act, the computer is ready. What happened since power on has no importance, but indeed, something did happen: a program called an operating system is running now. I call it the Big Operating System, or BOS.

#### 1.1 What is BOS?

BOS is a program that may have several names, such as CP/M, XTALDOS, DOS, Z-COM, CONIX, QP/M etc...

Don't be disturbed by all these different names, they are just different versions of BOS, but they all have the same functions, which are to manage the basic resources of a computer system:-

- devices
- information
- memory
- processes and processors.

### 1.2 Virtual machine concept

The aim of the BOS, is to create a virtual machine, that is, a machine which exists at a higher level of abstraction than the physical machine which is supporting it. The virtual machine concept is used to raise the details of interfacing with a machine to a higher level of abstraction so that the intimate details of a machine need not be of concern. This virtual machine, on which all BOS software runs (including BOS itself), always behaves in the same manner and has the same logical interface to application programs, regardless of the actual hardware used to implement it. This permits the exchange of software regardless, by and large, of the actual computer hardware involved.

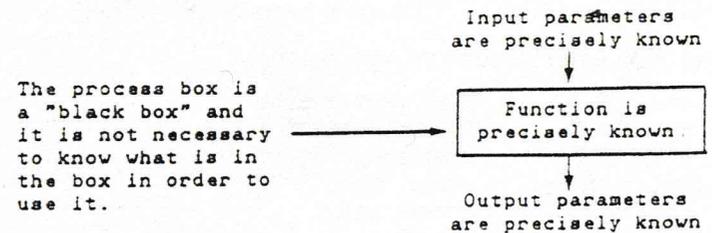
### 1.2 Process box concept

A process box concept is used to raise at a higher level a process, regardless of how the process performs, but only knowing what the process does and how to communicate with it.

BOS has several built-in functions, they are seen as a process box, and this contributes to the virtual machine concept seen from the software point.

Indeed, if the software communicates with BOS, respecting the process box conventions, this makes such software transportable, at the binary level, between any computers running BOS. The only proviso is that the application program must use the standard BOS function calls when requesting service, otherwise, it may result in a loss of portability to some systems.

The process box may be illustrated as follows:-



## 2. DEVICES MANAGEMENT.

The beauty of BOS is the way it does device and information management. Device management is performed by the Basic Input Output System (BIOS) of BOS, the BIOS provides a standardised, hardware-independent interface to the devices attached to the computer. The routines controlling these devices may be accessed by way of a table of jump instructions located at the beginning of the BIOS. The parameters passed to these routines and the parameters returned by these routines are precisely defined. The application programmer does not need to know how the routines perform their functions, but only what they do and how to communicate with them. This is the process box concept.

The BIOS performs all of the functions necessary for BOS (and the programs which run under BOS) to control and communicate with the disc subsystem and most commonly used peripherals, creating the virtual machine. These functions are:

### 1. Initialisation functions;

- \*Cold boot initialisation (when the system is first turned on).
- \*Warm boot initialisation (performed periodically after the system is turned on).

### 2. Character input/output functions;

- \*Console status (check for availability of a character at the console).
- \*Console input
- \*Console output
- \*List status (check to see if List device is ready to output the next character).
- \*List (printer) output.
- \*Auxiliary output.
- \*Auxiliary input.

### 3. Disc input/output functions;

- \*Home drive (move head to track 0).
- \*Select drive (which drive to use).
- \*Select track.
- \*Select sector.
- \*Select memory address to read into or write from.
- \*Read block (at selected track and sector) into memory (at selected memory address).
- \*Write block (at selected track and sector) from memory (from selected memory address).
- \*Logical-to-physical sector translation (for efficiency of disc use).

The standard BIOS jump instructions look like this:

xx00	jp BOOT	;"Cold" (first time) bootstrap
xx03	jp WBOOT	;"Warm" bootstrap
xx06	jp CONST	;Console input status
xx09	jp CONIN	;Console input
xx0C	jp CONOUT	;Console output
xx0F	jp LIST	;List output
xx12	jp PUNCH	;"Punch" output
xx15	jp READER	;"Reader" input
xx18	jp HOME	;Home disc head (track 0)
xx1B	jp SELDSK	;Select logical disc
xx1E	jp SETTRK	;Select track number
xx21	jp SETSEC	;Set sector number
xx24	jp SETDMA	;Set DMA address
xx27	jp READ	;Read (128 byte) sector
xx2A	jp WRITE	;Write (128 byte) sector
xx2D	jp LISTST	;List device output status
xx30	jp SECTRAN	;Sector translate

where xx is the high byte address of the beginning of BIOS.

## 3. INFORMATION MANAGEMENT

In the BOS context, it consists of the control of files on disc. BOS shines here too, extending the virtual machine concept to the management of files on disc. The Basic Disc Operating System (BDOS) portion of BOS creates this file-oriented virtual machine. To illustrate this point, some (but not all) of the functions provided by BOS are:

- \*Reset disc system
- \*select disc
- \*Create file (actually, create a directory entry for a file)
- \*Open file (make a file ready for reading or writing)
- \*Close file (terminate the read/write process)
- \*Delete file
- \*Rename file
- \*Set memory address to read into or write from
- \*Read next block from file
- \*Write next block into file

note the similarity between these BDOS functions and the BIOS disc functions. These BDOS functions are accessed in a different way from the BIOS, but the process box concept is maintained. All one needs to know are the input parameters, the output parameters and what the function performed is. Once more, transportability is realized at the binary level, but this time it is with respect to the information manipulated by the computer (a more general concept than merely performing disc and character I/O).

The BDOS functions are:

### CODE/DESCRIPTION

#### *Simple byte-by-byte Input/Output*

- 0 Overall system and BDOS reset
- 1 Read a byte from the console keyboard
- 2 Write a byte to the console screen
- 3 Read a byte from the logical reader device
- 4 Write a byte to the logical punch device
- 5 Write a byte to the logical list device
- 6 Direct console Input/Output
- 7 Read the current setting of the IOBYTE
- 8 Set a knew value for the IOBYTE
- 9 Send a "\$" terminated sting to the console
- 10 Read a string from the console into the buffer
- 11 Check if a console key is waiting to be read
- 12 Return the CP/M version number

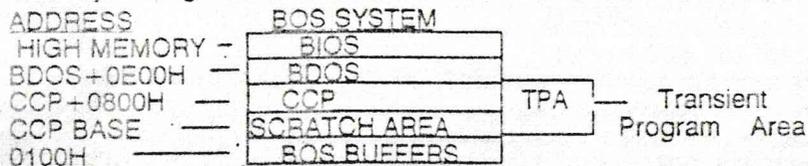
#### *disc file Input/Output*

- 13 Reset disc system
- 14 Select specified logical drive
- 15 Open specified file for reading/writing
- 16 Close specified file after r/w
- 17 Search file directory for 1st name match

\*\*\*\*\*

- 18 Search directory for next name match
- 19 Delete (erase) file
- 20 Read the next "record" sequentially
- 21 Write the next "record" sequentially
- 22 Create the new file with the specified name
- 23 Rename a file to a new name
- 24 Indicate which logical discs are active
- 25 Return the current default disc drive number
- 26 Set the DMA address (r/w address)
- 27 Return the address of an allocation vector
- 28 Set specified logical disc drive to Read-only
- 29 Indicate which discs are currently Read-only
- 30 Set specified file to System or Read-only
- 31 Return address of disc parameter block
- 32 Set/get the current user number
- 33 Read a "record" randomly
- 34 Write a "record" randomly
- 35 Return logical file size
- 36 Set record number for the next random r/w
- 37 Reset specified drive
- 38
- 39
- 40 Write a "record" randomly with zero fill

**4. MEMORY MANAGEMENT** BOS does very little in the way of memory management. It merely defines the basic memory structure as follows:



**4.1 THE CONSOLE COMMAND PROCESSOR** The Console Command Processor (CCP) is the first part of BOS that is encountered going "up" through the memory address. This is significant when you consider that the CCP is only necessary in-between programs. When BOS is idle, it needs the CCP to interact with you, to accept your next command. Once BOS has started to execute the command, the CCP is redundant; any console interaction will be handled by the program you are running rather than by the CCP. Therefore, the CCP leads a very jerky existence in memory. It is loaded when you first start BOS. When you ask BOS, via the CCP, to execute a program, this program can overwrite the CCP and use the memory occupied by the CCP for its own purposes. When your program has finished, BOS needs to reload the CCP, now ready for interaction with you. This process of reloading the CCP is known as a "warm boot". In contrast with the cold boot mentioned before, the warm boot is simply reloading the CCP. The BDOS and BIOS are not touched. The running program tells BOS that it has finished initialising itself

\*\*\*\*\*

during the cold boot routine, it puts an instruction at location 0000H to jump to the warm boot routine, which is also in the BIOS. Once the BIOS warm boot routine has reloaded the CCP from the disc, it will transfer control to the CCP.

**4.2 THE TRANSIENT PROGRAM AREA** The Transient Program Area, is the memory available for programs.

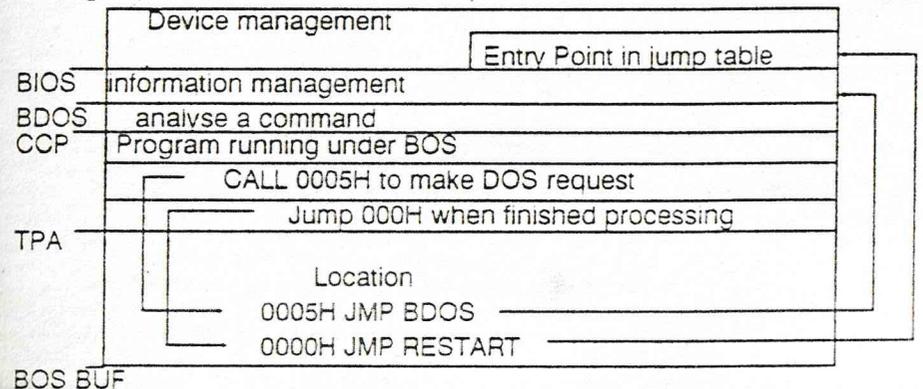
**4.3 THE BUFFERS** This is the base page of BOS. It contains important values, for the interaction between the three main parts of BOS, which communicate among themselves, and for the way in which programs running under BOS interacts with the BDOS.

ADDRESS	BOS BUFFERS
0000H	Jump to warm boot entry in BIOS
0002H	BIOS jump vector page
0003H	Input/Output redirection byte (IOBYTE)
0004H	Current user and default logical disc
0005H	Jump to BDOS entry
0007H	Top page of usable memory
005CH	File control block
006CH	File control block
0080H	Default address to/from which data is read/write (known as DMA address)

BOS does not allocate sections of the memory to application programs or do any of the partitioning usually associated with the memory management function.

**5. PROCESS MANAGEMENT** BOS does no process management. There is only one processor, so no process management is needed. Only one process at a time can run, so BOS simply starts the process and then relinquishes all control. The process then has complete control over the entire computer and BOS does nothing to stop it from doing anything it wishes to do.

**6. CONCLUSION** The following figure summarizes the BOS functional framework or the way each part of BOS communicates among themselves, to manage the basic resources of a computer.



\*\*\*\*\*

*Applications to operating systems.*

Introduction: Here we will examine several BOS and compare them with the standard one, CP/M. As CP/M was the first BOS for the 8 bit system, it became the standard with vast amounts of software written under it. So, understanding how CP/M is built, will help us to: adapt it adapt some software install it on our own system other BOS were written, but all take CP/M as the kernel and bring in their own features. I'll try to explain the differences with the standard that sometimes cause software to crash.

2. Memory management: The standard length of each part of BOS are: CCP 800 hex (constant) BDOS E00 hex (constant) BIOS 900 hex (variable) The length of CCP and BDOS are constant and any true CP/M compatible BOS must respect these lengths, otherwise software may crash. look at the table below, which shows you the location for different BOS. As you can see, CCP and BDOS length is not always respected, and this can be the reason for crashes using some software under the non-compatible BOS.

BOS	CCP	BDOS	BIOS	COMPATIBLE
CP/M2.2	E200	EA00	F800	YES
XTALDOS				
1.11	E300	EB00	FA00	NO
1.31	E300	EC00	FA00	NO ONLY FOR CCP
2.02	E200	EB00	FA00	YES
DOS80				
V1.0	E200	E800	F600	NO
ZDOS				
1.6 V4.1x	E000	E800	F600	YES

3. The Console Command Processor: Here too, each BOS has its own commands, that are sometimes more powerful than the standard CCP in their use

FUNCTION	CP/M	XTALDOS	ZDOS
DIRECTORY OF FILES	DIR	DIR	DIR
ERASE A FILE	ERA	ERA	ERA
RENAME A FILE	REN	REN	REN
PRINT FILE TO CONSOLE	TYPE	DISP	TYPE
SAVE MEMORY INTO FILE	SAVE	SAVE	SAVE
CHANGE DISC DRIVE	D:	D:	D:
CHANGE USER	USER u	PSW	D: u
ENABLE PASSWORD	NO EQUIV	PSW	PASS
DISABLE PASSWORD	NO EQUIV	PSW	NORM
JUMP TO 100HEX	NO EQUIV	GO	GO
SET DEFAULT DRIVE	NO EQUIV	DRIVE	NO EQUIV
SET DEFAULT USER	NO EQUIV	PSW	DFU
LOAD FILE INTO MEMORY	NO EQUIV	LOAD	LOAD
JUMP TO LOCATION	NO EQUIV		JUMP
LOCK A FILE	NO EQUIV	LOCK	NO EQUIV
UNLOCK A FILE	NO EQUIV	UNLOCK	NO EQUIV
GO TO MOS	NO EQUIV	MOS	NO EQUIV

4. The BDOS functions: XTAL DOS uses two functions, code number 38 and 39, not used under CP/M2.2, to perform "return to MOS" (38), and "set password" (39). Otherwise, the standard functions are respected. The first bytes of XTAL DOS are not compatible with CP/M since the first six bytes contain a serial number. BDOS beginning at xx06, thus XTALDOS has not respected convention, this may explain crasnes of some software under this "not truly" compatible BOS.

\*\*\*\*\*

5. The standard BIOS jump vector: As in the memory management, the BIOS of some BOS doesn't respect the standard. So, some software will crash when it attempts to call a BIOS routine that is not implemented. This is the case of the LISTS jump in XTAL DOS 1.11 and 1.31, and in DOS.80v1.0 also in the case of the SECTRAN jump in all XTAL DOS versions and DOS 80v1.0.

ADDRESS	CP/M	XTALDOS1.11	ZDOS
xx00	JP BOOT	JP BOOT	JP BOOT
xx03	JP WBOOT	JP WBOOT	JP WBOOT
xx06	JP CONST	JP CONST	JP CONST
xx09	JP CONIN	JP CONIN	JP CONIN
xx0C	JP CONOUT	JP CONOUT	JP CONOUT
xx0F	JP LIST	JP LIST	JP LIST
xx12	JP PUNCH	JP PUNCH	JP PUNCH
xx15	JP READER	JP READER	JP READER
xx18	JP HOME	JP HOME	JP HOME
xx1B	JP SELDSK	JP SELDSK	JP SELDSK
xx1E	JP SETTRK	JP SETTRK	JP SETTRK
xx21	JP SETSEC	JP SETSEC	JP SETSEC
xx24	JP SETDMA	JP SETDMA	JP SETDMA
xx27	JP READ	JP READ	JP READ
xx2A	JP WRITE	JP WRITE	JP WRITE
xx2D	JP LISTST	NO	JP LISTST
xx30	JP SECTRAN	NO	JP SECTRAN

6. Conclusions: As we have seen, to be a truly CP/M compatible BOS, some rules must be respected in the general structure of the BOS. We have seen just some of these rules, but others exist which are not followed. To build a new truly CP/M compatible BOS a full understanding of the interface between the physical machine and the virtual machine run under BOS is essential.

\*\*\*\*\*

INTERRUPTS AND POLLING

Philip Eley brings us a tale of gardeners, postmen and fishmongers????

If you'd always thought that polling was something to do with voting in an election, and that an interrupt was something like, 'the phone ringing whilst you were taking a bath', then you'd be right. But, like so many other words, these two words have been hi-jacked from their familiar surroundings and applied to specific computer concepts, acquiring an air of mystery in the process. In computer terminology interrupts and polling are alternative ways of detecting when something has changed, for example, when a key is pressed or a printer has run out of paper. This article is an attempt to de-mystify these two concepts, with the aid of a problem from real life.

Imagine the following: I want to spend the day gardening; I am also expecting a very important letter to arrive, but I can't see or hear the letter-box while I'm gardening. How do we solve the problem of gardening yet still getting the letter soon after it has arrived? For the purposes of this explanation, I'll look at two possible solutions.

The first is to stop gardening every few minutes, walk into the house, and check if the letter has arrived. Thus I will be able to do two things at once, although gardening will obviously take longer to complete due to the time involved in checking the letter box. The alternative solution is to leave a note for the postman asking him to ring the bell when he delivers the post. That way I can spend all my

time gardening until I hear the doorbell, with no unproductive activity. Also, I will know immediately that the letter has arrived - if I had used the first method and the letter arrived just after one of my checks, it may be five or ten minutes before I was aware of its arrival.

These two solutions represent polling and interrupts respectively. From a computer viewpoint, the processor does the polling (equivalent to the gardener checking the letter-box), but an interrupt comes into the processor from an external source (the postman rings the bell). The Machine Operating System, (MOS)), in the Einstein uses a mixture of polling and interrupts to control the peripherals attached to the Z80 microprocessor. Two examples from MOS: the keyboard is polled, the timer, which updates the real time clock, generates interrupts. Deeper examination of these examples should further clarify the subject.

Polling of the keyboard takes two distinctive forms depending on the reason a key might be pressed. Like any computer, the Einstein spends a great deal of time waiting for commands or data to be input from the keyboard. Nothing else can happen until the user types something, so a form of polling is used which does nothing else but check if a key has been pressed. In the postman example this is equivalent to sitting next to the letter-box doing nothing else but waiting for the letter to arrive. It wastes lots of time, but that's okay for the computer which, as I've already said, has nothing else to do anyway.

Once the commands to run an XBAS program have been typed, the keyboard is still polled, but only after each BASIC instruction has been executed. XBAS checks in case SHIFT/BREAK has been pressed, stopping the program if this happens. This is the exact equivalent of the gardener checking the letter-box every few minutes and has the same drawbacks: time is wasted checking the keyboard, and the keypress will not be picked up immediately, although, in practice, the delay is hardly noticeable due to the speed of the Z80. Balanced against the slight loss of execution speed is the ability to break into a program that might have got stuck in an infinite loop.

The example of the timer demonstrates how certain effects may only be achieved through the use of interrupts; in this case the provision of a real time clock, updated every second regardless of what program (if any) is running. For a start, think about how to program such a clock without interrupts. The only approach is to set up a loop which takes exactly one second to execute, then update the clock, then go into the loop again. This is quite feasible to program but it has the big disadvantage of completely excluding any other processing - imagine trying to keep the timing correct whilst doing other things inside the loop. The amount of work needed to add this to every program would be astronomical, so we can effectively rule out this idea.

The programming problems are drastically simplified with the introduction of a hardware interrupt. The price paid is more complicated (and thus more costly) hardware circuitry - in this case a timer which interrupts every second. The processor detects each interrupt and executes a small chunk of assembly code built into the operating system. The interrupts will continue to update the clock, stealing only the required amount of time from whatever program is running; the program will be unaware that anything has happened unless, of course, it is looking for a change in the real time clock. The clock is simply there when needed, and may be ignored by a program if not required. Thus interrupts make the clock a practical proposition.

Having covered the differences and uses of polling and interrupts I'll now dig deeper into the MOS and describe the Einstein's interrupt system in more detail. There are a couple of other aspects to be covered, and I will introduce you to a travelling fish-salesman, who will help with the explanations. Read on....

Back to our example, we will extend the original scenario thus: apart from the postman, I might receive a visit from Fred the travelling fish-salesman. I don't always wish to buy fish, but Fred will always ring the doorbell "just in case". We now have a situation where the doorbell might ring for one of two reasons - the postman or the fishmonger. Two different courses of action are required depending on the person ringing. If it is the postman, I don't have to open the door (he is not expecting me to), just collect the post; if it's Fred, he will expect me to answer, even if it is only to say that I don't require any fish. Because of these two alternate courses of action it would be handy if I could arrange for two different "signals" e.g. the postman to give one ring, Fred to give two. (Ed. The postman never knocks twice).

The Z80 processor used in the Einstein is capable of handling 128 different interrupts and so it is not just "handy" to be able to distinguish the source of a particular interrupt, it is imperative in order that the right course of action may be followed. More of how that's done later.

Having trained the postman and Fred (and any other potential doorbell ringers) to use a code (the number of rings), I wish to have a lazy day in bed, without being interrupted by the doorbell... so I disconnect the doorbell from its battery. In computer terminology this is known as DISABLING the interrupts. Notice that this stops ALL the interrupts. It also be desired to stop particular interrupts, whilst letting others happen. In our real-world example, this is equivalent to pinning a note to the door asking that only the milkman should ring.

Back to the Einstein. What follows is a description of the way the hardware and software are prepared to deal with interrupts. This is part of the MOS.

When the machine is switched on, or when the reset button is pressed, the micro-processor starts executing the machine code resident in the ROM. At this point all interrupts are disabled, because the so called Interrupt Service Routines (ISRs) have not been set up, and this is one of the first tasks of the MOS. This consists of copying a table of addresses to a known place in the RAM. The address of this table is placed in a special register in the Z80 (the I-register); if you type Z2 at the MOS prompt, you will see, under the I, the actual value of FB (hex) giving a base address of FB00 (hex). This is used, whenever an interrupt occurs, in conjunction with an offset provided by the interrupting device (equivalent to the number of rings made by the postman or fishmonger). The Einstein uses only two interrupts (of the potential 128) - the timer, and the parallel port, the actual offsets being 06 and 10 (hex) respectively. These offsets, along with other parameters, are programmed into the devices during this initialisation process. The addresses in the copied table are the actual addresses of the ISRs which deal with the interrupts (in MOS 1.21 these are FC17 and FC84 (hex)). The actual ISRs are also copied from the ROM into the RAM at the relevant addresses.

\*\*\*\*\*

You may wonder why the table and routines are copied into RAM, instead of being left in the ROM. There are two reasons: by placing them in RAM, they can easily be changed by a program which has its own ISR; more importantly, the architecture of the Einstein is such that the addresses in the ROM are not directly available during normal operation, because they duplicate those of the lower part of the RAM, which is switched in after initialisation. The higher addressed RAM is not switched, thus is available to all programs whether in ROM or RAM. Having completed its initialisation, the MOS enables interrupts, the immediate effect being to start the real time clock; driven, as I've already said, by interrupts from the timer. The user is now in control, and the MOS commands may be used, or DOS loaded and used. Irrespective of what is done, the timer is still updated every second. We can now complete the explanation by following the actual path of one of these timer interrupts.

The timer, after each second has elapsed, sends an interrupt to the Z80. In addition, the timer sends the pre-programmed offset (06) which identifies it as a timer interrupt. The Z80 uses its I-register (containing FB (hex)) to form an address in the ISR address table, in this case FB06 (hex). It then takes the contents of FB06 and FB07, forming them into the ISR address (FC47) - note that addresses are formed in reverse order (low byte first); FB06 contains 47, FB07 contains FC. Before jumping to this address, further interrupts are automatically disabled, and the address of the instruction which would have been executed had not the interrupt occurred is saved (on the stack) so that the interrupted program can be resumed after dealing with the interrupt.

The ISR must first save the contents of any registers that it is going to change; they will be restored unchanged when the interrupted program is resumed. The ISR for the timer, like all ISRs, is very simple and quick. It updates a counter of seconds. If the counter reaches 60 then it is reset and a counter of minutes is incremented. In a similar fashion, when 60 is reached on the minute counter, the hour counter is incremented and the minute counter is reset, likewise the hour counter when 24 is reached. Once the processing is finished, the saved registers are restored, interrupts are enabled, and the interrupted program is resumed exactly where it was interrupted, without any ill effects. And that's all there is to it.....

**ADDING A PRINTER DUMP TO DOS 1.31**

(Graham Bettany - pinched from Chris Giles!)

Here is the reply to the letters page request, this will only work with DOS 1.31. The dump uses printer control codes for the Tatung printer but should also work with Epson compatible units. The control codes start from location 81E2.

- 1 ... Insert disc for modification to drive 0.
- 2 ... MOS<E>.
- 3 ... R 8000 99FF<E>
- 4 ... M 8170<E>.
- 5 ... Enter the following code from 8170 onwards :-

```
8170 DB 20 E6 1C FE 10 C0 2A
8176 9A FB E5 2A 9C FB E5 21
8180 EF 00 22 9C FB 21 00 00
8188 22 9A FB E5 DD E1 E5 FD
8190 E1 DE E5 06 08 21 E1 E2
```

\*\*\*\*\*

```
8198 7E CF 9F 23 10 FA FD E5
81A0 0E 01 C5 CF C7 C1 28 01
81A8 37 CB 11 FD 28 30 F3 79
81B0 CF 9F FD E1 DD 23 DD E5
81B8 F1 B7 28 E2 DD E1 01 F8
81C0 FF FD 09 3E 0A CF 9F FD
81C8 E5 E1 01 BF 00 09 38 C1
81D0 3E 1B CF 9F 3E 40 CF 9F
81D8 E1 22 9C FB E1 22 9A FB
81E0 C9 OD 1B 41 08 1B 4B 00
81E3 01.
```

The spaces between the bytes are only for clarity, press enter after the end of each line and use the full stop to exit the Modify command.

6 .. W 8000 99FF<E>

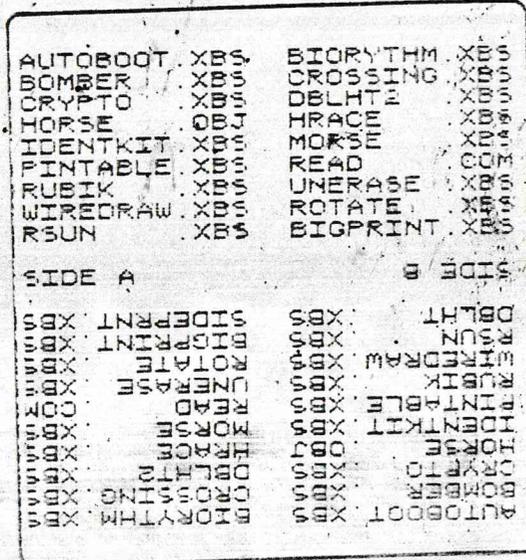
You now have a DOS with a printer dump program that will be automatically loaded into memory on pressing CTRL-BREAK. To use from XBAS you will need to add a couple of lines:

At the start of your program -

```
CLEAR &E26F
and to use the dump
CALL &E270.
```

**AUSSIE PRINTING?**

An unusual program by Stuart Marshall that enables the printing of disc labels on both sides the right way up on one pass. If you didn't understand that statement then a quick look at the example print below should make things a little clearer.



The program is written in XBAS running under Dos 1.31, it uses a graphic dump to print the characters and Stuart has modified the Dos to contain the dump as previously printed or in the Compendium. Line 10 Clears memory so Xbas will not overwrite the dump code and it is called from line 300, you may have to experiment

\*\*\*\*\*

with 32 and 40 column mode to get the correct spacing on the printout for your printer by altering line 1000. The program reads the directory from the disc and prints its contents automatically on the label, saves typing and spelling! For those of you who have not got a modified Dos, here is how to add the dump to 1.31, alternatively you could load any other dump programs you have making sure lines 10 and 300 are changed to call the correct locations. It is advisable to use a spare disc for this as it is very easy to get wrong and you will probably need to reformat the disc if it screws up!

To add a dump file to Dos 1.31 go into MOS;

Type MOS<E>

Type M 0100<E>

Type in the following code with Enter after each complete line

```
DB 20 E6 1C FE 10 C0 2A
9A FB E5 2A 9C FB E5 21
BF 00 22 9C FB 21 00 00
22 9A FB E5 DD E1 E5 FD
E1 DD E5 06 09 21 71 A0
7E CF 9F 23 10 FA FD E5
0E 01 C5 CF C7 C1 28 01
37 CB 11 FD 2B 30 F3 79
CF 9F FD E1 DD 23 DD E5
F1 B7 28 E2 DD E1 01 F8
FF FD 09 3E 0A CF 9F FD
E5 E1 01 BF 00 09 38 C1
3E 1B CF 9F 3E 40 CF 9F
E1 22 9C FB E1 22 9A FB
C9 OD 1B 41 08 1B 4B 00
01 .
```

Do not forget the full stop after the last number.

Type R 8000 9800<E>

Type C 0100 0178 8170<E>

Type M 8196<E>

Type E1E2.<E> (Full stop after the 2)

Type W 8000 9800<E>

That's it!

To check all is well do G E270<E> (A dump of the current screen should print)

From Xbas use Clear &E26F and Call &E270.

\*\*\*\*\*

Having modified the DOS for inclusion of the dump we can type in the actual program:

```
1 REM *** LABEL PRINTER ***
2 REM *** STUART MARSHALL, 25 CARLC
ROFT, STONYDELPH, TANWORTH, STAFFS, B77
4DL. ***
3 REM *** TESTED ON MICRO PRO CPASO
***
4 REM *** DOS 1.31 WITH GDUMP ***
5 REM *** XBAS 4.2 ***
6 REM *** POSSIBLE CLS40 IN LINE 10
00 ***
10 CLEAR &E260
20 CLS40:PRINT@4,10,"PLEASE WAIT SET
TING CHARACTER SET"
30 FOR F=6400 TO 7152 STEP 8
40 C=0
50 FOR G=0 TO 7
60 A(G)=VPEEK(F+G)
70 A$=BIN$(A(G),8):B=0
80 IF MID$(A$,1,1)="1" THEN LET B=B+4
90 IF MID$(A$,2,1)="1" THEN LET B=B+8
100 IF MID$(A$,3,1)="1" THEN LET B=B+16
110 IF MID$(A$,4,1)="1" THEN LET B=B+32
120 IF MID$(A$,5,1)="1" THEN LET B=B+64
130 IF MID$(A$,6,1)="1" THEN LET B=B+128
140 A(G)=B:NEXT G
150 FOR G=7 TO 0 STEP -1
160 VPOKE (F+1024+C),A(G):REM ASCII C
DE+1024 GIVE INVERTED LETTER*GRAPH/LETTE
R.(LOWER CASE) SHIFT/GRAPH/LETTER (UPPER
CASE)*
170 C=C+1
180 NEXT G,F
190 BEEP2
195 CLS40:PRINT@6,10,"PLEASE INSERT D
ISK SIDE A ":PRINT@9,12,"THEN PRESS SPAC
E BAR"
196 A$=INCH$:IF A$<>" " THEN 195.
200 GOSUB 500
```

\*\*\*\*\*

```

210 CLS40:PRINT@4,10,"PLEASE TURNOVER
DISK TO SIDE B":PRINT@9,12,"THEN PRESS
SPACE BAR"
220 A$=INCH$:IF A$<>" "THEN220
230 GOSUB600
240 GOSUB1000
300 CALL&E270
310 CLS40:PRINT@4,10,"PRINT ANOTHER L
ABEL (Y/N)";A$=INCH$:IFA$<>"Y"THEN40
GOTO220
320 PRINT#1:PRINT:PRINT:PRINT#0
330 RUN 190
400 CLS40:END
500 CLS:DIR
510 FORA=1TO10
520 X$(A)=MID$(SCRN$(A),4,12)+" "+RIG
HT$(SCRN$(A),23)
530 NEXTA
531 RETURN
540 FOR A=1TO10:PRINT@0,A,X$(A):NEXTA

600 CLS40:DIR
601 DIM L(26,10),B(26)
610 FORA=1TO10
620 Z$(A)=MID$(SCRN$(A),4,12)+" "+RIG
HT$(SCRN$(A),23)
630 FORB=1TO26
640 L(B,A)=ASC(MID$(Z$(A),B,1))+128:N
EXTB:NEXTA
641 RETURN
1000 CLS32:FOR A=0TO9:PRINT@0,A,X$(A):
NEXTA
1010 PRINT@0,11,"SIDE A":REM LINE 12
1020 PRINT@20,11,CHR$(66+128);" ";CHR$(
69+128);CHR$(68+128);CHR$(73+128);CHR$(
83+128):REM * PRINT SIDE B UPSIDE DOWN *

1030 A=1:FOR S=22TO13STEP-1:D=-1:FOR B=
26TO1STEP-1:D=D+1
1040 PRINT@D,S,CHR$(L(B,A)):NEXTB:A=A+
1:NEXTS
1050 RETURN
    
```

\*\*\*\*\*

COL256

A COLOUR SCROL DEMO FOR THE 256 FROM  
M.R.V. PETERSON

Having recently aquired the data manual for the Yamaha V9938 display processpr as used on the Einstein 256 I wrote this short routine to demonstrate the 256 colours available in mode 7, a mode unsupported by BASIC.

WHAT.IT DOES:

This routine runs as a.COM file from DOS, changes to the 256 colour screen mode, fills the screen with a pattern and scrolls it continuously.

Pressing A,B,C,Q or E changes the pattern, unless the pattern selected is already current. R reverses the direction of scrolling and Q leaves the program and returns to DOS.

To enter, type.. MOS <E> where <E> is the ENTER key and then type.. M0100 <E> followed by

0100	21	8F	01	CD	C5	01	AF	0E	0E	CD	BD	01
010C	AF	D3	09	3E	40	D3	09	1E	00	16	00	7A
0118	AB	4F	CD	AA	01	D3	08	14	20	F5	1C	20
0124	F0	06	00	78	0E	17	CD	BD	01	CD	9D	01
0130	05	20	F4	CF	9B	F6	20	FE	61	20	07	3E
013C	AB	32	18	01	18	C4	FE	62	20	07	3E	A3
0148	32	18	01	18	B9	FE	63	20	07	3E	B3	32
0154	18	01	18	AE	FE	64	20	07	3E	93	32	18
0160	01	18	A3	FE	65	20	07	3E	83	32	18	01
016C	18	98	FE	72	20	16	3A	30	01	CB	47	28
0178	07	CB	87	32	30	01	18	A5	CB	C7	32	30
0184	01	C3	25	01	FE	71	20	9B	CF	BE	C9	0D
0190	0E	40	1F	00	00	F4	1E	00	0B	82	00	01
019C	00	F5	D5	11	00	04	1B	A2	B3	20	FB	D1
01A8	F1	C9	C5	CB	01	CB	17	06	07	CB	01	CB
01B4	01	CB	01	CB	17	10	F6	C1	C9	D3	09	79
01C0	CB	FF	D3	09	C9	0E	00	46	23	7E	CD	BD
01CC	01	0C	10	F8	C9	.	.	.	.	.	.	.

Have fun and if you know anyone with an Atari ST, show it to them and say "Do that in 224 bytes!"

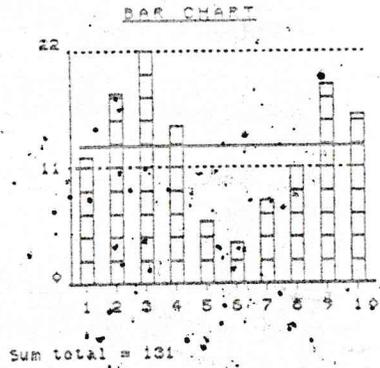
BAR CHART AND PLOTTER

Here are a couple of routines written by David Williams that may prove useful if you run a club or small business.

Barchart allows you to enter 10 values that will be pictorially represented.

```

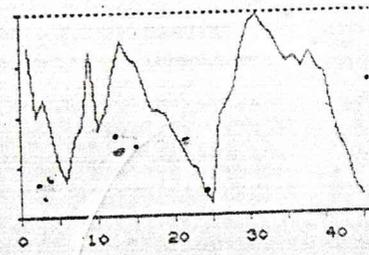
10 RST:BCOL2:TCOL1,0:GCOL1,0
20 GOSUB 440
30 FOR X=1 TO 10
40 PRINT@5,3+X;"Input No.:";X;
50 INPUT"=";Z(X):S=S+Z(X)
60 IF B<Z(X) THEN B=Z(X)
70 IF B=0 THEN B=1
80 NEXT:AV=INT(47+S/10*120/B)
90 GOSUB 440.
100 FOR Q=47 TO 170 STEP 60
110 DRAW 58,Q TO 216,Q,2:NEXT
120 DRAW 220,47 TO 59,47 TO 59,168
130 PRINT@8,19;"1 2 3 4 5 6 7 8 9 10"
140 FOR A=67 TO 220:STEP 16
150 DRAW A,47 TO A,44:NEXT:GCOL1,11
160 FOR N=1 TO 10:X=X+16
170 IF Z(N)=<0 THEN 250
180 C=48+(120*Z(N)/B)
190 IFC>50 THEN C=C-1
200 DRAW 37+X,48 TO 37+X,C:REM UP
210 DRAW 37+X,C TO 44+X,C:REM ACROSS
220 DRAW 44+X,C TO 44+X,48:REM DOWN
230 FOR V=47 TO C STEP 12
240 DRAW 37+X,V TO 44+X,V:NEXT
250 NEXT N:GCOL4,2
260 DRAW 64,AV TO 215,AV:POKE64335,40
270 SPRITE0,220,AV+4,4,65
280 PRINT@8,17,"0":A=.1*INT(B)
290 AS=STR$(5*A):L=LEN(AS)
300 IF L>8 THEN L=9
310 PRINT@9-L,10;AS
320 AS=STR$(10*A):L=LEN(AS)
330 IF L>8 THEN L=9
340 PRINT@9-L,2;AS
350 IF L>7 THEN L=0:ELSE L=4
360 PRINT@L,22;"Sum total =";S
370 PRINT@25,22;"Again (Y/N) ";
380 AS=INCH$
390 IF AS="Y" OR AS="y" THEN 430
400 IF AS="N" OR AS="n" THEN 420
410 BEEP:GOTO370
420 BCOL4:TCOL15,4:RST:NEW
430 RUN 10
440 CLS32:PRINT@11,0;"BAR CHART".
450 PRINT@11,1;"11111111":REM GRAPH/1 BETWEEN QUOTES
460 RETURN
    
```



Plotter allows for a more detailed analysis.

```

10 DIMAS(46):BGS=MUL$( " ",30)
20 BCOL2:TCOL1,0:CLS40:GCOL1,0
30 FOR A=1 TO 46:AS(A)="":E=E+1
40 IF A=1 OR A=16 OR A=31 THEN CLS:PRINT@17,0;"PLOTTER"
50 IF A=1 OR A=16 OR A=31 THEN E=1:PRINT@17,1;"111111":REM
    GRAPH/1 BETWEEN QUOTES
60 IF A=46 THEN X=68:GOTO 100
70 TCOL1,3:PRINT@9,21;"Press key D to display":TCOL1,2
80 PRINT@9,E+3;"Input";A;"=";
90 X$=INCH$:X=ASC(X$):IF X=13 THEN 160
100 IF X=68 OR X=100 THEN C=A:A=50:GOTO 160
110 IF X=47 OR X<46 OR X>57 THEN BEEP:GOTO 90
120 AS(A)=A$(A)+X$
130 IF A<10 THEN PRINT@19,E+3;A$(A)::GOTO 150
140 PRINT@20,E+3;A$(A);
150 IF X<>13 THEN GOTO 90
160 NEXT
170 B=VAL(AS(1))
180 FOR A=2 TO C-1:D=VAL(AS(A)):REM MAX
190 IF B<D THEN B=D :REM MAX
200 NEXT :REM MAX
210 CLS:PRINT@17,0;"PLOTTER"
220 PRINT@17,1;"111111":REM GRAPH/1 BETWEEN QUOTES
230 DRAW 39,168 TO 223,168,2
240 DRAW 39,167 TO 39,63 TO 223,63
250 DRAW 39,168 TO 37,168
260 DRAW 39,141 TO 37,141
270 DRAW 39,115 TO 37,115
280 DRAW 39,89 TO 37,89
290 DRAW 39,63 TO 37,63
300 FOR A=39 TO 231 STEP 20
310 DRAW A,63 TO A,60:NEXT:TCOL1,11
320 FOR A=3 TO 15:PRINT@7,A;BGS
330 NEXT:IF B=0 THEN B=.000001
340 GCOL1,11
350 FOR A=1 TO C-1
360 Q=VAL(AS(A+1)):V=VAL(AS(A)):Y=104*V/B:R=104*Q/B:GCOL4,11
370 IF V=0 THEN Y=0
380 IF A=C-1 THEN 410
390 DRAW 39+4*A,63+Y TO 39+4*(A+1),63+R
400 NEXT
410 GCOL1,2:DRAW 39,63 TO 223,63:TCOL1,2
420 PRINT@6,17;"0 10 20 30 40"
430 IF B<.000001 THEN B=0
440 TCOL1,3:PRINT@6,21;"Max =";B
450 PRINT@27,21;"Again:"
460 IF AS="Y" OR AS="y" THEN RUN
470 IF AS="N" OR AS="n" THEN 510
480 PRINT@34,21;"Y":GOSUB 520
490 PRINT@34,21;"N":GOSUB 520
500 AS=KBDS$:GOTO 460
510 RST:BCOL4:TCOL15,4:CLS:NEW
520 FOR A=1 TO 300:NEXT:RETURN
    
```





\*\*\*\*\*

(XBAS)

MAP

A very good educational program from David Williams that will have you scurrying for your Atlases. (NOT XTALS)  
Instructions are contained in the program.

```

10 RST:MAG0:DIMAS(37),R(37)
20 DIMB(97),C(97),D(37),E(37)
30 CS=MULS(" ",11)
40 DS=MULS(" ",13)
50 SHAPE254,"40E0400000000000"
60 SHAPE253,"187E3C1800000000"
70 SHAPE252,"08183C787860E040"
80 SHAPE251,"3060200040400040"
90 SHAPE250,"081C1C3838703060"
100 SHAPE249,"0303000000000000"
110 TCOL1,11:CLS40:GCOL1,11:BCQL4
115 DRAW0,191TO256,191
116 DRAW0,0TO256,0
120 PRINT#14,1;"GREAT BRITAIN"
130 PRINT#14,2;"1111111111111111":REM GRAPH/1 BETWEEN QUOTES
140 PRINT#2,4;"You will be shown a map of Britain"
150 PRINT#2,5;"and a frame containing five towns"
160 PRINT#2,6;"selected at random. The position of"
170 PRINT#2,7;"one of these towns is indicated on"
180 PRINT#2,8;"the map by a flashing cursor."
190 PRINT#2,10;"You are invited to select which town"
200 PRINT#2,11;"this might be by using one of the"
205 PRINT#2,12;"number keys (1 to 5)."
```

\*\*\*\*\*

```

460 TCOL1,14:FORA=3TO7
470 PRINT#4,A;DS:NEXT
480 DRAW24,127TO103,127
490 DRAW24,126TO103,126:GCOL1,0
500 DRAW15,16TO233,16TO233,7TO15,7TO15,16
510 TCOL15,6
520 PRINT#3,22;"SCORE "":TCOL1,14:PRINT" Watch this space.
530 GCOL1,14:DRAW220,16TO220,7
540 DRAW56,16TO56,7
550 GOSUB820:TCOL15,0
560 SPRITE0,225,13,1,254
570 FORY=1TO900:NEXT
580 FORY=13TOE(R(RT))
590 SPRITE0,225,Y,1,254:FORW=1TO5:NEXT:NEXT
600 FORX=225TOD(R(RT))STEP-1
610 SPRITE0,X,E(R(RT)),1,254:FORW=1TO5:NEXT:NEXT
620 PRINT#2,11;"Which town - press a"
630 PRINT#2,12;"number key to select."T=T+1:POKE#013D,0
640 BS=KBD$;B=ASC(BS):V=VAL(BS)
645 IFB=27THENSPRITEOFF:TCOL15,4:CLS:NEW
650 IFB<49ORB=53THEN760:ELSEBEEP:PRINT#2,11;CS:PRINT#2,12;CS
660 IFV=RT+1THENZ=Z+1:PRINT#2,11;AS(R(RT));" is correct."GOTO875
670 PRINT#2,11;"Wrong - "AS(R(RT))
675 IFZ=18THENPRINT#2,16;"Medal awarded "":TCOL10,13:PRINT""
680 TCOL1,14:PRINT#10,22;Z;"right out of";T;"tries."":FORA=1TO1300:
NEXT:SPRITEOFF0:TCOL15,4
700 IFT=20THENT=0:PRINT#2,11;"End of this game. "":Z=0
710 IFT=0THENPRINT#2,12;"Press A for another."":N=INCH:IFN<>65ANDN
->97THEN710
720 PRINT#2,11;CS:PRINT#2,12;CS:PRINT#2,13;CS
730 IFT=0THENPRINT#2,16;CS:GOTO510
740 TCOL1,14:GOTO550
750 REM***** CURSOR BLINK *****
760 SPRITEOFF0
770 FORA=1TO200:NEXT
780 SPRITE0,D(R(RT)),E(R(RT)),1,254
790 FORA=1TO200:NEXT
800 GOTO640
810 REM**** RANDOM SELECTIONS *****
820 FORA=0TO4
830 R(A)=RND(37)
840 FORB=0TOA:IFB=ATHEN860
850 IFR(B)=R(A)THEN830
860 NEXTB,A
870 FORA=0TO4
880 PRINT#4,3+A;DS
890 PRINT#4,3+A;A+1;AS(R(A))
900 NEXT:RT=RND(5):RETURN
910 REM*****
920 DATA150,156,169,168,158,160,158,166,178,181,172,174,164,174,178,
184,188,198,204,201,205,207,214,218,220,219,213
930 DATA202,217,218,206,201,194,192,182,178,175,172,166,162,154,144
940 DATA144,140,147,153,158,159,170,174,179,169,158,156,149,146,157,
159,159,150,158,170,171,169,169,170,172,168
950 DATA164,156,170,165,160,157,155,152,150,148,153,152,147,144,142,
143,143,144,139,140,142,141,143,142,146,146,148,148,151
```

\*\*\*\*\*

- 960 DATA181,179,182,178,170,169,163,166,166,162,144,140,136,136,134,128,113,104,88,83,80,84,83,80,76,70,59,56,55,53
- 970 DATA45,47,45,46,45,42,41,43,42,35,38,31
- 980 DATA33,33,39,49,50,52,54,57,63,58,59,62,61,63,71,73,83,79,89,92,98,99,102,103,106,104,111,117,119,119
- 990 DATA116,117,114,117,114,118,127,133,137,125,124,133,135,145,148,150,151,159,165,167,168,171,177,180,181
- 1000 DATAAberdovey,158,78,Aberdeen,177,157,Birmingham,184,77
- 1010 DATABrighton,199,48,Bristol,175,58,Skegness,203,90
- 1020 DATAWorcester,176,70,Nottingham,190,85,Doncaster,191,96
- 1030 DATASwindon,183,59,Luton,196,65,Norwich,215,80
- 1040 DATACambridge,202,71,Exeter,164,45,Plymouth,156,39
- 1050 DATATorquay,163,39,Carlisle,167,119,Glasgow,157,135
- 1060 DATATHurso,164,183,Wick,167,179,Stirling,162,141
- 1070 DATAGloucester,177,63,Portsmouth,192,47,Grimsby,200,99
- 1080 DATAYork,189,104,Liverpool,169,96,Minehead,165,55
- 1090 DATABoston,199,86,Shrewsbury,173,81,Fishguard,148,67
- 1100 DATAPenzance,140,35,Cardiff,169,61,Ipswich,213,66
- 1110 DATAIlfracombe,151,50,Inverness,158,165,London,204,59
- 1120 DATADolgellau,162,80

ALTERWP

Convert your WDPRO text files. A.B.Wilmot

Some while ago I purchased the latest version of WDPRO, essentially because I had upgraded to the 80 column card. Previously I had used WDPRO 2.40 extensively and had accumulated about 70 text files.

When I got to grips with version 2.60/2.61 I found to my horror that KUMA had changed the formatting character and my 70 files created under version 2.40 just did not respond as they should.

I then looked a bit closer at the manual!! I duly discovered that a conversion utility had been provided on the new disc. What a relief I thought. So I proceeded to try it - CVRT NAME.UFT - and it worked of course, but then came the rub - only on one file at a time and only in one direction from 2.40 to 2.60/2.61. So what you may say. Very true if you have only a few text files to convert, but remember I had more than 70, and I also wanted them to work with the 40 column screen as well sometimes.

So I decided to write my own multiple conversion program and this is the result. The program will convert all or one file from 2.40 to 2.60/2.61 or vice-versa on any disc, telling you if a file was already converted in the process. Locked files are also transformed.

To execute the program, type (from DOS) ALTERWP [d:] \*.UFT to convert all files or ALTERWP [d:] fname.UFT for just the one file, where [d:] means an optional drive number. If you have typed everything correctly then you will be presented with a menu to choose which way to convert, and off it goes.

It is essential, though that your text files are terminated with the formatting character followed by Z, as the manual says.

\*\*\*\*\*

The source code was written and assembled using HISOFT's excellent DEVPC80 package, although others should be suitable. For anyone who prefers or who is brave enough, I have included below a hex dump of the executable code which can be entered directly from MOS starting at M0100 and then saved as SAVE 5 ALTERWP.COM, when back in DOS.

```

CD 4D 03 CD 60 02 CD 2D 01 CD 55 01 21 0E 05 CD 97 01 C8 CD D7
01 CD E5 01 CD F7 01 CD 17 02 CD C8 01 CD D7 01 CD 83 02 CD FE
02 18 E2 11 8E 04 0E 09 CD 25 03 1E FF 0E 06 CD 25 03 FE 31 C8
FE 32 28 02 18 F0 21 09 05 46 21 0A 05 7E 32 09 05 78 32 0A 05
C9 CD 57 02 CD 62 01 CD 76 01 EB 36 1A C9 CD 4C 02 0E 11 CD 25
03 FE FF CA F2 02 11 0E 05 CD 86 01 C9 CD 4C 02 0E 12 CD 25 03
FE FF C8 CD 86 01 18 F0 21 80 00 06 05 87 10 FD 85 6F 23 01 08
00 ED B0 C9 7E FE 1A C8 11 5C 00 3A 0C 05 12 13 06 0B CD B4 01
7E 12 23 13 10 F7 E5 CD 4C 02 E1 C9 3E 03 B8 C0 CB 7E 3E FF 32
00 05 C8 3E 00 32 0D 05 CB BE C9 E5 3A 0D 05 FE FF 28 05 21 65
00 CB FE E1 C9 E5 CD 4C 02 E1 11 5C 00 0E 1E CD 25 03 C9 E5 11
5C 00 0E 0F CD 25 03 3C 11 7C 04 CC AD 02 E1 C9 E5 21 00 00 22
7D 00 11 5C 00 0E 21 CD 25 03 B7 20 0C CD 1D 02 0E 22 CD 25 03
23 B7 28 E6 E1 C9 0E 10 CD 25 03 C9 E5 D5 3E 00 32 0B 05 21 80
00 06 80 3A 09 05 BE 28 09 3A 0A 05 BE 28 0C 23 18 0F 3A 0A 05
77 3E FF 32 0B 05 23 3E 5A BE 28 02 10 E0 D1 E1 C9 21 68 00 06
18 36 00 23 10 FB C9 21 0C 05 11 5C 00 1A 77 C9 21 65 00 3E 55
BE 20 0D 3E 46 23 BE 20 07 3E 54 23 BE 20 01 C9 11 2F 03 0E 09
CD 25 03 0E 00 CD 25 03 C9 E5 21 0A 05 7E FE AF 28 0F 3A 0B 05
FE 00 11 6B 04 28 12 11 3F 04 18 0D 3A 0B 05 FE 00 11 5A 04 28
03 11 24 04 CD AD 02 E1 C9 D5 21 5C 00 7E FE 00 CC EA 02 C6 2F
5F 0E 02 CD 25 03 1E 3A 0E 02 CD 25 03 06 08 23 5E 0E 02 CD 25
03 10 F7 1E 2E 0E 02 CD 25 03 06 03 23 5E 0E 02 CD 25 03 10 F7
D1 0E 09 CD 25 03 C9 0E 19 CD 25 03 C6 01 C9 11 7C 04 CD AD 02
0E 00 CD 25 03 C9 0E 0B CD 25 03 0F D0 0E 01 CD 25 03 FE 1B F5
11 05 05 0E 09 CD 25 03 F1 C0 0E 30 CD 25 03 C9 1E 0C 0E 02 CD
25 03 C9 D5 C5 E5 CD 05 00 E1 C1 D1 C9 49 6E 63 6F 72 72 65 63
74 20 66 6F 72 6D 61 74 0D 0A 55 73 65 20 41 4C 54 45 52 57 50
20 5B 64 3A 5D 20 61 66 6E 2F 55 46 54 0D 0A 0D 0A 54 68 69 73
20 75 74 69 6C 69 74 79 20 77 69 6C 6C 20 63 6F 6E 76 65 72 74
20 57 44 50 52 4F 20 74 65 78 74 0D 0A 66 69 6C 65 73 20 66 72
6F 6D 20 76 65 72 73 69 6F 6E 20 32 2E 34 30 20 74 6F 20 32 2E
36 30 20 6F 72 0D 0A 76 69 63 65 2D 76 65 72 73 61 2E 20 45 69
74 68 65 72 20 6F 6E 65 20 66 69 6C 65 20 6F 72 20 61 6C 6C 0D
0A 66 69 6C 65 73 20 6F 6E 20 61 20 73 65 6C 65 63 74 65 64 20
64 72 69 76 65 20 77 69 6C 6C 20 62 65 0D 0A 63 6F 6E 76 65 72
74 65 64 2C 20 69 6E 63 6C 75 64 69 6E 67 20 6C 6F 63 68 65 64
20 66 69 6C 65 73 2E 0D 0A 41 2E 42 2E 57 69 6C 6D 6F 74 20 31
39 38 36 0D 0A 24 20 20 32 2E 34 30 20 74 6F 20 32 2E 36 30 20
43 6F 6E 76 65 72 74 65 64 0D 0A 24 20 20 32 2E 36 30 20 74 6F
20 32 2E 34 30 20 43 6F 6E 76 65 72 74 65 64 0D 0A 24 20 20 41
6C 72 65 61 64 79 20 32 2E 36 30 0D 0A 24 20 46 69 6C 65 20 6E 6F 74 20 66
64 79 20 32 2E 34 30 0D 0A 24 20 46 69 6C 65 20 6E 6F 74 20 66
6F 75 6E 64 0D 0A 24 57 44 50 52 4F 20 4D 55 4C 54 49 50 4C 45
20 46 49 4C 45 20 43 4F 4E 56 45 52 54 20 55 54 49 4C 49 54 59
20 76 31 2E 31 0D 0A 0D 0A 43 6F 6E 76 65 72 74 20 32 2E
34 30 20 74 6F 20 32 2E 36 30 2E 2E 2E 2E 2E 2E 2E 2E 2E 28 31
29 0D 0A 0D 0A 43 6F 6E 76 65 72 74 20 32 2E 36 30 20 74 6F 20
32 2E 34 30 2E 2E 2E 2E 2E 2E 2E 2E 2E 28 32 29 0D 0A 0D 0A 24
OD 20 OD 24 1F AF 00 00 FF
    
```

## TRICKS OF THE TRADE

(BY VIN DAVIES COPYRIGHT (C) 1989)

On looking at many of the programs presented in the magazine it occurred to me that it could be helpful if some of the methods used by professional programmers were illustrated. These are the techniques for writing programs that are easy to maintain and writing code that is simple but fast and sometimes minimises the amount of space used by the program. Perhaps the best approach is to use examples taken from programs in the magazine. I trust that the authors of these programs will appreciate that it is constructive help that is being offered, NOT criticism. It must also be mentioned that any changes to an authors code does not affect the original copyright. So with the preliminaries out of the way we can get down to some detail.

In this article I will concentrate on those situations where it appears as if a multiple of 'IF' statements are required.

Here is part of a program from the January 1989 mag. Copyright of Mark Ball:

```

50 IF A<1 THEN O=0
60 IF A>1 AND A<25 THEN O=1
70 IF A>25 AND A<50 THEN O=2
80 IF A>50 AND A<75 THEN O=4
90 IF A>75 AND A<100 THEN O=8
100 IF A>100 AND A<125 THEN O=16
110 IF A>125 AND A<150 THEN O=32
120 IF A>150 AND A<175 THEN O=64
130 IF A>175 AND A<200 THEN O=128
140 IF A>200 AND A<250 THEN O=240
150 IF A>225 AND A<250 THEN O=15
160 IF A>250 THEN O=255
  
```

(note that there is no test for equality - what happens if A = 100? In what follows I have assumed that the test is IF A >= 100 etc.)

This sort of code can almost always be improved. There are two methods which are often useful.

1. Analyse the the relationship between the condition and the assignment. In the above example looking at lines 60 to 130 it can be seen that the value of O can be calculated instead of using IF statements.

E.g. (1) take the integer result of dividing a by 25

(2) Raise 2 to the power of the result from (1) above and this gives the value of O. For example if A=110 then A/110 is 4 and 2 to the power of 4 is 16. This looks after lines 60 to 130 (inclusive) but lines 50,140,150 and 160 do not fall into this scheme. But we can still use IF for these so the code would be...

```

50 IF A%<1 THEN O% = 0 : ELSE O% = INT(2^INT(A%/25))
60 IF A%>=200 AND A<225 THEN O%=240
70 ELSE IF A%>=225 AND A%<250 THEN O%=15 ELSE IF A%>250
   THEN O%=255
  
```

As you can see it is much shorter and faster. Don't forget to make use of the else clause, it can prevent tests being made which it is already known are not relevant. This could be improved slightly but for the above example there is an even better method - called a TABLE LOOK-UP. The advantage of a table look up method is that there is no need for a relationship between the condition and the assignment. It is necessary however that the condition value can be expressed as a sequence of integers. We already know that this is alright in the above example. We simply divide A by 25. Now we shall use the result of that division to access an array element. All that is necessary is to initialise the array to the required values. In the code shown below the initialisation of the array is done in lines 10-70 (incl). The bank of IF THEN statements have been replaced by one line, line 90. Lines 80 and 100 have been included to test the table look-up and show that the value of O is, indeed, set up correctly.

```

10 DIM ARR 11
20 X%=1
30 FOR I%= 0 TO 8
40 ARR(I%) = X%
50 X%=X%+X%
60 NEXT
70 ARR(8) =240:ARR(9) = 15 :ARR(10) = 255
75 REM the array has been initialised
80 FOR A% = -1 TO 275 STEP 25
90 IF A%<1 THEN O% = 0: ELSE O% = ARR(A%/25)
100 PRINT A%,O%
110 NEXT
  
```

I hope that I have demonstrated that programs can be improved by looking carefully at the data (including the values involved in conditional statements). Careful organisation of data can often lead to simpler programs and simpler file handling. Finally if you are using integers do use the % notation because it saves space and execution is faster. simpler file handling. Finally if you are using integers do use the % notation because it saves space and execution is faster.

# Joystick Conversions

I have recently bought an Einstein TC01 secondhand, and while looking for a joystick I was dismayed by the lack of choice in quality joysticks for Albert. It could be argued that an adapter could be used, however this would add around £10 to the cost of the joystick.

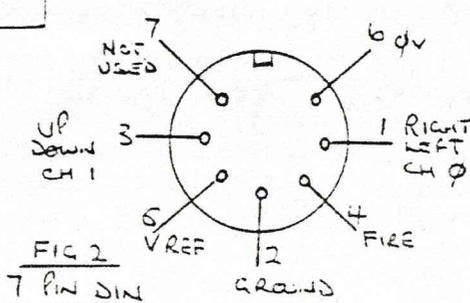
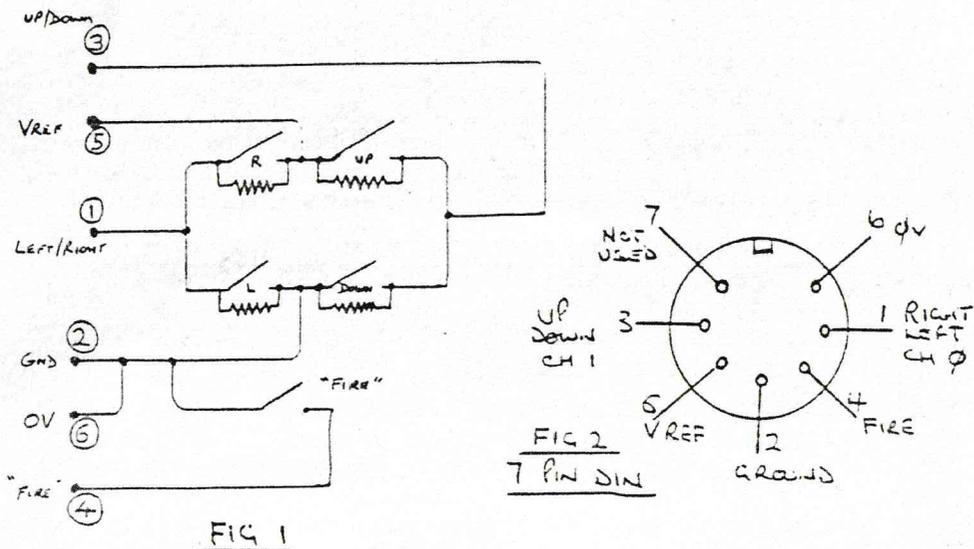
Most good quality joysticks use micro-switches and provided this is the case they can be converted to work with the Einstein for a little over a pound and fifteen minutes work. Joysticks such as the Powerplay Crusier and Competition Pro are suitable for this conversion.

All that you will need are 4 \* 100 ohm resistors, any tolerance and a low wattage, (so as fitting them inside the case is not a problem), and a 7 pin din plug.

Remove the 9 pin plug from the joystick and strip back the cables ready to solder to the din plug. Open up the joystick and rewire as per the diagram.

As a novice to Albert, but having experience with other micros, I would be interested in contacting other users in my area, (Glasgow), and would be pleased to answer questions on joystick conversions.

Stuart Wilson, 124 Collessie Drive, Craigend, Glasgow, G33 5QB.  
Tel: 041 7740607.



## BURGLAR ALARM.

A gem of a project from Dave Arts, just what you need to guard your gems.

This is a project for Einstein Users who like to put Albert to practical use and at the same time have the peace of mind in knowing that one of the most powerful 8-bit micros is guarding your home while you are away.

The idea here, is to use 4 chains of switches each chain comprising either normally open switches in parallel or normally closed switches in series which Albert can monitor. (Fig 1).

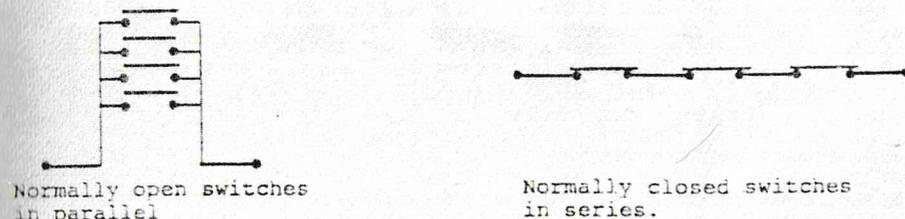


FIG 1.

These switches can be in the form of microswitches, pressure mats, tilt switches or more simply a fine wire broken when a door opens. This information is initially fed into Albert by a Dual in line switch package (D.I.L. Switch) this is then removed and the chain connector PL1 is plugged into SK1 on the unit, then, on engaging a secret master switch S5; Albert enters a routine which monitors the sensor switches in the chains with the information previously programmed in by the D.I.L. switch; any discrepancy results in the Alarm sounding unless the master switch is reset a short time after you re-enter the house.

### COMPONENT LIST

- R1 - R5, R13, R14 10K watt resistors
- R6 - R12 390 watt resistors
- IC1 7486 QUAD 2 INPUT EXCLUSIVE OR GATE
- IC2 7432 QUAD 2 INPUT OF GATE
- IC3 7447 7 SEGMENT DISPLAY DECODER
- IC4 7805 5 VOLT VOLTAGE REGULATOR
- DISPLAY Common Anode Type FR39N (Maplin) or equivalent
- If different display used check pinouts.
- Q1 TIP31C
- D1, D2, D3 IN4001
- C1, C4 470UF 25v.w. electrolytic capacitor.
- C2, C3 0.1UF Ceramic disc capacitor.
- D.L.L. Switch OCTAL Single pole single throw. (4 poles used).
- Relay RL1 Any small 12 volt relay contacts to switch Alarm in use.
- 14 pin D.I.L. IC SOCKET 2 OFF (For 7432+7486)
- 16 pin D.I.L. IC SOCKET 2 OFF (For D.I.L. SWITCH & 7447)
- TRANSFORMER 12-0-12v secondary rated at amp primary 240 volt mains.
- I.D.C. Socket 16 pin (For Alberts user port).
- S5 on/off Toggle switch or Keyswitch if preferred.
- Miscellaneous Microswitches; pressure mats etc.
- SK1 9 pin 'D TYPE' CONNECTOR SOCKET CONTACTS.
- PL1 9 PIN 'D TYPE' CONNECTOR PIN CONTACTS.

The user port is configured so that the lower 4 bits D0 to D3 are connected to the switch chains (inputs) D4,D5 & D6 are output lines and D7 is connected to the secret master switch S5 which is an input, this gives us:-

D7 D6 D5 D4 D3 D2 D1 D0  
 1 . 0 0 0 1 1 1 1

&8F or decimal 143 as the port control word.

FIG 2 shows the circuit of the Burglar Alarm. (p.53)  
 All this can be built on a piece of Veroboard and mounted in a suitable plastic box with leads coming off the board to a suitable socket SK1 into this connects P11 which is the remote sensor chains connector.

On switching on Albert, before running the program, all Data lines D0-D7 are effectively high (at the + 5v level). The output lines D4,D5 & D6 are fed to a gating system which holds off the relay RL1 by sending the base of Q1 low.

The display, which is driven by decoder chip type 7447, whose inputs are also connected to D4,D5 & D6 will display figure 7 (binary 111); when the programme is run, when these three output lines go low the display will now show a 0 and again the gating arrangement will hold off the relay. However, when the Alarm is triggered, the chain number which has been tripped will be output on these three lines either 1,2,3 or 4. Under these conditions the display will register the chain number and the alarm must sound i.e. Transistor Q1 must obtain base drive (the gate output must be high).

The truth table for our required gating system is shown in FIG 3.

D6	D5	D4	DISPLAY	OUTPUT	
1	1	1	7	0	No Alarm
1	1	0	X	X	Irrelavent
1	0	1	X	X	Irrelavent
1	0	0	4	1	Alarm Condition
0	1	1	3	1	Alarm Condition
0	1	0	2	1	Alarm Condition
0	0	1	1	1	Alarm Condition
0	0	0	0	0	No Alarm

FIG 3.

The arrangement of gates which meet these conditions are shown in FIG 2 and are two 2-input exclusive or gates (XOR) and a single 2-input or gate.

The truth table for an exclusive or (XOR) is shown in FIG 4, note that the output goes high only if the inputs are different. Note this point well, as the software equivalent functions is used in the program later.

INPUT A	INPUT B	OUTPUT Y
0	0	0
1	0	1
0	1	1
1	1	0

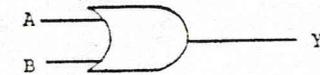


XOR

FIG 4.

The truth table for a standard or gate is shown in FIG 5. Maybe the more ambitious of you may like to work out how this combination of gates produce our desired truth table.

INPUT A	INPUT B	OUTPUT Y
0	0	0
1	0	1
0	1	1
1	1	1



OR

FIG 5.

The input lines D0 - D3 & D7 are pulled up to the +5v supply line by 10K pull-up resistors R1-R5, closing the D.I.L. switches will ground the selected data line D0-D3 to the 0V level. As stated before the D.I.L. switch feeds the chain conditions into Albert, for instance a typical input could be:-

D3 (chain 1) could be normally closed switches in series monitoring downstairs windows (0)

D2 (chain 2) could be normally closed switches in series monitoring upstairs windows (0)

D1 (chain 3) could be normally open pressure mats placed in front of the front & rear door< these are of course in parallel. (1)

D0 (chain 4) could be normally closed switches in series monitoring Garage windows & door. (0)  
 Hence the 4 lower bits would be

D3 D2 D1 D0  
 0 0 1 0

The Secret Master Switch.

This simple on/off switch controls the input to D7 and is interpreted by the software at three points in the programme in different ways and so bestows on this switch 3 distinct functions.

- 1) It starts Albert executing the programmes main loop.
- 2) It inhibits the Alarm should you come back and open the front door and step in.  
 (Therefore its position should be near the front door, but hidden).
- 3) It turns off the Alarm should it be sounding.

Operation.

When the programme is run a zero is output to the 7-segment display and the required chain conditions are read into the accumulator and transferred to the 'E' register (in the above example 0010) the programme then enters a loop which constantly monitors the input D7 on each pass and tests for Master Switch set low (0V). This gives the user time to check all windows and doors, disconnect the D.I.L. Switch and connect the chain connector P11 into SK1 and go to the front door.



\*\*\*\*\*

The Master Switch is now set low and the computer sensing this enters a short delay loop to allow the user to exit the house. This delay is typically 5 seconds. Albert now enters the main loop of the programme.

The Main Loop.

The first thing Albert does is read the user port into the accumulator and test the lower 4 bits D0-D3 with the 'E' register it does this by logically XORing the accumulator with the 'E' register. If they are identical (no trip occurred) the accumulator will be modified by the XOR function so that it will contain zero's in bits D0-D3. The 'E' register is unaffected.

D3	D2	D1	D0	
0	0	0	0	Accumulator ('A' register)
0	0	1	0	'E' register
0	0	1	0	XOR 'E' (result in accumulator)

After XORing, the accumulator is modified showing a '1' at position D1, on the first rotation into the carry flag with 'B' count at 4 there is no '1' and so 'B' is decremented and the accumulator is rotated again, this time with 'B' count at 3. The carry flag is set to '1', when this happens the 'B' register is transferred into the accumulator, but remember, as we are outputting data on D4, D5 and D6 we must rotate the accumulator 4 times, this time to the left ready to output.

However at this moment in time Albert cannot tell whether you have returned home from a hard days work. It determines this by monitoring D7 (master switch reset) for a '1' after a short delay. If a '1' is present you have reset the switch and the computer returns to the BASIC program. If a '0' is present the alarm chain has been tripped and the master switch not reset. The computer interprets this as an intrusion and outputs the count to the display, the relay energises and the alarm sounds. The computer then enters another loop, this time again testing bit 7 and resetting the alarm if a '1' is detected (i.e. the master switch turns off the alarm, once sounded).

The software:

The assembler listing with explanation and machine code follows.  
The code starts at &8000 and is saved by;

```
SAVE "0:ALARM.OBJ",&8000,&8055
```

The small BASIC program reads...

```
10 CLEAR &8000
20 LOAD "ALARM.OBJ"
30 CALL &8000
```

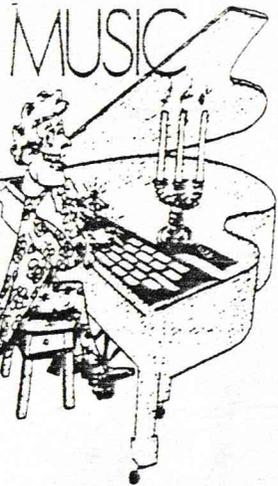
\*\*\*\*\*

```

3ECF          LD A,207
D333          OUT (&33),A      INITIALIZE + CONFIGURE
3E00          LD A,0          THE PORT FOR &8F
D333          OUT (&33),A      10001111
3ECF          LD A,207
D333          OUT (&33),A
3E8F          LD A,143
D333          OUT (&33),A
DB32          START:IN A,&32    READ USER PORT (DIL SWITCH)
5F           LD E,A          TRANSFER TO 'E' REG.
DB32          LOOP:INA,&32     READ USER PORT
CB7F          BIT 7,A         HAS MASTER SWITCH 'S5' BEEN SET?
20FA          JR NZ,LOOP     LOOP UNTIL SET TO 0
CD4280        CALL DELAY     MASTER SWITCH HAS BEEN SET
DB32          MAINLOOP:IN A,&32 READ USER PORT
AB           XOR E          COMPARE WITH 'E' REG.
0604          LD B,04        LOAD 'B' WITH COUNT (CHAIN No.)
1F           BITROT:RRA      ROTATE 'A' REG RIGHT 1 BIT
3804          JR C,4        TEST CARRY FLAG. JUMP IF SET TO 1
10FB          DJNZ BITROT    TEST NEXT CHAIN No.
18F4          JR,MAINLOOP   ALL CHAINS O.K. LOOP AGAIN.
CD4280        CALL DELAY
DB32          IN A,&32       READ USER PORT
CB7F          BIT 7,A       TEST FOR MASTER SWITCH RESET
C0           RET NZ        RETURN TO BASIC AS SWITCH RESET
78           LD A,B        ALARM CHAIN BROKEN SWITCH NOT RESET
17           RLA          TRANSFER 'B' REG TO ACCUMULATOR
17           RLA          AND ROTATE 4 TIMES
17           RLA
17           RLA
D332          TEST:OUT (&32),A OUTPUT CHAIN No. ALARM SOUNDS
DB32          IN A,&32       READ USER PORT
CB7F          BIT 7,A       TEST FOR RESET
28FA          JR Z,TEST     LOOP UNTIL RESET
3E00          LD A,0        LOAD ACCUMULATOR WITH RESET DATA
D332          OUT (&32),A   OUTPUT RESET DATA (ALARM STOPS)
C9           RET          RETURN TO BASIC
C5           DELAY:PUSH BC
060A          LD B,10
C5           PUSH BC
21FFFF       LD HL,65535
010100       LD BC,1
ED42          SBC HL,BC
20FC          JR NZ,-4
C1           POP BC
10F2         DJNZ -14
C1           POP BC
C9           RET          RETURN FROM DELAY SUB-ROUTINE

```

NOTE: if delay is considered insufficient then location &8044 should be poked with values greater than 10 (0A) although values greater than 30 (1E) should be avoided.



JOSEF KARTHAUSER THROWS A FEW RESISTORS TOGETHER TO PRODUCE POLYPHONIC HI-FI FROM ALBERT

At the last Einstein show many of you were probably quite amazed to hear an Einstein blasting out pure polyphonic sounds from its printer port into an external amplifier and I had many comments from people who wanted to have the same facilities from their computers, so due to popular demand this is how it is done. Many people know that you can digitise sounds by using an Analogue to Digital Converter, or ADC, to sample sounds into memory and then play it back using a Digital to Analogue Converter, or DAC, into an amplifier. This method of playing music can be quite hard because of the amount of memory required, but what a lot of people don't know is that by using only a DAC on its own some quite good effects can be achieved.

Using software like R.T. Russell's music system (just released on the UKEUG Public Domain library) or the Musicraft System (due for PD release in about a months time) which can create the correct digital signals to control the converter up to four channels of sound can easily be utilised. This can be achieved by outputting digital values which refer to waveform patterns, for example if the two values 1 and 255 were output one after the other continually then a square wave could be produced, by altering the speed at which this is done the frequency of the note can be changed. By merging four of these together and making one waveform from them all four notes can be played at once, which is roughly what the above mentioned music systems do. But in order to enable Albert to play this music you must build a DAC.

There are all sorts of fancy chips and circuits designed to operate in this fashion, but for the beginner, or experimenter, the simplest method works quite well.

All a basic DAC consists of is a resistor network (the one I used at the show consisted of only nineteen resistors) connected to an 8 bit I/O port (see figure one). Fortunately Albert has two such ports built in and you can take your pick of either the User Port or the Printer Port. The only difference between the two is the cost of the connectors. The Printer connector is more expensive than the User port connector but is easier to get; in fact I couldn't get hold of a User port plug for the show so I used the printer port instead. Apart from this there are no software incompatibilities because both ports are controlled by the Z80-PIO and the only differences are the I/O port numbers (for details of how to connect the two plugs see figure two). This music system consists of two PD disks, both of which contain various music (.MUS) files and documents on how to program it. MUSIC.COM is already configured for a DAC on Albert's printer port but it can be changed by using the INSTALL.COM program and answering the relevant questions (see figure

three for the port data for the User and Printer Ports.) When the DAC is plugged between Albert and a suitable amplifier, some HI-FIs will do, he will finally be able to play some decent music.

FIGURE 1. A

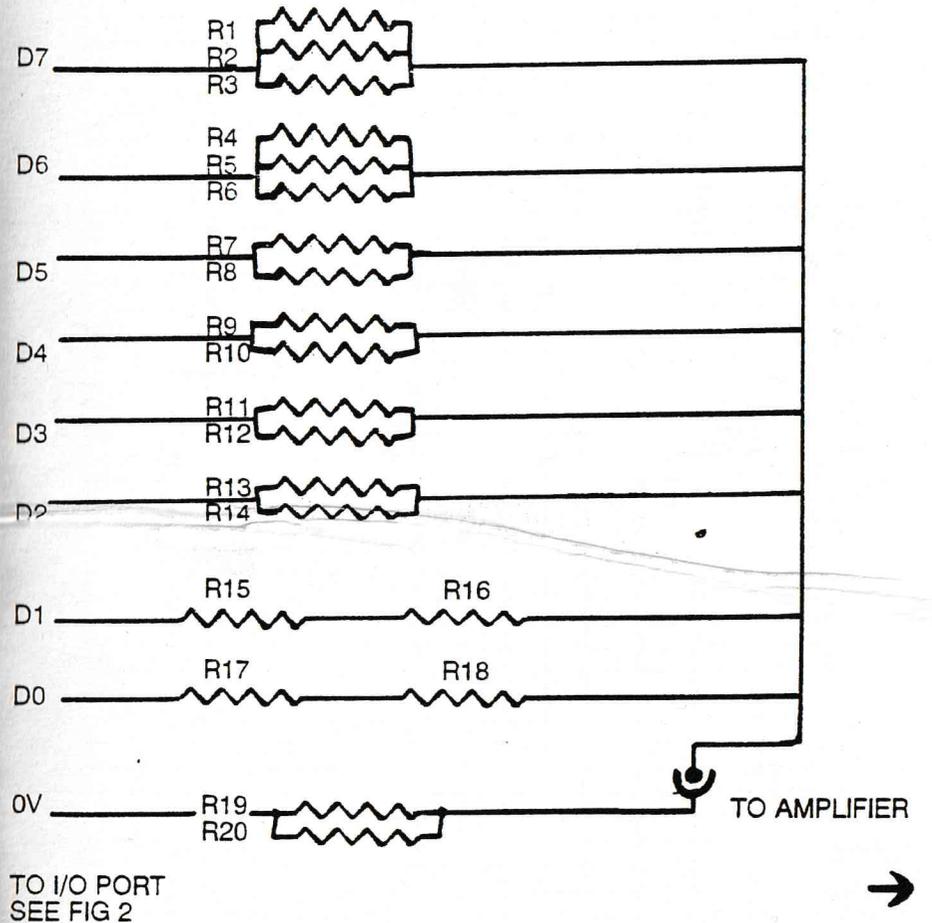


Figure 1A connection to 0V should look like this: ADDRESS OF CIRCUIT

→ FIG 1.B STRIPBOARD LAYOUT

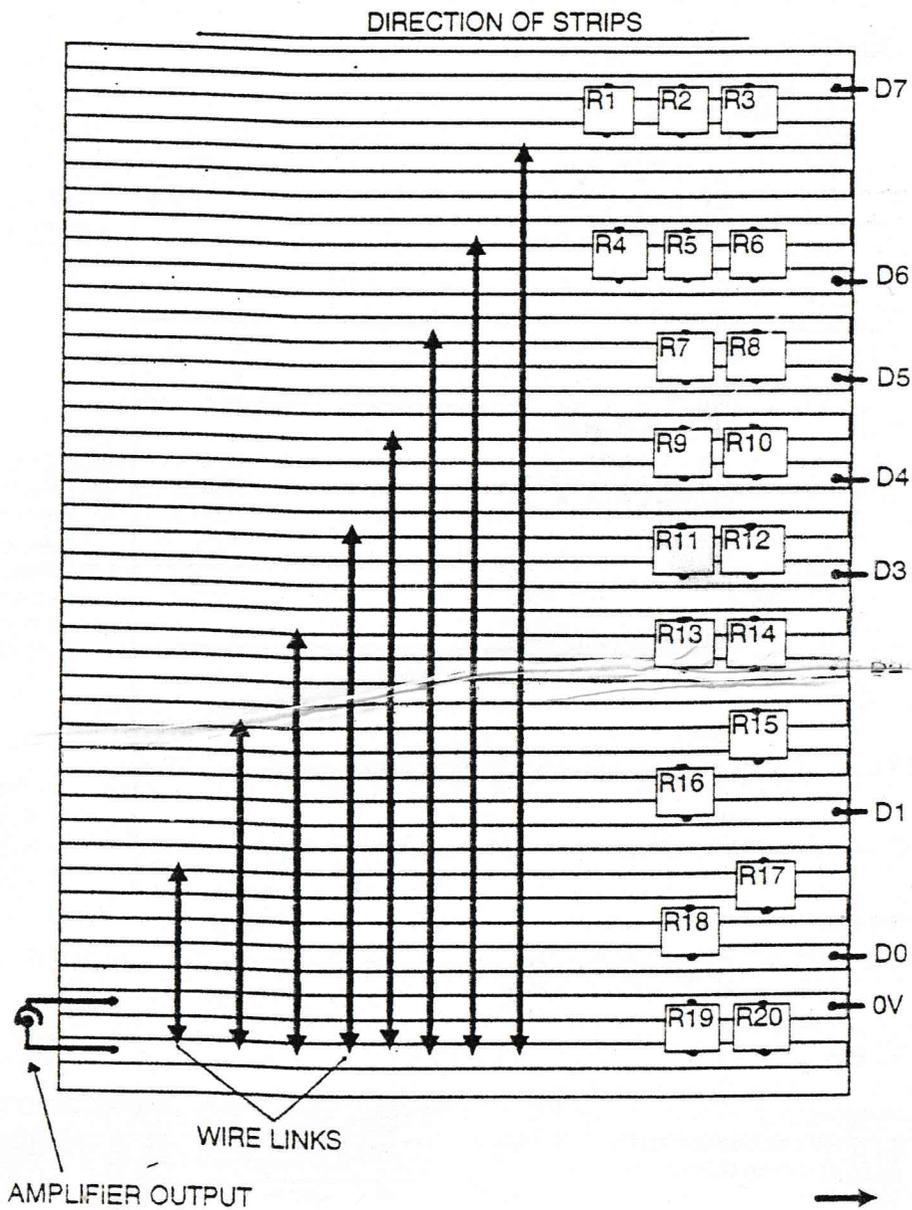
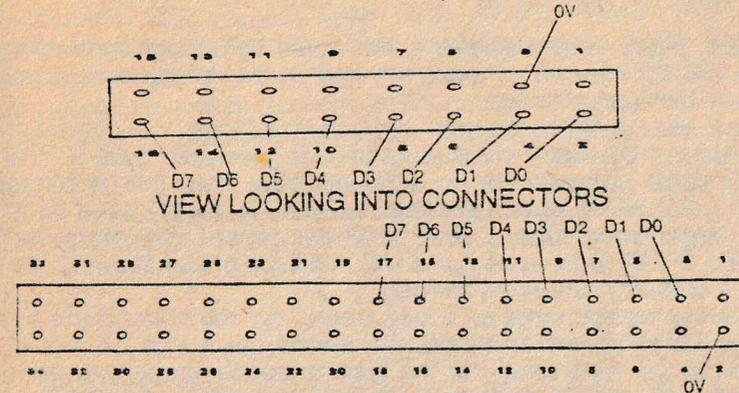


FIGURE 2 PORT CONNECTIONS.  
 USER PORT



The program for running the music via the port is set up for the USER port, if you want to use the printer port then run the install program on the disc.  
 Here is a list of components:

Stripboard 21 strips wide \* 13 holes high

R1.....	1K	RESISTOR
R2.....	13K	RESISTOR
R3.....	6.8K	RESISTOR
R4.....	2.2K	RESISTOR
R5.....	22K	RESISTOR
R6.....	27K	RESISTOR
R7.....	27K	RESISTOR
R8.....	4.7K	RESISTOR
R9.....	8.2K	RESISTOR
R10.....	330K	RESISTOR
R11.....	39K	RESISTOR
R12.....	27K	RESISTOR
R13.....	39K	RESISTOR
R14.....	150K	RESISTOR
R15.....	39K	RESISTOR
R16.....	27K	RESISTOR
R17.....	120K	RESISTOR
R18.....	27K	RESISTOR
R19.....	100OHM	RESISTOR
R20.....	47OHM	RESISTOR

- 1 \* 16 WAY IDC CONNECTOR IF USING USER PORT OR
- 1 \* 34 WAY IDC CONNECTOR IF USING PRINTER PORT
- 1 \* PHONO SOCKET (OR SIMILAR, FOR OUTPUT TO AMP)
- 1 \* SMALL BOX
- 6 INCHES (APPROX.) 34 WAY RIBBON CABLE

➔ For those of you who have never attempted any projects before this is a good example to start with. Very few tools are needed and the results are very impressive.

Side cutters, pliers, and a soldering iron would be enough to build the DAC, and a drill will be needed to make the hole in the plastic box to mount the phono or similar output socket.

If you don't own a soldering iron we would suggest you obtain one that produces around 50 watts, the amount of heat generated by a 50 watt iron is enough for large connections on a board, more than this is probably not needed for component work. Less than 30 watts is usually used for fine work where the larger irons heat is likely to lift the copper track from the printed circuit board, (PCB) Most irons come with a choice of tips allowing the user to change to a different size when needed.

When soldering components on a strip board, 'tin' the hole on the board first by applying some solder and heat, the purists get a nice square of solder melted around the hole ready for the lead of the component to be inserted. Having tinned the copper track the first task is to solder in the wire links, make sure you get these right. Next, the resistors, insert the lead of the resistor, do not cut the leads first as it is easy to get them too short! instead push the lead through and solder in place adding a small amount of solder, remove the iron as soon as you see the solder melt to form a nice shiney, or 'wet' joint. Now cut the excess lead back to the joint with the side cutters. Only 39 more to do! Having mounted the resistors the next job is to make up the connecting lead, this uses what is known as an IDC socket, and some ribbon cable. These type of connectors are very easy to assemble. The use of a small vice greatly aids the operation but a large pair of pliers will work with care. The IDC socket is squeezed together onto the ribbon cable and the opposite ends are stripped back for soldering to the strip board. Ensure the DATA lines are connected to D0 - D7 and the 0v to 0v, tin the wires and the board before soldering together. All that is left now is to mount it in the box complete with connections to the output plug, 0v goes to the outer connection.

To use the completed DAC plug the IDC connector into the relevant port on the rear of the Einstein and connect the output to an external amplifier, such as your HI-FI. Insert the MUSIC disc and from DOS type, yes you've guessed, MUSIC<E>.

After the music compiler has loaded select a tune with PLAY TUNE<E> The MUSIC disc is available from the P.D. library as disc P.D.193, it includes, as well as the compiler, a large document file and over 40 tunes. There is also a second disc of further tunes, under P.D.192, this disc will not work on it's own and all the tunes will only work with the DAC.

Installation data:-

**USER PORT**

Control Port = &33 or 51 in decimal  
Control Data = &0F,&00 in Hex  
or 15,0 in decimal  
Data Port = &32 or 50 in decimal

**PRINTER PORT**

Control Port = &31 or 49 decimal  
Control Data = &0F,&00 in Hex or  
15,0 in decimal  
Data Port = &30 or 48 in decimal

## 4.5KHZ LOW PASS FILTER

COLIN COKER OF THE S.W.E.U.G. CLEANS UP THE DAC O/P

This filter was designed to 'clean up' the sound produced by the digital to analogue converter (DAC) when used with MUSIC (PD192-193) or the MUSICRAFT system (PD195-197). This high frequency noise is known as quantisation noise and is inherent in this type of conversion. Electrically, the filter is a 4 pole active filter with Butterworth characteristics, ie flat response to the frequency breakpoint, and -24dB per octave attenuation above that frequency.

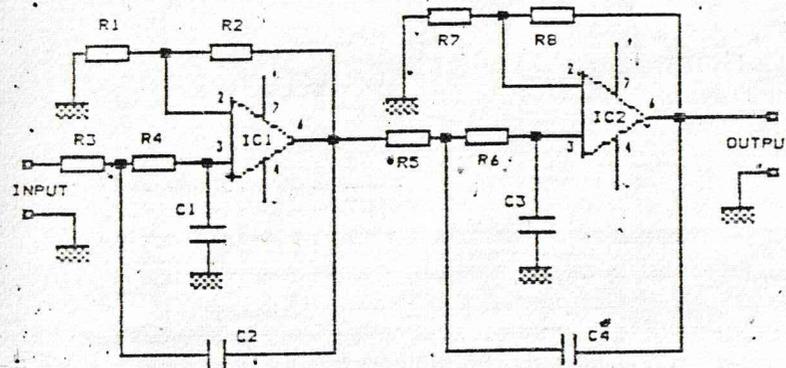


Fig 1

**CONSTRUCTION:**

The construction of the filter follows the same lines as the DAC described in EM vol 2/3. Firstly cut the veroboard to be a snug fit in the plastic box, and break the copper strips where indicated with 'x' in Fig 2. Assemble the board carefully taking care to position the components correctly. It is suggested that the wire links are inserted first, followed by the IC sockets if used (recommended for inexperienced constructors), then the resistors and capacitors. Finally fit veropins at the board edge to take the wires to the sockets and on/off switch.

When assembly is completed, double check all connections and component positions, insert the two IC's, taking care to get them the right way round, and put the board to one side.

\*\*\*\*\*

The next step is to drill the plastic box to take the on/off switch and the input & output sockets. These should be fitted at one end of the box to allow room for the batteries. Mark the positions of the holes and drill very carefully as the plastic will split easily. Hole sizes will depend on the exact component used.  
Before fitting the sockets and switch, fit the circuit board.

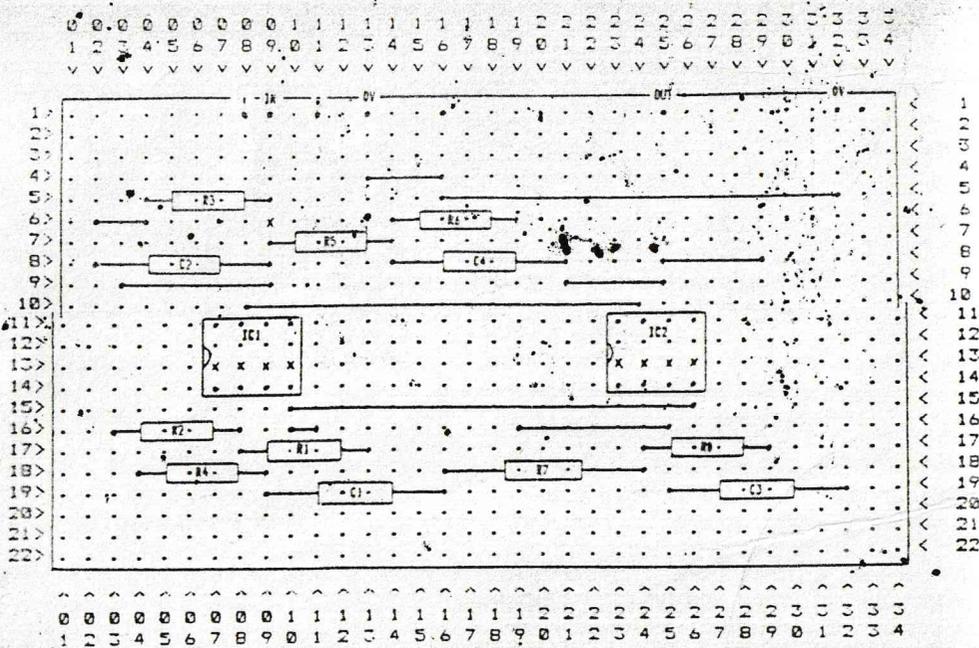


Fig 2

into the box, ensuring that the right hand end is adjacent to the end of the box where the sockets are to be fitted. If the board is a snug fit, then no fixing is necessary, otherwise it can be secured with double sided tape or mirror pads.

Finally fit the switch and sockets. When wiring the sockets, ensure that the veropins marked IN & OUT go to the centre pin connector on the socket. The earth connection on each socket should be wired together and taken to one of the veropins marked 0V. Now wire up the on/off switch to the circuit board and the battery connectors as shown in Fig 3.

\*\*\*\*\*

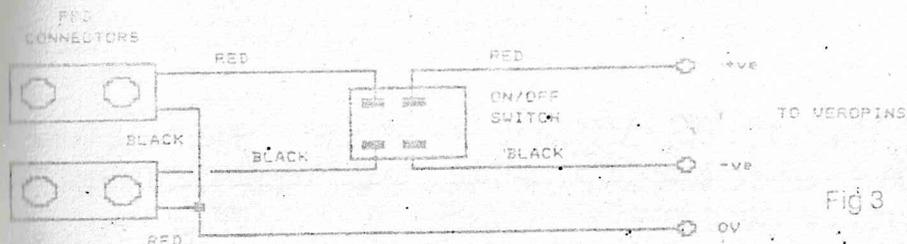


Fig 3

Having done a final check of all work, connect the output of the DAC to the filter. Input, and the filter output to the amplifier. Connect 2 PP3 batteries and switch on. Play your favourite piece of music and notice the much improved sound quality. Please note that the filter has about 8dB amplification, so you will need to run your amplifier at a lower volume setting than with the DAC alone.

When you are happy that all is working correctly, place the batteries on top of the circuit board, packing with small pieces of foam or bubble packing to stop any possibility of them moving about, and secure the lid. Mark the input and output sockets accordingly.

If you are building both the filter and DAC then they could both be mounted in the same box, although the one listed will not be large enough. Maplins stock many such enclosures.

COMPONENTS LIST:

- R1 39K 1% Resistor (orange white orange brown) M39K
- R2 5K6 1% Resistor (green blue red brown) M5K6
- R3 3K3 1% Resistor (orange orange red brown) M3K3
- R4 3K3 1% Resistor (orange orange red brown) M3K3
- R5 3K3 1% Resistor (orange orange red brown) M3K3
- R6 3K3 1% Resistor (orange orange red brown) M3K3
- R7 39K 1% Resistor (orange white orange brown) M39K
- R8 47K 1% Resistor (yellow violet orange brown) M47K

- C1-4 10n 1% Capacitor BX86T
- IC1-2 uA741 Op-Amp QL22Y
- Veroboard 34 strips \* 22 holes FL10L
- Small plastic box LF01B
- 2 \* 3.5 mm jack sockets HF82D
- 2 \* 3.5 mm jack plugs HF80B
- Miniature DPDT switch FH04E
- 2 \* PP3 battery clips HF26F
- 2 \* 6 pin dil IC sockets BL17T
- 7 \* veropins FL24E

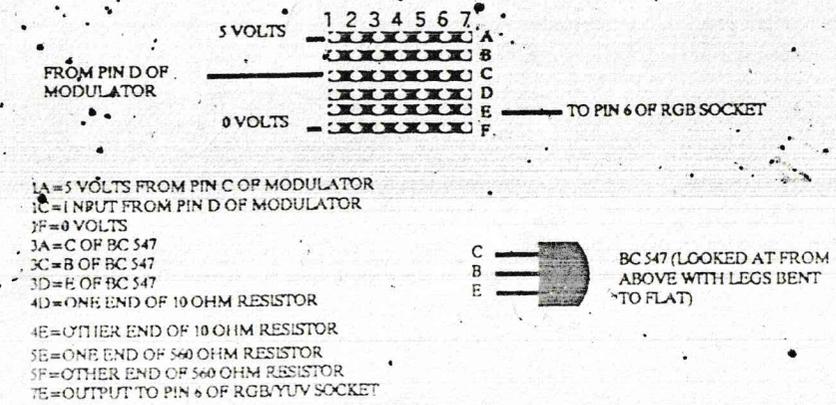
# COMPOSITE VIDEO A PROJECT FROM STUART MARSHALL TO IMPROVE THE QUALITY OF ALBERTS OUTPUT TO A VIDEO RECORDER.

If you have ever tried video recording Graphdraw screen from Alberts T.V. output, you may have found the results left something to be desired.

What we are actually going to do is supply the video with a composite video input. If you look at your video manual you will see where to input the signal.

There are two ways we can go about providing our composite video output from Albert, both require you to lift the lid from your machine and introduce a soldering iron, so if you are unsure of your abilities it may be a good idea to give this project a miss or get someone to do it for you.

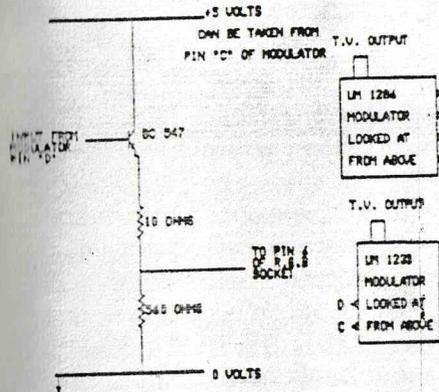
The first method is the simplest though also the least reliable, all that is required is a wire soldered from pin D of Alberts T.V. modulator, (see diagram), to pin 6 of the YUV/RGB socket at the rear of the Einstein. Now simply connect a lead from the YUV/RGB socket (using a 6 pin din plug) to your videos composite input (either BNC or UHF plug depending on your machine), ensuring that pin 6 on the Einstein plug goes to the centre pin on your video plug. Pin 5 on the Einstein plug should go to the outer (earth) on the video plug. This will work for nine videos out of ten, if you have the one it will not work for, then you need method two.



- 1A=5 VOLTS FROM PIN C OF MODULATOR
- 1C=INPUT FROM PIN D OF MODULATOR
- 1F=0 VOLTS
- 1A=C OF BC 547
- 1C=B OF BC 547
- 1D=E OF BC 547
- 1D=ONE END OF 10 OHM RESISTOR
- 1E=OTHER END OF 10 OHM RESISTOR
- 1E=ONE END OF 560 OHM RESISTOR
- 1F=OTHER END OF 560 OHM RESISTOR
- 1E=OUTPUT TO PIN 6 OF RGB/YUV SOCKET

For method two you will need:-

- 10 ohm resistor 1 off
- 560 ohm resistor 1 off
- BC 547 transistor 1 off
- some wire and a piece of veroboard 6 lines \* 7 holes

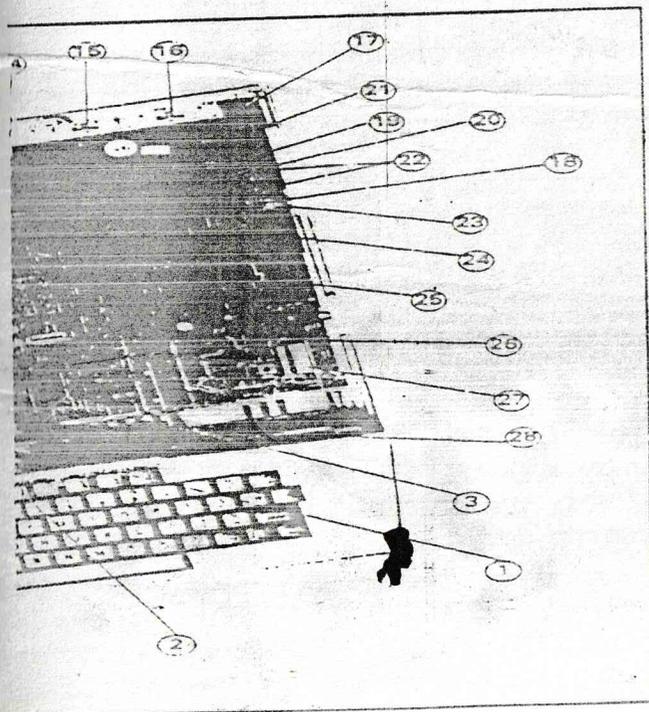


Wire up the small cct board as laid out using the table on the previous page.

Locate the de-modulator, item 27 in the picture below and connect as instructed.

Once connected up, link the output on pin 6 of the 6 pin Din to your video as per the previously stated arrangement.

You should now be able to use your Einstein with your video without any problems.



1. Keyboard connector
2. Loudspeaker
3. Video display generator
4. Sound generator
5. 3 1/2 compact floppy disk drive
6. Power connectors
7. Disk drive ribbon cable
8. 64K RAM
9. Switched mode power supply
10. System firmware ROM
11. Buffers
12. External disk connector
13. Mouse 'Mouse'
14. Z80 microprocessor
15. User port
16. Printer port
17. Monitor output socket
18. Analogue to digital converter
19. Parallel interface chip
20. Serial interface chip
21. RS232-C serial port
22. Analogue input
23. Volume control
24. Counter/timer
25. Spare ROM position
26. Power connector for second disk drive
27. UHF modulator
28. Position for second internal disk drive

## EINSTEIN MACHINE CODE MONITOR SYSTEM

Did you know that MOS has a powerful debugger? The monitor is the section of MOS which can be controlled from the keyboard and it contains commands which allow individual instructions in a program to be examined as they are processed by the Z80 chip -- although reading the DOS/MOS manual is unlikely to produce any enlightenment until you know what you need.

There are two ways to debug a program: you can insert write statements by trial and error until you find the variable which fails to change as it should. Or you can step through the program to the stage at which the error appears and then follow the register changes which occur. The second method requires a program called a monitor (or various other names). There are Public Domain (e.g. Z8E) and commercial programs which provide sophisticated control over the progress of the run if you can master all the commands. The MOS monitor is simpler to use, but its main advantage is that it is in ROM and so there is only minimal code exposed if your program goes off the rails. As well as debugging new programs, the monitor is very useful for finding the correct location for modifications or why a perfectly legitimate instruction causes Syntax Error.

### PREPARATION

There is no doubt that the quickest way to learn machine code is to step through a short program. You will need a list of Z80 opcodes in numerical order, together with their operands (that is the bytes which follow each opcode). The list should show the number of bytes and type of operand -- data, address, port, offset etc. Omit the codes DD and FD (using registers IX and IY) until you understand the other codes. You will probably have to write out the list yourself, but it is worth checking as many assembler and disassembler manuals as you can find. Assembly language textbooks tend to copy the Zilog manual which gives the codes in binary and needs a lot of work to sort out.

For example the assembly language instruction CALL is represented by code CD and is followed by two bytes specifying the address of a subroutine to be called (remember that the HIGH byte is SECOND). So your list will contain...

CODE	OPCODE	BYTES	TYPE
CD	CALL	2	Address.

Making a list like this takes you half-way to learning machine code. It will also become obvious that there are various ways to condense the list into a reasonably small space.

The opcodes are always given as hexadecimal numbers but don't let this worry you. There is no arithmetic needed for opcodes, so think of them as names. The only time arithmetic is needed is to convert relative jumps to addresses and MOS provides the A instruction to do this for you.

### STARTING THE PROGRAM

Getting started is the most difficult part and the DOS/MOS manual gives no clues. The simplest procedure is to use the LOAD command and then go into MOS. If you have a short well-behaved program TEST.COM, proceed as follows: Remember to type <ENTER> after each line

PROMPT	TYPE IN	COMMENTS
0:	LOAD TEST.COM	Reads program into memory
0:	MOS	Enters
MOS >	G 0100 0100	Set up monitor

This will only work if the program

- (1) makes no use of the system stack and
- (2) does not expect a command tail.

The command tail is anything which you would type after TEST when the program is started normally. Look for it in the buffer at address 080H.

There are many ways of starting the monitor and most of them work most of the time but you are always liable to run into obscure errors on occasion which will waste a lot of time. After experimenting for more than three years, I have found that the simplest way to be sure that the monitor is set up correctly is the following sequence (assuming that you would normally start the program by typing TEST Q.DAT):

0:	LOAD TEST.COM	Begin as above
0:	MOS	
>	G 0 0100	Dont confuse G0 and GO
0:	GO Q.DAT	Load command tail

The monitor is now set up at the start of the program with the registers displayed to prove it. This procedure should be used even if there is no command tail, because it ensures that there is a valid stack. The DOS/MOS Manual has a good description of the effect of these instructions which is much easier to follow when you know what needs to be done. The second address in

the G instruction can be in the middle of the program when you have located a suitable position.

An alternative procedure if you want to skip straight to the middle of the program is to modify the .COM file by placing an FF code at the required start address. This can be done with a disc editor or from MOS using LOAD and SAVE (remember to note the number of blocks). The program is run with a normal call and it will stop in MOS when it reaches the FF code. Change the FF back to the original value with the M instruction and you are ready to start.

### RUNNING THE PROGRAM

The MOS E instruction is the core of the monitor. When you have set up the program, type

```
T 0100 010F
```

to display the first sixteen bytes of code. Suppose the value at 0100H is 31 (LD SP, ): this loads the stack pointer and is followed by the address (2 bytes) to which the SP register should point on completion of the instruction. The whole instruction occupies 3 bytes and therefore the next opcode is at 0100H + 3 = 0103H. So type

```
E 0103
```

The Z80 registers are displayed, showing that the program counter (PC) is now at 0103H. To check the SP register type

```
Z2
```

The next opcode is determined (the code display should still be on the screen) and the whole process is repeated. This leads to single-stepping through the program. When you get more expert, it will be possible to jump further ahead than one instruction: whatever address is specified after E, the program will run on until it encounters an opcode at this address, which is called a breakpoint.

The E instruction works by changing the opcode at the address typed to FF, which calls MOS, so if you have not calculated the address correctly, the program will either run on to the end or more likely get lost because you have altered an address instead of an opcode. When this occurs, it is necessary to start again, but it is possible to jump directly to the last breakpoint encountered. For example if the last breakpoint examined was at 010FH you would start up the program again using the following in place of the G instruction above.

```
G 0 010F
```

### JUMPS

The first codes you must learn to recognise are the jump codes, because the next address may be in another part of the program. Unconditional jumps (C3) are easy -- the next address follows directly after the opcode. Relative jumps (18) need more attention because you have to calculate the address. For example if the program is

```
0108 18 02
010A 3E 01
010C B7
```

the value 2 at 0109H is the relative jump, but it must be added to the NEXT address, that is 010A + 2 = 010C, so control passes from 0108H to 010CH.

Conditional jumps require some care. It is necessary to check the flags given in the last register display (Z0 if it has scrolled off). For example code 20 (JR NZ, ) will continue to the next instruction if the Z flag is 1, but will do a relative jump if the Z flag is 0.

### CALLS

A call instruction (CD) goes to the address specified in the next two bytes in the same way as a jump instruction (C3). However when a return (C9) instruction is subsequently encountered, control NORMALLY returns to the opcode following the CD instruction, by fetching the address from the top of the stack. It is not difficult to program alterations to the stack so the less well-behaved programs may return to another location. For this reason when you reach a return instruction, it is always safer to check the new address by typing Z2, and then displaying the contents of memory at the address in SP using the T instruction. This shows the contents of the stack, and the program jumps to the address in the first two bytes.

### LOOPS

Tracing a loop can become very tedious even when there is a fairly low loop count, and the risk of mistyping rises rapidly with the count. If there is a single exit condition, skipping out of the loop is easy. With multiple exit points it may still be possible to work out a common meeting location for all exits, but if one exit is a conditional return and the other is a jump, it may be necessary to try each in turn and restart if the guess is wrong. Loops are the most difficult code to trace but in practice the constraints of assembly language programming frequently lead to short loops with long blocks of code placed in subroutines and long loops are not very common.

CONCLUSION

Single-stepping through code is necessary to follow register changes, but it is far too slow for working through a whole program. Faster progress can be made by counting through the displayed code to find the opcodes and setting a breakpoint at the next instruction which could produce a jump. Further improvements in speed will always need some degree of guesswork: for example skipping subroutine calls will make good progress through the code, but you will eventually come to a routine which does not return. In the best programs this would indicate that the program had detected a fatal error condition, but there are many other possible causes which can only be identified by tracing the subroutine.

Always keep a note of addresses and important values when tracing, so that if you make a mistake (which is only too easy) you can readily return to the same position by restarting. Programs selected for tracing should preferably have been written in assembly language. Compiled programs usually generate highly convoluted code, particularly if an intermediate code is produced. This is a list of machine code subroutines with a very short loop selecting each entry point. I find with this type of program (and in fact any file-using program) it is best to start by using E 05 repeatedly. This traps the DOS calls and allows you to follow what is being done to the files at each stage and provides a useful outline of the

program: Of course you need to read the return address from the stack and set a breakpoint at it in order to get to the next DOS call. Some programs have a sign on message output by single character DOS calls and you may have to work up through several subroutines to find the loop exit point.

It has only been possible to mention a few of the many code variations, with the aim of showing newcomers how to get started and providing some new ideas for regular users, but in this type of programming there is no substitute for experience. C.P. Wallis Nov. 1988

256\*256\*256\*256\*256\*256\*256\*256\*256\*256\*256\*256\*256\*256\*256\*256\*256

AUTO-BOOTING THE 256 (M/C)  
 Peter Mansell provides a utility for the 256. (The boots on the other foot?).

Using 'Wordstar' on the '256' requires 80 column mode and a switch to the ASCII character set, (which needs to be set either from MOS or via the DIP switches). If the following bytes are saved to the 'Wordstar' disc as an Auto-boot COM file it will prevent a lot of finger exercising. I saved the few bytes as BOOT.COM.

CODE:- 21 01 00 CF 8B 3E 10 CF 9E 97 00

To Auto-boot this or any other program requires a sub-routine to add to DOSCOPY (supplied with the '256').

First load DOSCOPY, go into MOS, and using the Modify command alter address &01E1 with..

01E1 CD <E>  
 01E2 80 <E>  
 01E3 03.<E>

Next enter the sub-routine at &0380 using the Modify command.

0380 210812E50618AF77 <E>  
 0388 2310FCCFCF4E414D <E>  
 0390 45204F462050524F <E>  
 0398 4752414D3F20A0CF <E>  
 03A0 A70600E1E523E5C5 <E>  
 03A8 CF9ECP9CC1FE0D2B <E>  
 03B0 0604E1772318EFE1 <E>  
 03B8 E170210010454DC9 <E>

On running the routine, follow the prompts and enter the name of the program you wish to Auto-boot without the .COM extension.

SCRATCHPAD LOCATIONS

Joseph Karthausser, of Synclavia and MIDI fame, has sent us a complete list of the scratchpad locations as used by the operating system. The Scratchpad is initialised by the MOS when the Einstein is switched on. These memory locations will then hold information on the state of various parts of the system, i.e. disc status, screen colours, position of cursor etc. you can use or alter the values held in the scratchpad locations within your own programs. From Basic you will need to use the commands PEEK or POKE, from MOS use the Modify option. There are some examples on this at the end of the list. The scratchpad starts at location FBBOHex.

Scr Equ	0FB00h	;Start of scratchpad
;Scr+00h		;Interrupt vector for CTC 0
;Scr+02h		;Interrupt vector for CTC 1
;Scr+04h		;Interrupt vector for CTC 2
;Scr+06h		;Interrupt vector for CTC 3 / RTC
;Scr+08h		;Interrupt vector for Keyboard
;Scr+0Ah		;Interrupt vector for A/D Converter
;Scr+0Ch		;Interrupt vector for Fire button
;Scr+0Eh		;Interrupt vector spare for user use
;Scr+10h		;Interrupt vector for Printer
;Scr+12h		;Interrupt vector for User Port

\*\*\*\*\*

```

RdMem  Equ  Scr+014h ;Routine to read memory from ROM
XecEnd Equ  Scr+017h ;End of a GO command from MOS
UpCopy Equ  Scr+01Ah ;Routine to block copy up
DnCopy Equ  Scr+021h ;Routine to block copy down
McXec  Equ  Scr+028h ;Routine for external call into MOS

BrkVct Equ  Scr+030h ;Break vector
.CldVct Equ  Scr+032h ;Cold start vector
WrmVct Equ  Scr+034h ;Warm start vector
VdpMpd Equ  Scr+036h ;Video mode
TColr  Equ  Scr+038h ;Text colour
GColr  Equ  Scr+039h ;Graphics colour
FmVct  Equ  Scr+03Ah ;Mos function table vector
FuVct  Equ  Scr+03Ch ;Mos user function routine vector
Flags  Equ  Scr+03Eh ;System flags:  bit 0 : Key in scan buffer
                        ; bit 1 : Function key in process
                        ; bit 2 : Alpha lock
                        ; bit 3 : Compare flag for scan
                        ; bit 4 : Graph/Alpha key
                        ; bit 5 : Control key
                        ; bit 6 : Shift key
                        ; bit 7 : Upper/Lower case

CusCde Equ  Scr+03Fh ;Cursor character
Prompt Equ  Scr+040h ;Prompt character
Blink  Equ  Scr+041h ;Cursor blink rate
RptLng Equ  Scr+042h ;Key repeat delay
RptSht Equ  Scr+043h ;Key repeat speed
SctSiz Equ  Scr+044h ;Size of disc sector
PcFlgs Equ  Scr+045h ;Syst flags 2 : bit 0 : Cursor flash/Cursor off
                        ; bit 1 : Printer echo On/Off
                        ; bit 2 : 80 Column card selected
                        ; bit 3 : Cursor line addressing
                        ; bit 4 : 0 X-Coord / 1 Y-Coord
                        ; bit 5 : NOT USED
                        ; bit 6 : NOT USED
                        ; bit 7 : Inverse text flag

LastKy  Equ  Scr+046h ;Code of last key pressed
ScrnCd Equ  Scr+047h ;Code of character at current cursor location
Store  Equ  Scr+048h ;Temporary store for key during scan
ArgFlg Equ  Scr+049h ;Temporary argument store for Monitor
VCol  Equ  Scr+04Ah ;Screen column
VRow  Equ  Scr+04Bh ;Screen row
ScrLsZ Equ  Scr+04Ch ;Used during scroll of memory text table
    
```

\*\*\*\*\*

```

ScrOnt Equ  Scr+04Eh ;Scroll offset on VDP
XLen   Equ  Scr+04Fh ;Line length
HstDsc Equ  Scr+050h ;Current drive
HstTrk Equ  Scr+051h ;Current track
HstSec Equ  Scr+052h ;Current sector
HstDma Equ  Scr+053h ;Current disk buffer address
RWFlag Equ  Scr+055h ;Read/Write status flag
Status Equ  Scr+056h ;Error status value
ErOnt  Equ  Scr+057h ;Number of tries before indicating error
TrkTbl Equ  Scr+058h ;Track number for drive 0

;Scr+059h ;Track number for drive 1
;Scr+05Ah ;Track number for drive 2
;Scr+05Bh ;Track number for drive 3

BpVal  Equ  Scr+05Ch ;Break point value
BpAdr  Equ  Scr+05Dh ;Break point address
VdpSt  Equ  Scr+05Fh ;Vdp status
OldI   Equ  Scr+060h ;Registers for MOS Z command
OldIX  Equ  Scr+061h

OldIY  Equ  Scr+063h
OldSP  Equ  Scr+065h ;Can use OldSP for Fill stack
FillSP Equ  OldSP

OldAF1 Equ  Scr+067h
OldBC1 Equ  Scr+069h
OldDE1 Equ  Scr+06Bh
OldHL1 Equ  Scr+06Dh
OldAF  Equ  Scr+06Fh
OldBC  Equ  Scr+071h
OldDE  Equ  Scr+073h
OldHL  Equ  Scr+075h
OldPC  Equ  Scr+077h
SavHL  Equ  Scr+079h ;Temporary store for HL
SavSP  Equ  Scr+07Bh ;Temporary store for SP
Disc   Equ  Scr+07Dh ;CP/M current disk drive
Trak   Equ  Scr+07Eh ;CP/M current track number
Sect   Equ  Scr+07Fh ;128 byte sector number for DOS
DBuf   Equ  Scr+080h ;CP/M Disk buffer
SekHst Equ  Scr+082h ;Seek Shift-Right count
UnaDsc Equ  Scr+083h ;Last unallocated disk
UnaTrk Equ  Scr+084h ;Last unallocated track
UnaSec Equ  Scr+085h ;Last unallocated sector
    
```

## SECTOR SKEWING ON A 256

Mr Peterson from Nottingham has sent in this interesting item for the 256.

Having recently upgraded my QL from a pair of 40 track, single sided drives to an 80 track double sided drive, the box containing the old drive has been attached to the previously diskless Einstein TC01. Drive 0 is in fact the original TEA 3" unit from the Einstein and work correctly. Drive 2 is a 3.5" Panasonic item that originally came as part of an Opus Discovery for a Spectrum, and has worked for several years with both that and the QL. Unfortunately I couldn't use it with the Einstein as all attempts to format a disk in it gave a "No Drive" error. After making a suitable 26 way to 34 way adaptor I tried it with the 256. Still nothing. Oh well, looks like the hard way. I disassembled BACKUP.COM and ran it from MOS to see where the error was occurring. It always formatted the first sector OK but dropped out with the error before finishing the first track. Bearing in mind that the QL used 9 sectors per track, and the Spectrum 18 sectors of 256 bytes each, it seems reasonable to assume that it was having trouble fitting 10 sectors onto the disk, possibly due to the drive running slightly fast. After altering the program to write shorter gaps between sectors, it worked properly.

While sorting out this problem I couldn't help noticing that the sectors are numbered 0 to 9 in numerical order on odd numbered tracks, with even numbered ones being offset by 5 sectors. Knowing that the QL disk systems conventionally place their sectors in the order, 0, 3, 6, 1, 4, 7, 2, 5, 8 to allow the system to deal with the current sector before the arrival of the next one at the read/write head, and suspecting that the 8 MHz 68008 generally shuffles blocks of memory at least as fast as a 4 MHz Z80, I started experimenting. The sector numbers are stored in a table at the end of the BACKUP.COM or FORMAT.COM are easily rewritten using MOS. A number of arrangements were tried, a simple test of efficiency being the time taken to copy 46 files (184K) from drive 0 to 1 using COPY \*.\* TO 1:. The standard format on both disks gave a time of 364 seconds. The order currently in use, 0, 5, 1, 6, 2, 7, 3, 8, 4, 9, gives a time of 224 seconds. While all you serious TC01 users out there presumably use System 5 or 80, which I believe use a different sector layout, us neglected 256 owners are stuck with the supplied operating system. I suspect that this modification is therefore of more use/interest to those lucky(?) few, so here a brief guide:

```

From DOS          LOAD FORMAT.COM
Into MOS          MOS <CR>
                  M 054A <CR>
                  0501060207030804090005010602 <CR>
                  . <CR>
                  Y <CR>
Back to DOS      & save it      SAVE 5 FORMAT2.COM <CR>
    
```

It costs nothing, it takes an hour or so to alter ones current disk, it should save wear on both disks and drives, and, amazingly, it almost make WordStar usable.

```

HstAct Equ Scr+086h ;Host active flag
UnaOnt Equ Scr+087h ;Unallocated record count
HstWrt Equ Scr+088h ;Host writted flag
ReadOp Equ Scr+089h ;Read/Write flag (1 = read)
RsFlag Equ Scr+08Ah ;Read sector flag
WrType Equ Scr+08Bh ;Write operation flag
Time Equ Scr+08Ch ;0 bytes for Hrs,mins,secs
Busy Equ Scr+092h ;Busy flag for interrupting devices
FnKey Equ Scr+093h ;Pointer for function key in video area
FnPtr Equ Scr+094h ;Pointer to funtion key area in video RAM
X1 Equ Scr+096h ;Destination address of draw to X
Y1 Equ Scr+098h ;Destination address of draw to Y
X2 Equ Y1 ;X2 is right end of line in Fill
OrgX Equ Scr+09Ah ;X Origin
OrgY Equ Scr+09Ch ;Y Origin
CX Equ Scr+09Eh ;Polygon centre X Co-ordinate
CY Equ Scr+0A0h ;Polygon centre Y Co-ordinate
RadX Equ Scr+0A2h ;Ellipse X Radius
RadY Equ Scr+0A4h ;Ellipse Y Radius
CiNc Equ Scr+0A6h ;Number of polygon sides
DotOn Equ Scr+0A8h ;Dot on
DotOff Equ Scr+0A9h ;Dot off
DotOn2 Equ Scr+0AAh ;Dot on 2
DotOf2 Equ Scr+0ABh ;Dot off 2
DotCnt Equ Scr+0ACh ;Store for dot count for DotOn/DotOff
FillMod Equ Scr+0ADh ;Fill mode (Background = 0, Foreground = 1)
StP Equ Scr+0AEh ;Pointer to current level of fill stack
StepR Equ Scr+0B0h ;Stepping rate for disk
SidFlg Equ Scr+0B1h ;Side flag (Single = 0, Double = 1)
ScanKey Equ Scr+0B2h ;Key scan location
DpbPtr Equ Scr+0B3h ;Disk parameter table pointer
KCnt Equ Scr+0B5h ;Repeat key count
Buffer Equ Scr+0B8h ;Input buffer for MOS entries
VdpBuf Equ Scr+0E0h ;VDP buffer for copying to/from VDP
BankZ Equ Scr+0109h ;Time interrupt service routine
    
```

A very useful scratchpad is StepR at location FBB0, this controls the stepping rate of the disc drives attached to the Einstein, the default is 01 which equates to 12milli-seconds, some drives, especially older full height types, require a slower stepping rate to work on the Einstein. To alter the stepping rate to 20ms change FBB0 to 02 or to 30ms to 03. Enter MOS, then type M FBB0<E>, the display will now show 01, type 03 (don't forget the full stop) <E>. To see the effect of altering locations try Modifying location FB38, MOS<E>, M FB38<E>, 00!!!

Ian Collins  
The Old Dairy  
2 Coombe Lane  
TORQUAY  
Devon TQ2 8DY

Dear Mr. Adams,

I saw your details in the Computer Shopper of September 1995.  
I am a former user of an Einstein computer and have the following items to sell.

Hardware:-

Einstein Computer TC01 with monitor TM01 - twin floppies.  
Tatung 80 column card TK02  
Spectrum Emulator by Syntaxsoft Ltd. with 40-80 column switch by Screens Microcomputers.  
Benchmark Data Transfer Switch - two computers to one printer or one computer to two printers.  
ZIP STIK Joystick.

Software:-

CP/M 2.2  
MicroPro WordStar Professional, DataStar, ReportStar, MailMerge, SpellStar & StarIndex with all manuals in five library cases.  
KUMA WDPRO, Database, Spreadsheet, Home Budget & Non-VAT accounts with all manuals.  
Tasword, Tasprint and Tas-sign Einstein with manuals.  
The Cracker spreadsheet with manual.  
Cash Trader by Quest with manual in case.  
The G4VPD Collection - RTTY, Morse & TNC programs.  
Einstein System Master with BBC Basic & DR Logo.

Books:-

2 copies " An Introduction to Einstein" TATUNG 1984  
"DOS/MOS Introduction" TATUNG 1984  
"Dr. Logo Introduction" TATUNG 1984  
"BASIC Reference Manual " TATUNG 1984  
"BBC BASIC (Z80) Reference Manual" TATUNG 1984

Games:-

Einsoft Super Six  
KUMA Pakman & Millipede

Magazines:-

Tatung Einstein User Vols 1,2,& 3 and 4 to No.3.

Have you any suggestions to make ? I would prefer to sell the whole as one lot.

Yours sincerely,

*Ian Collins*  
Ian Collins

3610

Miss Linda M. Howard,  
6, Fry Crescent,  
Oakhurst,  
Burgess Hill,  
West Sussex,  
RH15 8TP  
Telephone: (01444) 247628

Tony Adams, Esq.  
Einstein News,  
Ivy Cottage,  
Church Road,  
New Romney,  
Kent.  
TN28 8TY

Dear Tony,

RE: ABOVE MEMBER UKBUG 1441 - 82

Please advertise the following in the "Market Place" in your next magazine :-

£150.00 FOR THE WHOLE LOT, BUYER COLLECTS

Einstein Computer TC01 - 8" single floppy disc drive, TM01 colour monitor, fitted with TK02 - 80 Column Card  
Twelve 3" disks for formatting, Printer cable and 40/80 Column switching device  
Pristine Einstein Computer Upgrade Kit - second disc drive, still in its original packaging

Quendata Daisy-wheel Printer with TWO type-faces - Courier and Script, Operating Manual and Two new ribbons

Original Einstein Manuals - DOS/MOS Introduction  
- An Introduction To Einstein  
- BASIC Reference Manual  
- BBC BASIC (Z80) Reference Manual  
- Dr. Logo Introduction  
- 80 Column Card Unit Instructions  
- Einstein Reference Card

Software and Manuals as originally supplied - Master Disc  
- Database  
- Spreadsheet  
- Wdpro  
- Basic Tutorial  
- Programmers Kit  
- Pete's Utilities  
- Spell  
- Solo Maxima  
- The Cracker  
- Kuma Non-VAT Accounts  
- Small Business Accounts and VAT Made Simple  
- Super Six Games Pack  
- Oh Mummy  
- Cnuckie Egg  
- Disco Dan  
- FI Simulator/Soul of a robot  
- Buzzoff/Sharkhunter  
- Castle Quest  
- Cursed Chamber & Zrin

Separate Books

Einstein Magazines

- Programming the BBC Micro  
- Games BBC Computers Play  
- Tatung Einstein News Issue 1 (Dec. 1985) Issue 2 (May 1986)  
- Einstein User Volumes 1, 2 and 3 : Numbers 1,2,3 and 4 for each.  
Volume 4 : No's 1 and 3, Volume 5 : Numbers 1, 2 and 3  
- Alternative Micro News: Volume 1 Numbers 2, 4, 5.  
- All Micro News: Volume 1 Numbers 2, 7, 8, 9, 10, 11, 12, Volume 2 Number 1  
- Einstein Monthly: Volume 2 Numbers 8, 9, 10, 11, 12, Volume 3 Numbers 1 and 2  
- Combined All Micro News/Einstein Monthly etc. Issues 65 - 76

Many Thanks,  
Yours sincerely,

*Linda Howard*  
Linda M. Howard

James LISTON, 1 Sandgate Hill, Folkestone, Kent, CT20 2JF (phone 01303-254438, fax 01304-205146) seeks an offer for his 2-drive Einey, vgc, with 80-col card, TM-01 colour monitor, plus Wordstar/Datostar/Reportstar, KUMA WdPro/Database/Spreadsheet, FMA Personal Assistant, games software, etc, etc, etc, and full set of Manuals.

Ian Collins ("for sale" letter reproduced elsewhere in this mailing) has also found a spare Einey keyboard, but missing one keyswitch (used on a repair job). Also:- Micro Simplex Accounts, AMTAT (utility to read (and run?) Amstrad disks on Einey), and a null-modem cable, with extension to 12ft., connecting Einey 5-pin DIN to PC 9-pin serial connector, used to transfer lots of data files across. Ian says that all his software is original and legal, and confirms that the 80-col card is in gwo.

Just Received. Newsletter No.961 from Bull Electrical, 250 Portland Rd, Hove, Sussex, BN3 5QT (Phone orders 01273-203500, fax orders 01273-323077). If you haven't a use for Chieftain Tank gun target lasers at 349.00 plus VAT (and postage?), you may still have a use for short mains extension cables incorporating RCB units at 6.99 (or 4.99 each for 10 or more), Amsoft 3" disks at 15.00 for a box of 10, portable double glazed solar showers at 14.00, PC psu's (gwo) at 5.99 (or untested customer returns units at 9.95 for 5), 2-way mirror kits, or last 50 incoming phone calls recorder/display unit, or police car strobe bulbs.

Listed in their Bargain Hunters' Sheet is A4 dtp monitors, brand new, no data or info, 5.99 each, ref. BAR300. Have we any members out there who understand monitor technicalities? How about buying one of these to play with, and tell us if they can be put to use on an Einstein, and if so, what mods are needed?

Bull are also offering refurbished 1.44Mb 3.5" drives at 12.50 each. A number of other dealers frequently offer second-user 3.5" drives. Do we have a disk drive expert member out there, who can either produce simple diy instructions for adapting the various different types of 3.5" drive to run on Einey as a standard 80-track 720Kb external(/internal?/internally-wired external?) drive, or can put together working 3.5" drive upgrade kits to supply to members?

\*\*\*\*\*

#### SOMETHING NEW FOR THE EINSTEIN?

Andrew Dunipace has been working for some time on an Einey diskzine, and this now looks very promising indeed. A printed mag is very good at distributing many sorts of information, but other things are better done on disk. If each concentrates on complementing the other, not competing with it, Einey users can only benefit from this additional support. We're exploring with Andrew how we can actively co-operate with him on this project, and we'll bring you more news shortly.

#### ADDING AN EXTERNAL DISK DRIVE

Involves some difficult Dos problems. If you upgrade your Dos it costs money, and you risk trashing your disk contents by having directory tracks that are invisible to the Dos if the Dos sector on a disk that you boot up is configured differently from your actual configuration, or differently from the formatting on a disk that you're using. Or else you simply don't upgrade, and your disks are limited to 40-tracks, single-sided. Steve Potts has the answer -- even more incompatibility! A simple modification adds a side-select switch to your drive, and the operating system sees each side of the disk separately, just like the 3" disk drives -- but without having to flip the disk. Details coming soon in your favourite Einstein magazine.

Also more David Williams XBAS listings to tap in and learn from, plus fascinating experiments inside Einey with Les Foskett.

Profuse apologies if you've been inconvenienced by the delay in getting this issue of the magazine out to you, and by some pages being rather hard to read. Many causes contributed to this, the major ones being a rotten British Gas central heating system, & a rotten roof on a 3-centuries-old oak-beamed farmhouse. Or you could simply say that it was tempting fate way, way too far, saying in the editorial that if B&H really couldn't manage to deliver the Einstein magazine that their subscribers had paid them for, they ought to turn the job over to us!

In essence, we took a gamble and spent all the available cash on a good little copier that was making nice clean copies -- to give you a big bumper combined 2nd Compendium and 10th birthday magazine issue -- but expanding this special issue from 24 to 76 pages meant we didn't have enough paper or toner to complete the job, we couldn't get more locally, so we took the copier to Manchester over Xmas to complete it.

The machine was happy in Kent, but in Manchester was exposed to a gas "Don't you just hate being totally unable to control the damn thing?" central heating system that cycled madly between arctic and equatorial conditions several times an hour. The copier simply went haywire and went wrong twice as fast as Yours Truly could fix it. Having spent all the available cash (and blown a main fuse on his credit card) buying toner and paper, Yours Truly struggled on until he had to admit defeat

At this point the council served a Dangerous Wall notice for a 300-years-old farmhouse ground floor outer wall that was leaning over 2 inches or so. Investigation of the cause revealed a highly dangerous roof, just about to collapse & take the rest of the farmhouse with it. The magazine necessarily had to go onto the back burner until Yours Truly had taken down the dangerous roof, shored up the rest of the building, and put up temporary roof and walls to "hold the fort".

Clearly this is no way to run a tramway, and cannot continue. We'll do our best to make up enough good copies of this issue from the chaos that has ensued, but please forgive us for a magazine that will inevitably have some pages that are far below the print quality that we'd hoped to reward you with. If it really is unreadable, please return it and we'll try and get the worst pages reprinted just as soon as we can. We are tackling the print problem in a number of ways, but we simply can't justify keeping you waiting any longer for this issue.

\*\*\*\*\*

\*\*\* Greenweld sell the Dragon/Tandy Joystick Interface as Cat. No. X6267. £1 each (+P&P £3) (Order on 01703-236363). Get one now, ready for Les Stanley's forthcoming "adapt it to Einey" article.

\*\*\* Bob Stewart (Glasgow) has a bog-standard double-drive Einey plus printer, serial cable, mags, software, etc, available, and may donate to to a club or school. You collect or pay carriage. (0141-946-2885)

\*\*\* Paul Hawkins (Carlisle) has a double-disk drive TC-01 to sell, with 80-col card, brand-new self-powered 5.25" external drive, stacks of software/books/manuals, plus everything Einstein, (even the original receipts). (Phone him on 01228-511656)

\*\*\* John Parry (Anglesey): Einey + monitor to sell.

01248-714655 (evenings) 01248-382950 (days)

WANTS: Les Foskett (01803 813677) needs a back-up Osborne 1 CP/M system disk as his has died. SD/DD (or preferably both);

Steve Potts (01400 261839) needs a CP/M Manual. (Perhaps the Amstrad version will do nicely?) If you get this issue before the Spalding show, why not offer to give him a hand on the Einstein stand there?

UK EINSTEIN USER GROUP - PUBLISHERS OF THE EINSTEIN MAGAZINE

\*\*\*\*\*

Steam Computer Society (UKEUG)  
Ivy Cottage, Church Road, New Romney, Kent TN25 8TY

UK Einstein User Group was formed in 1985 from a local group to provide users of Tatung's TPC-2000, Einstein TC-01, and Einstein E-256 computers with a magazine-based mutual self-help group. It offers help, advice, assistance and encouragement, and all members (from complete novices to real experts) are encouraged to pool their ideas, knowledge, needs and experience so that the magazine is a means of keeping in touch with each other, an inspirer of ideas, a practical help, and also a growing work of reference.

Many members have their machines, and are encouraged to share the knowledge about these too. A many non-Einstein machines no longer have their own user group. Where such user groups still exist we actively co-operate with them.

New (or re-joining) members are very welcome, and in most cases enjoy half-price discounts on our range of back numbers, programs from our software library, our range of commercial software, and more. If you are not already a member, send two stamps and your address at the top of this page for our current subscription pack and sample magazine.

To meet the needs of our users in all circumstances we have a flexible subscription rate with discounts for bulk purchase, advance subscription payments, life membership (available on request) and for those on reduced circumstances (make payment in cash or in kind). Volunteers serving a community have their subscriptions credited to them, and contributions made by our members for the issue they are published in credited to them. We do not charge for the use of our facilities, and inexperienced users are encouraged to write in with ideas, queries, and pleas for help. We will have it all to the whizzo.

It saves a lot of letters (ASCII text files if possible) plus a printer (have a printer connected). If not send the disk copy anyway. DO get it back! Please label the disk to show what format it is.

We provide limited support to non-members, but we do NOT subsidise them from subscription income. So if you own an Einstein (or some other machine without its own user group) it DOES make more sense to join than to be out in the cold!

We'll be reprinting Einstein Monthly Vol.1 No.1 to Vol.1 No.4 shortly. We can supply all back numbers since then from stock -- with terrific discounts if you buy a full set !!!

ENTIRE

CANON

Canon is a registered trademark of Canon Inc. and its subsidiaries.

The price of this magazine is £1.00 per copy plus postage.