July-September 1987 Volume 1. Issue 3

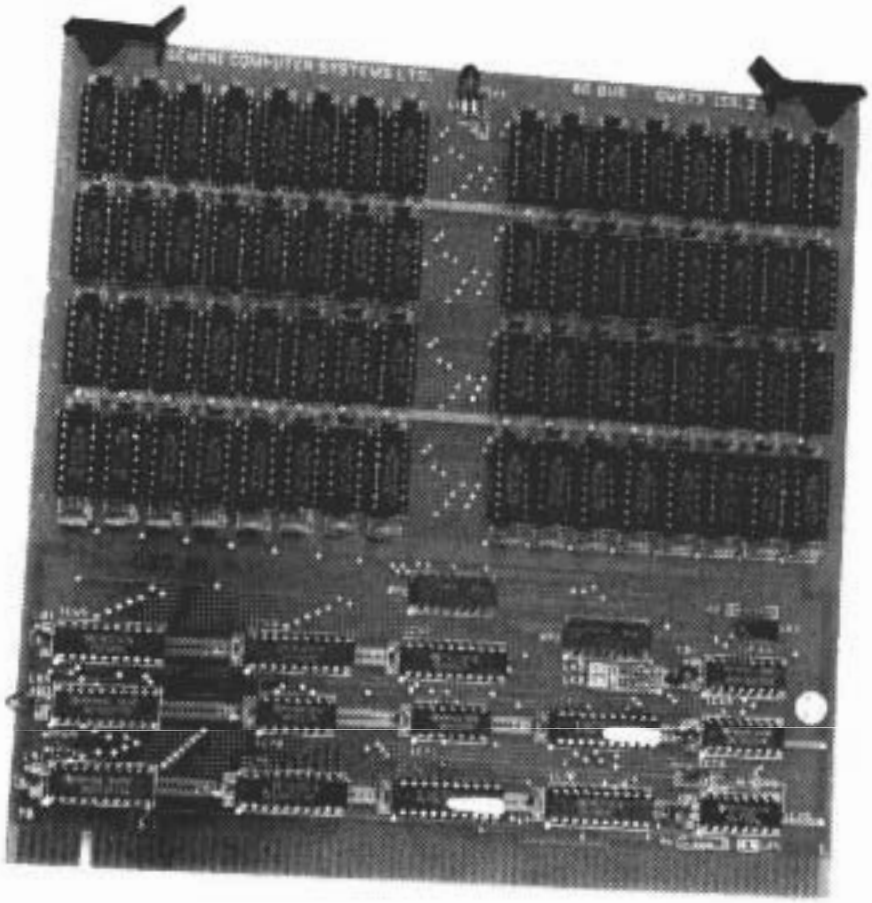
Scorpio News

Scorpio Systems
P.O. Box 286 · Aylesbury · Bucks · HP22 6PU

Gemini
Computer Systems Limited

News Release

GM873
2-Megabyte RAM Disk



Now available from stock.....

..... at only £595 plus VAT

Springfield Road, Chesham, Bucks HP5 1PU.
Telephone: (0494) 791010. Telex: 837788

JULY - SEPTEMBER 1987 SCORPIO NEWS VOLUME 1. ISSUE 3.

Contents

Editorial

Well, here we are again, and on-time again. It amazes me just how quickly three months goes by, and I'm glad that I'm not responsible for a weekly journal!

Response to Dave Hunt's "how about a small section on MS-DOS and IBM clones" has been very positive. 78% of respondents said "Yes", 11% said "OK, if you have to", and 11% said "No". On the basis of this alone there is absolutely no doubt that we should include further articles on clones/MS-DOS in the future.

Unfortunately, statistically this information is totally invalid, as only 9 people responded. So, how about this as a percentage of total readership, rounded to the nearest whole point? 1% said "Yes" and 99% didn't respond. Hmmm, that makes it rather more difficult!

When you receive the next issue of Scorpio News, you are going to be asked to vote with your cheque books. By then it would be nice if we knew what proposed future direction for the magazine is going to guarantee that the maximum number of people resubscribe. So this time, do please cast your vote (very topical this; as I write it's the General Election next week!). Let us know what you want to see happen. Are you going to resubscribe? If not, why not? What content do you want to see? Would you like the magazines more or less frequently? Would you like lower or higher subscription rates? Would you like more or less pages per issue? This is a very democratic publication!

We want to satisfy you, our customers, as satisfied customers give repeat business, which is what it's all about. But, do please be realistic too. Don't say "I want bigger, more frequent magazines for half the price", but work on the basis that more magazines means either more money or fewer pages; fewer pages means less money or more magazines, etc.

Nascom - 10 years on

This November sees the 10th anniversary of the launch of Nascom 1. I'm sure that some of you, as I do, see this as an ideal opportunity for a "happening".

We are quite willing to help arrange something, and again would ask for opinion on what that should be. The scope could range from half a dozen of us meeting in a Pub in the London area for a couple of drinks and a chat, to a hundred people at a week's conference in Jersey, with guest speakers such as Richard Beal, Dave Hunt, Doctor Dark, etc, plus representatives from various past and current 80-BUS suppliers! Of course, the cost of this might range from nothing to, say, £500 a head!

So, again, please drop us a line if you are interested. A quiet get together? Equipment demonstrations (your own, or manufacturers')? Talks? An afternoon, a night, a weekend? Chesham (where it all started), London, Scotland, Bahamas? How much are you willing to spend? If there seems to be any pattern to what people want (and please give some options/ranges), and if there is sufficient interest, then we will propose something in the next issue. I guarantee that there will be something happening, but it may just be John Marshall, Dave Hunt and I having half a pint in a Pub in Harrow! LET US KNOW WHAT YOU WANT.

Support our supporters

If you are purchasing something for your system, hardware or software, why not buy it from one of the companies that advertise in this or other issues of Scorpio News? They are supporting your magazine, so please support them. Let them know that your business is because of their advert here! And if your usual dealer doesn't advertise in Scorpio News, then tell him he ought too!

What's New?

New products continue to become available for 80-BUS based systems. Here are a few details - contact a Gemini dealer if you want to know more.

Gemini have released the GM873 RAM-DISK board. Exceedingly similar in looks and function to the GM833, a 512 Kbyte RAM-DISK board, the GM873 has a capacity of 2 Mbyte. All Gemini CP/M BIOSs, from Version 3.2 onwards, will automatically see and use it as drive 'M:'. Price is £595. (Add VAT to all prices here.)

PBM Memory Systems in Ireland have also launched a RAM-DISK board. This is fully compatible with Gemini's GM833, also with a capacity of 512K, but the PBM board is non-volatile. An on board battery gives two months data retention. The board can be supplied with or without memory devices, and can support any 28/32 pin device, including EPROMs. With 512K, £370. With 0K, £270.

A new range of frames and frame based systems are now available from Gemini. The frames are based on 19" 5U units and there are various optional power supply enclosures, drive enclosures, etc. The systems are targetted at Development markets, and are available in floppy or Winchester/floppy versions, with either Z80/CP/M 2.2, or 8088/MS-DOS, host CPUs/Operating Systems. Frame prices start at £72, system prices at £1495. The systems are called "Odin".

Gemini are now promoting the Costgold 8088 based CPU/256K RAM board, as used in certain versions of the "Odin" systems above. This board can run the MS-DOS Operating System, and price is £350. Rumours too of an 80286 in September?

Gemini have also launched a new version of their disk transfer Multi-Format-BIOS system. (M-F-B 2 remains available.) The new system is renamed DX-3 (for Data eXchange). Physically the system varies in having all of its drives in a single unit - 5.25" 96tpi, 5.25" 48tpi, 8", 3.5" and 3" floppies are supplied as standard, plus a Winchester. The system is now driven from a Wyse-50 terminal.

The DX-3 software is also extensively revised, and more comprehensive. Code conversion and mass duplication facilities are now incorporated as standard, and most commonly used facilities are available from a new menu driven front end, AMIE (All Media Interchange Environment - I think!). With AMIE, support is relatively easily added for different Operating Systems, and various options will become available over the next few months. Complete systems from £7950.

Doctor Dark's Diary - Episode 26

Gempen, alias GM519/520/521

This article, and the last one, come to you courtesy of Gempen, which is a version of Diskpen specially adapted to make use of those features of the Gemini GM812 IVC video board that make it so much better than a Nascom 1 display, for instance. It also works rather well with a GM832 SVC, and I find it ideal for writing these articles.

It isn't WordStar. I'm glad. WordStar is very powerful, and very expensive too. Gempen is sensibly priced, and is easy to learn to use. It does all the text formatting that I need for these articles, namely word wrapping and padding lines out to the right length (right justification), using very simple commands. It doesn't do footnotes, or cope with 32 Megabyte files. Still, you don't want me to write 32 Megabytes, do you? [Ed. - You do, and you're fired.]

The manual is written in the kind of English real people understand, and is fairly short. I was interested to note that not only does it tell you how to extend the program to do fancy printing with a suitable printer, but there is also a very clear explanation of how to get your text back if you accidentally exit from the program without saving. I bet the WordStar manual doesn't do that!

I could list all the commands the program uses, but it would be a waste of space. It does all the obvious things like inputting, deleting, search and replace, read from disk, that you would expect, with commands that almost all use the most obvious single letter of the alphabet. Why do control K Q, when you can do a nice simple Q? If you need a simple effective text editor, buy this one.

Hisoft C - is it all wet?

Sorry about the heading, I tried to resist the temptation, but failed. There seems to be a habit among the authors of C books and manuals to include odd headings and quotes from Lewis Carroll's "The Hunting of the Snark", or poems that mention the sea. These people have corrupted me. It can't have been the BASIC, Mr Dijkstra, I haven't touched any in ages...

Anyway, Hisoft C++, it says here, is an extremely fast C compiler for CP/M-80 machines, and is also available for those machines in plastic cases imported by that nice Mr Sugar. I had no idea that people who bought those were likely to want to write programs on them, let alone in C, but let it pass. C++ is supplied complete with the excellent program editor that I mentioned in the last episode, so that it is easy to create and edit your source code, with no risk of getting it word-wrapped by accident. (An aside on editors. A friend of mine has told me about the latest trend, the folding editor. (Don't panic, Paul! I'm talking about text editors.) One of these is supplied by Inmos, with their Occam development system, apparently. It seems the "paper" in this editor can be "folded" so that you can have several parts of the file on screen at once. Much better than having to scroll up and down to refer to other parts of the code, I would imagine. And not as ugly as all these awful windowing systems, either, even if they are able to do the same trick, which I doubt. Hurry up and write one, somebody. Preferably Hisoft.)

The compiler runs fast, as is usual. I found using a certain other firm's compiler on the office Sirius was almost an overnight batch job, but this one doesn't mess about. On the other hand, the poor Sirius does have to spend an awful lot of its time changing the speed of its disk drives. Random access files are catered for by C++, and the manual describes itself as "large and comprehensive", with "numerous easy worked examples of C and a complete guide to the language." This, I think, is optimistic. The manual is excellent, but trying to learn a language from the manual is not the best way of doing it. Does anyone remember the Nascom BASIC manual? So I read some books...

"C Language" by Friedman Wagner-Dobler, published by Pitman. £3-95. Don't buy this, I think it is awful. Look at pages 25 and 26 in Boots, if you don't believe me. [Ed. - can't I just wear sneakers?]

"Practical C" by Mark Harrison, published by Sigma Press. £7-95. Much better, especially as it uses Hisoft C in the examples; shame it had to be the Amstrad version, but nobody is perfect. Don't be put off by the blurb telling you to "avoid C sickness".

"Understanding C" by Bruce H Hunter, published by Sybex. I seem to have peeled the price tag off this. It was expensive, but is good.

"The C Compendium" by David Lawrence and Mark England, published by Sunshine. £12-95. This gives a lot of examples of routines, building up into a useful library. I forgive the authors for saying Quicksort is "probably" the quickest in-memory sort. I thought Knuth said it was definitely the quickest.

Enough name dropping. Having read all those, I thought I ought to be able to program in C. I was right. The input and output the language uses are like those in Unix, which is right and proper, and unpleasant to use as well. As languages go, C is very concise, which is a good thing sometimes. It can be a bad thing as well, and I found myself frequently baffled by code written to impress, when I looked at some more C programs recently. The problem is that many programmers like to use slick tricks, which are difficult to read. Anyone lumbered with the task of maintaining such a program will know what I mean.

Hisoft C follows the standards laid down by Kernighan and Ritchie, who apparently are major deities in the Unix/C view of the universe. You can write i++ or ++i instead of i = i+1, which is fairly comprehensible. And you may be easily able to appreciate that there is a difference between comparatively simple statements like i=(j++)+(k++) and i=(++j)+(++k). But sooner or later, you will fail to understand what a bit of code does, and alter it disastrously. (The first assignment increments j and k after it has set i to their sum, so i is j+k, whereas the second assignment makes i hold j+k+2.)

The language is certainly very powerful, but it does not protect the programmer from making mistakes. Where Pascal will notice you are trying to access an array element that does not exist, C will cheerfully amend the operating system for you. The resulting crash is usually a real mess. Anything you write that fulfills the rules of C syntax will compile...

Still, Hisoft were only following the rules there, and none of that is their fault. The real catch, with this compiler, is that there are no "reals". Now this is a big disappointment, and not just because it means we are unable to run that damnable benchmark program. Floats are an important part of much of computer programming, and this compiler would have been a really great one if it had had them. It hasn't, and that is my only genuine (nearly said "real") criticism of it. I hope Hisoft are going to produce a further version, with floats, and I would suggest that they do it soon, and include support for the Belectra HSA-88B as well. Otherwise, everyone is going to buy Mix-C, and I for one won't blame them.

Hisoft Pascal 80

Yes, another Hisoft language review. The latest version of Hisoft's Pascal is now available for a very sensible price. And if you already have an earlier version, you don't have to pay out all over again: you just get an upgrade for £10 by sending them the old disk back. This way of doing business is the only acceptable way, if one is going to bring out upgraded versions of things like compilers. Unfortunately we will never be able to convince the hardware manufacturers of the wisdom of taking their old machines back, and selling the new ones for only a £10 upgrade fee. What a pity!

What you get, whichever way you pay for Pascal 80, is a disk and a manual in a ring binder. The front of the manual consists of lots of update pages, which it is up to you to put in the right place, and remove the old ones. I managed it,

and didn't have to curse very many times, but I have to admit I was surprised at having to do this. Then I thought how much more money Hisoft might have wanted from me if they had had to do the job, and I decided it was a good way to update a manual...

Modern Hisoft manuals are good, and full of useful advice for first time users, without having too much of it. (The old ones, as I remember, were rather hard work to use and to read; we all have to start somewhere.) This manual contains a section describing the improvements over HP4, which corresponds very closely with what I wrote ages ago about the parts of Pascal that HP4 couldn't do, although I don't suppose for a moment that I had anything to do with this. For instance, Variant records are now possible, as are files of type other than text. I have not yet needed to use a variant, but I used to hate having to save integers, reals and booleans as text.

You still can't have files of files, but who wants them? (Cue for at least twenty angry letters from academics.) Another new feature is that the compiler can now cope with much bigger programs, because it saves each routine to disk as it is compiled. The symbol table is twice as big, too, not that I ever managed to fill the old one. Reserved words can now be in lower case, if you like. Apple Pascal always used to look nice because of that, but it isn't vital, just a thoughtful touch.

The manual gives detailed descriptions of the syntax and semantics expected by the compiler, and describes the various predefined identifiers that are available to the programmer. Once again, I will save a lot of space by not copying out a great list of these, as they are almost all exactly the same as any other Pascal. Instead I'll attempt to restrict myself to writing about the interesting, or changed, parts.

A major new feature of the system supplied is the appearance of a menu at the start of operations. This offers a choice of editing a program, compiling it, running it, executing it (which means compile AND then run) or quitting. It is rather like the options offered by p-system Pascals. The editor is the now standard Hisoft program editor that I have described before, and can of course be modified for the SVC as mentioned in the last article. This time, however, when you exit from editing, the menu reappears, and you can compile the program from there with a single keypress. This really does save a lot of time, compared with the long commands one had to enter to use earlier versions of the compiler, especially if one is being naughty and programming and debugging at the keyboard. Slapped wrist from Niklaus Wirth if he catches you at it. If by some mysterious chance your program contains a syntax error, and the compiler spots it, you no longer get a mystery number on screen and have to dive for the manual. The error messages are in English. Yet another major advance, not that I ever see error messages, of course...

The New, Mark Dispose and Release procedures have been changed so that they now conform to Wirth's standard, and this allows more sophisticated use of dynamic variables, without which fancy list handling is out of the question. You can do astounding things with lists, dear BASIC users. The Inline and User procedures allow you to put in machine code, if necessary. This is, of course, disgusting to all who regard programming as an abstract art form. Have you noticed how slow their programs are? Peek, Poke, Inp and Out give you access directly to memory and Z80 ports, and are consequently vital if you want to write programs that work, say, a Pluto, or indeed, any of the other 80-BUS boards that expect to be accessed via Z80 ports.

The most tremendously useful of all the new procedures, however, is Chain. It loads and runs another program, so now it is possible to write vast systems of programs, way beyond the size of memory. And they don't have to communicate with each other in the clumsy way I once advocated, by passing data in files, either. As long as the variables used in more than one program are global variables, and are declared in the same order in each program, they will stay in memory ready for the next program to use them. The best way to do this, says the manual, is to store all these declarations in a file, and use the compiler's "include" option to read in this file as each program is compiled. This

procedure is excellent, even if it does make my old chaining method obsolete. It has the added advantage over my offering that it works on any disk drive, not just drive A. The speed at which programs stored on a MAP 80 ramdisk chain one another is truly inspiring for all those who are intending to write large systems of programs. Many people wonder when I will actually get round to writing a large system of programs, but I say to them, patience. After all, the OU keeps me rather busy at this time of year, and I have not yet finished my Prestel program...

All the programs I have written with this compiler have done what they were supposed to, except when the compiler didn't realise what I meant, and did what I had said. That sort of machine intelligence is a few weeks away, at present. I have had a slight difficulty with my version of CP/M not behaving properly when the menu system is running, but it works with older CP/Ms, so it isn't the Pascal causing the problem. I hope to sort this one out soon, and if there is any useful advice to share, I will pass it on. The compiler gets my recommendation. There are more expensive Pascals around, but I don't know of any that can be definitely said to be better. Go on, somebody, tell me I'm wrong...

SVC Graphpac - alias GM575

I have been generously supplied with a copy of Gem-Graphpac to try out on my Gemini GM832 SVC video board. It always surprises me how few software authors want their products to be reviewed. I wrote to a supplier of Pluto software, suggesting that they might like me to write about their product, but they never answered my letter. I can only assume that they don't like my style, or that there may be something horribly wrong with their product and they don't want me warning you.

Meanwhile, enough sidetracking and back to the subject in hand. Graphpac works with Microsoft's MBASIC interpreter and an SVC, to allow easy control of the SVC's extensive graphic ability from programs written in (takes stiff drink) BASIC. Actually, it reminds me of something I once wrote to go with Nascom BASIC, called Vortex, except that Vortex was rather clumsy by comparison. Anyone still use Vortex?

The software automatically links itself into MBASIC for you, and has the effect of adding a large number of new commands to the existing BASIC ones. Graphpac is able to set, reset, invert or test any pixel on the screen; it is also able to draw lines using either rectangular or polar coordinates. There are two resolutions, 256 x 256 or 160 x 75 using block graphics. In either case the top left of the screen has coordinates 0,0. Circle and arc drawing are also included, and can use either a Graphpac routine or the SVC's own faster routines. Various routines for screen handling are included, such as SCROLL followed by a number, which sets up the SVC so that only the specified number of lines at the bottom of the screen will scroll.

Graphs, in the form of vertical bar charts, are specially provided for, using either solid bars or half tone. Although the results are not as sophisticated as graphs drawn on QLs (a good machine, even if it did have a plastic case), they are a great deal better than no graph at all. The ones in the demo program on the disk were quite effective, and the firm was obviously doing very well. Odd, isn't it, the way these profit graphs always go uphill? Once you have created a work of art on screen, you won't want to lose it forever, so the PSI and GSI commands which put and get a screen image will come in handy. Given the speed of the SVC, it comes as no surprise that these are quick. Another very handy facility enables full use to be made of the large number of SVC commands that are accessed by "printing" escape sequences.

Provided you have an SVC and MBASIC, Graphpac is more or less an essential buy, particularly as it is sensibly priced. I wish Vortex had been as good!

Pluto upgrade saga

Every 80-BUS user who isn't immensely wealthy must have had the Pluto 2 on his wish list for ages now. If I had won on the pools, I would have three Pluto 2's linked together to provide proper colour graphics, with each pixel being any one of a range of over 16 million colours. Regrettably, though, the foolish football teams persistently get their scores wrong, and that remains a wish. All I can hope to do is upgrade the Pluto 1 as far as I can manage.

The first thing I considered was something called "the double speed processor kit" in old price lists, which used to sell for £60, and was supposed to give a 60% increase in the work rate. I assume the other 40% was eaten by memory wait states, or some such phenomenon. This kit seems not to be available any more, and the people at Io Research didn't seem to have heard of it. Still, I had a way to make things go a little faster, by replacing the 8088 processor with an NEC V20, which thankfully has now come down to a sensible price.

I needed a benchmark of some sort, so I rewrote my Union Jack program so it repeated itself ten times, to make timing easier. With the 8088, it took 26 seconds; with the V20, this was reduced to 19 seconds. All very satisfactory, as far as it went, but I still could only produce eight colours. You can actually produce very good effects that look like more colours, just by putting together two different coloured pixels side by side, since they are so small, but it's not the same as having 16 million colours to play with, or even 4096.

In the hope of making savings by assembling something myself, I wrote to Io Research and tried to buy a bare board and parts list for either the palette board or the mini palette board. Io Research are a very unusual firm! Not only do they answer my letters, but when they say they will phone back, they always do. I don't want to go too far over the top and embarrass them, but their standard of service makes a very refreshing change these days. Cynics may think I wrote that because Io managed to find a spare mini palette board for me, in spite of the fact they stopped making them over two years ago, but cynics are wrong on this occasion. I have not yet finished the installation of the improved board, so I am unable to go into raptures about the 4096 colours, as yet, and still am not sure I have made the right links inside my Microvitec monitor. I think I am going to have to put some sort of switch gear in the monitor, as it will soon be used by more than one machine. You will read about it all here, when I get it going.

Local firm in interview shock horror probe!

Amazingly, a quite large computer communications firm actually interviewed me in March. The interviewer seemed interested in the Open University courses I had been studying, and it all seemed to go quite well. They are now two months overdue on their promise to "let me know", and ignored the telex message I sent them about this. I can only assume they are trying to avoid Dr Dark's curse, which results in every firm that turns me down going broke, without having to take me on. The curse is real, folks: Quest Automation and DJAI Systems found out the hard way! (Although somebody later baled Quest out by buying them up.) So, I still have my dull job. Maybe I could be a lumberjack?

Information wanted

The loan of any documentation for the following items would be very welcome indeed: the Gemini real time clock board, the Iotec Iona (heavy), and a huge S100 system with two 8" drives and the words "Sytem III" on the front (very heavy.) I will gladly pay postage both ways for suitable manuals to look at.

Old thingies for sale dead cheap

For the poverty stricken enthusiast, I have a Nascom 1 in poor health. This would make an interesting restoration project for a masochistic enthusiast. The Nascom 1 has about half its chips missing, although all the RAM is there. The keyboard seems to be unbroken. There is a 3 amp Nascom PSU, with no transformer, and some other parts missing. I have a Nascom 1 construction guide

to go with this lot. Offers in the region of not a lot, to the usual address, please. This one is a real challenge!

Keep IBM stuff out, says Doctor Dark

Or I will start writing about the Iotec Iona, as soon as I manage to get it going, and bore you all to bits. Seriously, it is my opinion that anyone wanting to read magazines about Itty Bitty Machines will find dozens of colourful comics in their newsagent's shop. Besides, Lawrence the long haired weirdo and his mate Headcrash might come and sort you out...

Gemini Maintenance Memorandum

Gemini have relatively recently started releasing what they call "Maintenance Memoranda" to their dealers. Reproduced below is one that may be of interest to various Scorpio News readers. If you have any further queries as a result of this, contact a Gemini dealer.

Issue 4. 26/02/87. Ref: Disk drive/Controller card/Software compatibility

Every effort has been made to make all releases of software to be compatible with existing products. The following is a guide to the hardware/software restrictions.

All SIMON or RP/M EPROMs will operate correctly with all FDC cards provided that a floppy only system is used, but because of the difference in side change on the WD2793 FDC chip BIOS 3.4 or greater must be used if a double sided drive and GM849(A) is in the system.

FDC Card	SIMON Version	BIOS Version
GM809	3.1 or later	3.2 or earlier
GM829	3.1 or later	3.2 or later
GM849	3.3 or later	3.4 or later
GM849A	4.2 or later	3.5 or later

Miniscribe M3213 and M3425 Winchester Disk Drives

When replacing a Rodime with a Miniscribe Winchester drive a Miniscribe FORMWIN.COM is required, together with GENSYS.COM Version 1.2 or later. Please note that the reference to a Winchester type in the SYSTEMx.CPG files should be changed from R020x to either M3212 for a 10MB or M3425 for a 20MB.

FORMWIN.COM

Version 0.7 or greater is required for a GM849A controller card and version 1.0 or later if an Adaptec 4000 controller card is used instead of the Xebec.

NEC D5126 20MB Winchester Drive

These drives will be available shortly and are software compatible to the Miniscribe M3425 20MB provided that a Xebec S1410A controller card with Revision F firmware is used. A NEC D5126 FORMWIN.COM must be used to format the Winchester (Version 0.7 or later if Xebec/GM849A combination) or Version 1.0 or later for an Adaptec controller card.

Adaptec ACB-4000A Winchester Controller Card.

Certain 80-BUS systems manufactured after 1st March 1987 may use the Adaptec Controller Card. Please note that this may only be used providing the following are in the system: GM849A, SIMON 4.4 or later, BIOS 3.6 or later and FORMWIN.COM 1.0 or later.

Disk Formats and CP/M Disk routines - Part 3 by M.W.T. Waters

Disk Parameter Headers

The CP/M 2.2 DPH

A DPH for CP/M 2.2 is a table of eight 16 bit words broken up as follows:

```
DPH:    XLT     ; Translate table address
        0000    ; 6 bytes scratchpad for BDOS use
        0000
        0000
        DIRBUF  ; Address of a buffer used
                ; for reading directory information
        DPB     ; Disk Parameter Block Address
        CSV     ; Checksum Vector address
        ALV     ; Allocation Vector address
```

Note that CP/M 1.4 doesn't use DPH's and that the DPH's for CP/M 3 are longer.

The BIOS has a DPH for each drive supported by the system. When a drive is logged in, the BIOS returns the address of the appropriate DPH to the BDOS. The BDOS then uses the DPH to locate any remaining data structures that it requires.

The first entry is the address of a sector translate table which the BIOS will use to convert from logical to physical sector numbers before accessing the disk. If no sector translation is required, this word will contain zero. Sector translation is discussed in some detail elsewhere so all that will be said at this point is that CP/M 2.2 only allows for translation of 128 byte sectors. If sector translation is required on systems using sector sizes of more than 128 bytes, XLT must be set to zero and the BIOS should do the translation separately. CP/M 3 doesn't suffer from this limitation as all disk transfers and sector translation are done in terms of physical disk sectors.

The next 6 bytes are reserved for use by the BDOS as workspace. Their initial value is immaterial.

DIRBUF is the address of a buffer used by the BDOS and BIOS when reading directory information. This buffer is one physical sector long (512 bytes for Gemini DDDS, QDDS and QDSS formats, 128 bytes for SDDS format) and may be shared by all drives in the system. CP/M 1.4 is different in that it uses the default buffer at address 80H for this purpose. CP/M 3 buffers the directory records differently and consequently CP/M 3 DPH's do not contain this field.

DPB is the address of a Disk Parameter Block for the drive concerned. A separate DPB is required for each disk format handled. DPH's for drives using the same disk format may all address the same DPB. The DPB is explained in full later.

The Checksum Vector

CSV is the address of a table that CP/M uses to check whether a disk has been changed. The table is used to store a checksum byte that uniquely identifies a directory record (128 bytes containing 4 directory entries). A good definition of a checksum is, in this case, a single byte which is derived from a block of data and whose value is unique to that block. As far as we are concerned, a block is 4 directory entries totalling 128 bytes (i.e. one CP/M record) and the checksum byte is obtained by adding together all of the bytes in the record, ignoring overflow, to produce a total. This total, and the totals for the remaining directory records are stored in a part of the BIOS known as the checksum vector and its address is given as CSV in the DPH. There is a checksum vector for each drive supported by the system. The length of each vector depends upon the number of directory entries. Since the checksum applies

to a directory record containing four 32 byte directory entries, the total space required for each checksum vector is (DRM+1)/4 where DRM is one less than the total number of directory entries permitted on the disk. In the case of the QDDS format, the figure is 128/4=32 bytes.

Whenever the directory of a disk is accessed, a checksum is produced and compared with the appropriate value in the checksum vector. If the checksum is different, the BDOS knows that the disk has been changed and will set it R/O under CP/M 1.4 or CP/M 2.2 or log in the new disk under CP/M 3.

Clearly, this directory checking is a waste of time for drives, such as a Winchester hard disk, where the disk cannot be changed. This type of drive is said to have fixed media. Digital Research had the foresight, when designing CP/M 2.2 and CP/M 3, to allow for this type of drive. If no directory checking is required, the checksum vector may be omitted with a resultant saving of space in the BIOS.

The Allocation Vector

The last entry in the DPH is the allocation address. This is the address of a table which the BDOS uses to keep track of the data blocks on disk. This table is called the allocation vector and it is a bit map of the blocks on the disk such that each individual bit represents one block. Consequently, the state of eight disk data blocks (ie allocated to a file or not) can be shown in one byte of the allocation vector. Block 0 is represented by bit 7 of the first byte in the table, block 1 by bit 6 and so on for each byte in the table. There is an allocation vector for each disk drive in the system and the address of the vector for the logged in drive may be obtained by calling BDOS function 27.

When a drive is logged in for the first time, the BDOS scans the directory entries on the disk and extracts the numbers of the blocks already allocated to files. For each block used, the BDOS sets the appropriate bit in the allocation vector. Quite simply, when you wish to write to disk, the BDOS searches the allocation vector for a 0 bit. If none are found then the disk is obviously full. If a 0 bit is found then the BDOS may allocate the associated block to the file.

To compute the amount of free space on a disk, it is a simple matter to count the number of zeros in the allocation vector and then multiply by the block size. Programs such as STAT.COM obtain their free space figures in this manner and as you will probably noticed do not access the disk at all when, for example, STAT B: is typed in while currently logged into drive A and assuming that drive B has been accessed since a ^C was last typed.

The BDOS updates the allocation vector during disk write and delete operations so that the amount of free space shown by STAT.COM will always be correct. Having said that, imagine that we are writing to a disk. The BDOS will set bits in the allocation vector according to which blocks are being allocated. If we return to CP/M by executing a RET instruction so that a warm boot is avoided and, at the same time neglect to close the file, STAT.COM will return an incorrect amount of free space as the allocation vector will not agree with the disk directory entries actually written to the disk. This is because the final directory entry for a file being written to disk isn't written to the directory track until the file is closed. The only way to reclaim the lost disk space and obtain a correct free space figure is to force a disk reset by typing Control C.

Under CP/M 1.4 and CP/M 2.2, if a disk is changed without performing a warm boot to reset the disk system, the amount of free space indicated by STAT.COM will be inaccurate as the BDOS will have the allocation vector set up for the previous disk. Under CP/M 3, as soon as the BDOS realizes that a disk has been changed, it logs in the new disk and sets up the allocation vector accordingly.

The length of the vector is calculated to be (Number of blocks)/8 bytes long, rounded up to the nearest whole byte. In the Digital Research manuals, this is quoted as (DSM/8)+1 where DSM is one less than the total number of blocks contained on the disk. Here is another good reason for having larger block sizes in that the bigger the block size, the fewer blocks there will be and the smaller the allocation vector will be (i.e. the BIOS will be smaller and the TPA, consequently, may be larger).

The CP/M 3 DPH

By contrast, a CP/M 3 DPH is given below. Digital Research needed to increase the amount of data provided by the DPH while providing upwards compatibility with CP/M 1.4 and CP/M 2.2. Further, they have come to realise that many systems require to access disks of different formats and that few systems use a 128 byte sector size on their disks.

```
        READ    ; Address of sector read routine
        WRITE   ; Address of sector write routine
        LOGIN   ; Address of disk login routine
        INIT    ; Address of disk initialisation routine
        UNIT    ; FDC relative drive code for this drive (Byte)
        TYPE    ; BIOS scratchpad. Current density etc. (Byte)
XDPH:   XLT     ; Translate table address
        0000    ; 9 bytes scratchpad
        0000
        0000
        0000
        00
        MF      ; Media Flag (Byte)
        DPB     ; Disk Parameter Block address
        CSV     ; Checksum Vector address
        ALV     ; Allocation Vector address
        DIRBCB  ; Directory Buffer Control Block address
        DTABCB  ; Data Buffer Control Block address
        HASH    ; Hash Table address
        HBANK   ; Hash Table Memory Bank (Byte)
```

The major differences between a CP/M 2.2 and CP/M 3 DPH are quite obvious at a glance. The DPB entry is as described in the section concerning the CP/M 2.2 DPH and will not be repeated here. Under CP/M 3, all disk transfers are in terms of physical disk sectors and the sector translation table, at long last, now also applies to physical sectors of whatever size. This means that it is no longer necessary to provide special translation routines in the BIOS for sectors that are larger than 128 bytes. Note also that the BDOS scratchpad area has been increased to 9 bytes and that there is no entry for DIRBUF.

Additionally, the DPH for CP/M 3 has been extended and is now known as the eXtended Disk Parameter Header or XDPH. For compatibility with CP/M 2.2, the BIOS still returns the address of the DPH when a drive is logged in and in fact returns the address of the XLT field of the DPH as with CP/M 2.2.

Before examining the XDPH fields in detail, a word or two is required about the CP/M 3 Drive Table.

The drive table (@dtbl in the CP/M 3 manuals) is a 32 byte area of RAM that is comprised of 16 addresses. Each entry in the table points to the XDPH for its associated drive so that the first entry applies to drive A and the last entry applies to drive P. If no physical drive exists, the table entry is zero but there must be an entry of some sort for each of the 16 drives that may be supported. The BDOS can obtain the address of the drive table by calling BIOS function 22 (ie calling address WBOOT+3FH in the BIOS jump table) but for most purposes, the BIOS is the main user of the table.

When a drive is logged in by calling the SELDSK entry in the BIOS jump table, the BIOS uses the drive code supplied by the BDOS as an index into the drive table and it then extracts the address of the XDPH for the required drive. If this address is zero then no physical drive exists and an appropriate error code is returned to the BDOS. If the drive does exist, the drive is logged in and the drive code is stored away for reference by the disk read and write routines (in variable @cdrv).

Similarly, when the READ or WRITE entry points in the BIOS jump table are called, these routines use the stored code for the current drive to index into the drive table and fetch the address of the appropriate XDPH. Bear in mind that any errors due to there being no physical drive present should have already been found by the SELDSK routine.

Once armed with the address of the correct XDPH, the SELDSK, READ and WRITE routines use it to extract any other information they require. A detailed description of each field of the XDPH is given below.

The INIT field provides the address of an initialisation routine for use with the drive addressed by the XDPH. The drives are initialised by the BIOS during cold boot. Other than this, the INIT routine will not be used except, perhaps, during error recovery.

The READ and WRITE fields of the XDPH are fairly self explanatory. These fields provide the addresses of the sector read and write routines for the selected drive and are extracted from the XDPH by the BIOS when called at the appropriate place in its jump table. Before either of these routines is called, other calls to the BIOS will have performed any necessary sector translation, logged in the appropriate drive, set the track and sector numbers, set the DMA address and indicated which memory bank is to be used in the coming disk transfer. All that is required of the BIOS in its simplest form is to select the required memory bank and transfer data to or from one physical disk sector.

A facility exists within CP/M 3 for the BDOS to instruct the BIOS to transfer multiple sequential sectors to a contiguous area of RAM, starting at the DMA address. The BDOS tells the BIOS how many sectors to read or write and the BIOS will, if this facility has been implemented, continue transferring sectors to or from disk until the required number have been copied. In case this facility has not been implemented by the writer of the BIOS, the BDOS will continue to set track and sector numbers and DMA address and will call the BIOS the appropriate number of times. It is up to the BIOS to ignore the surplus calls if it has already transferred the required amount of data.

The LOGIN field is used by SELDSK to find the address of the routine appropriate to the disk drive being selected.

The UNIT field is a single byte that holds the FDC relative drive code, i.e. the actual value that is output to the disk controller to select the required drive for use. The BIOS extracts this code when any of the SELDSK, READ or WRITE routines is called.

The TYPE entry of the XDPH is a scratchpad location for use by the BIOS. It may be used for any purpose but Digital Research recommend that it is used to hold drive relevant information such as current density in systems that support single and double density.

The CSV entry under CP/M 3 is identical to that of CP/M 2.2 but the checksums contained in the vector are calculated in a slightly different manner. Instead of adding together all 128 bytes in a directory record, CP/M 3 produces sub-totals for each of the four 32 byte directory entries in the record. These four sub-totals are then exclusive ORed together to produce the checksum byte. This XORing together reduces the number of bits lost due to overflow (due to adding as for CP/M 2.2) and therefore each checksum byte contains more information about the directory record. For this reason, the likelihood of identical checksums being returned for directory records on different disks is reduced and hence the possibility of a disk change being missed is also reduced.

ALV is similar to that for CP/M 2.2 in that the BDOS uses the allocation vector to identify which data blocks have been allocated to files and which are free for use. CP/M 3, however, uses two bits in the allocation vector for each data block on the disk. The only exception to this is in a non-banked CP/M 3 system where the option is given to use single bit entries as with CP/M 2.2. Since there should be no non-banked CP/M 3 systems sold commercially (the non-banked version of CP/M 3 is merely an aid in the generation of a banked system), all users of CP/M 3 can assume that their system uses double bit entries in the allocation vector.

Obviously, a double bit allocation vector is going to occupy twice as much memory as a single bit allocation vector so why then use two bits per entry? Remember how STAT.COM calculates free disk space under CP/M 2.2 and how it is possible to fool it by forgetting to close a file. The only way to update the allocation vector under CP/M 2.2 is to hit ^C to reset the disk system and log in the drives again. This of course takes time and CP/M 3 boasts better disk performance. Digital Research have gone to great lengths to avoid resetting the disk system with each warm boot - particularly since CP/M 3 has also spent time buffering both directory and data blocks to reduce disk accesses still further. Control C is still available under CP/M 3, when running the CCP, to allow the user to force a disk reset but, to improve performance, this should be done as little as possible.

To eliminate the need for disk resets on warm boot, Digital Research came up with the double bit allocation vector. If, as before, a file is written to disk, the BDOS will keep track of which data blocks it is allocating by using one of the two bits in the allocation vector for each of those blocks. When a file is closed, the second set of bits for the affected blocks are updated to match the first. If, on the other hand, the file is not closed before returning to CP/M, the first set of bits can be reset to their original state because the second set provide a record of which blocks remain free. Consequently, the disk free space shown by SHOW.COM (CP/M 3's equivalent to STAT.COM) is always correct and the need for a disk access has been alleviated. Unlike CP/M 2.2, the size of the allocation vector for CP/M 3 is calculated as (DSM/4)+2. The "+2" gives us the clue that the double bit allocation vector is in fact configured as two single bit allocation vectors joined end to end.

MF is referred to as the media flag. This byte is set to zero by the BDOS when a drive is logged in. If the disk/computer hardware supports "door open" interrupts, the BIOS can set this byte to 0FFH when a drive door is opened. If this is the case, before the BDOS next performs a file operation on the affected drive, it will check for a disk change and perform a login if necessary.

NOTE. The Media Flag is used in conjunction with the variable @MEDIA contained in the CP/M 3 System Control Block and will normally only be implemented on systems supporting a "door open" interrupt.

DIRBCB is the address of the directory buffer control block on non-banked CP/M 3 systems. On banked systems, DIRBCB points to the head of a list of buffer control blocks (BCB's). The list head is a 16 bit address which points to the first BCB in the list. The first and subsequent BCB's contain a 16 bit address field which points to the next BCB in the list while the last BCB contains a zero value in this field. This type of list is, for obvious reasons, known as a linked list. A comprehensive description of the BCB format is given later so, all I shall say at this point is that each BCB contains the address of a directory buffer which will be one physical sector in length (sound familiar to CP/M 2.2 users?). The maximum number of directory buffers required will be (DRM+1)*32/Physical sector size. For the Gemini QDDS format, this figure will be 8 per drive. Different drives may share directory buffers under CP/M 3 and consequently one or more drive XDPH's may refer to the same BCB list.

DTABCB is the address of the data buffer control block on non-banked CP/M 3 systems or a list head on banked systems. The BCB format is identical to that for directory BCB's except that each data BCB points to a data buffer which will be one physical sector in length. CP/M 3 uses the physical record buffers for blocking and deblocking the physical sector into 128 byte CP/M records and

consequently doesn't require buffers where the physical sector size is 128 bytes (Blocking and Deblocking are discussed later). When used, CP/M 3 discards these buffers and overwrites the data in them on a least recently used (LRU) basis so that when access is required to a recently used sector, CP/M 3 reads the data from memory rather than from the disk giving an appropriate increase in performance. CP/M 3 doesn't use the buffers at all if it knows that one or more complete physical sectors are to be read and hence deblocking of the data will not be required. In this case, the BIOS is instructed to read the disk data directly to the TPA at the DMA address.

CP/M 3 directory hashing

The HASH parameter contains the address of a table used for directory hashing. Hashing is a term used to refer to the process of converting a string of characters into a single number that is unique to that string. There are a vast number of methods of doing this, all varying in complexity. The problems encountered with hashing are guaranteeing that the number produced is in fact unique to the particular string involved (and no other) while keeping the number small enough to be manageable.

The problem of uniqueness can be explained with the use of an example. A simple way of converting a character string into a number is to add the ASCII values of the characters together. Taking combinations of the characters A-D for simplicity, there are 16 permutations of those characters that will give the same result. If we then permit characters to repeat, the number of combinations giving identical results increases still further. A few examples are given below, values are in decimal:

```
"ABCD" = 65 + 66 + 67 + 68 = 266
"BDAC" = 66 + 68 + 65 + 67 = 266
"AADD" = 65 + 65 + 68 + 68 = 266
"ACCC" = 65 + 67 + 67 + 67 = 266
```

One solution to this problem is to weight the character values differently depending upon which position they occupy in the string. If we were to assign values to the characters such that A=1, B=2 etc., and multiply the character values by multiples of 27 depending upon their position, the same strings as we used above would give different results as shown below:

```
"ABCD" = 1 + (2 * 27) + (3 * 54) + (4 * 81) = 541
"BDAC" = 2 + (4 * 27) + (1 * 54) + (3 * 81) = 407
"AADD" = 1 + (1 * 27) + (4 * 54) + (4 * 81) = 568
"ACCC" = 1 + (3 * 27) + (3 * 54) + (3 * 81) = 487
```

This example is fine as far as it goes but we want to hash a directory entry of 11 characters. Using the method described above, the lowest number produced will be 1486 for a file "AAAAAAAA.AAA" and the highest number will be 28636 for file "ZZZZZZZZ.ZZZ". However, filenames can also contain figures, spaces and some other characters in addition to upper case letters. CP/M 3 needs to include the user area byte and both extent bytes (EXT and S2) so that it can differentiate between files with the same name in different user areas and also between different extents of the same file. All of this serves to increase the range of numbers generated as a result of hashing.

In CP/M 3, Digital Research hash the file name, user area and extent bytes into a four byte number which constitutes the hash table entry for the directory entry. The user area number is stored as the 4 least significant bits (lsbs) of the first byte. The file name is reduced to an 18 bit number of which the lower 16 bits are stored in the two middle bytes of the hash table entry and the 2 most significant bits (msbs) are stored in the 2 msbs of the first byte. The extent and S2 bytes are combined to form a directory entry number for the file such that the first entry number is 0 and directory entry number 'n' is numbered n-1. This entry number is truncated to form a 9 bit value and is stored so that the msb is held as the 6th bit (bit 5) of the first byte of the hash table entry and the lower 8 bits are stored in the last byte. The method of hashing used on

the file name is fairly complex and involves adding the characters into an 18 bit partial result. Characters whose ASCII values are odd numbers are simply added in while even ones are right shifted first. The position of the character in the file name determines whether the partial result is left shifted none, one or two places after the addition.

The directory entry number for the file is calculated by dividing the absolute extent number by EXM+1 where EXM is the extent mask as defined in the disk parameter block. The absolute extent number is formed by joining the S2 and EXT bytes end to end to form an 11 bit number. (Under CP/M 3, there are 6 bits for the S2 byte and 5 bits for the EXT byte.) This number represents the highest extent number addressed by the current directory entry and falls in the range 0 to 2048 (2048 * 16K = 32Mbytes).

Considering the Gemini QDDS format for a moment, we know that each directory entry can control 4 extents. If we take the absolute extent number and divide this by 4, we will have calculated the directory entry number for the current entry.

All this is fine but why hash the directory entries of a disk at all? CP/M 3 maintains a table of hashed directory entries for each drive in the system. Each table is (DRM+1)*4 bytes long allowing four bytes per directory entry as previously stated. When CP/M 3 needs to access a directory entry for a file, it is a relatively simple matter to hash the file control block and search the hash table for a matching entry. Once this entry is found, its position in the hash table tells CP/M 3 precisely which physical disk sector contains the entry and enables it to directly access the wanted sector rather than having to sequentially read the disk directory from the start. Of course, if the wanted physical sector is already in a physical record buffer, this makes the process faster still.

HBANK refers to the memory bank containing the hash table for the associated drive.

That's all again for this issue. The concluding part of this article will be in the next issue of Scorpio News.

Dealer Profile - Off Records

[Ed. - In our continuing series looking at the roots of various 80-BUS dealers, this issue it is the turn of Battersea based OFF Records.]

By the year 1975 computing on mainframes had become quite tedious. A single machine had to support so many users that it was impossible to do the more interesting experiments on them and computing centres had become bureaucratic empires which stifled all work other than the trivial and routine. People who have only worked on personal computers find it difficult to visualize the straight jacket imposed by a large time-shared machine. They are fine if you happen to be a physicist running your application programs, but if you are actually interested in the machine itself, the lack of access to the innards of the machine becomes quite frustrating.

The mathematics department at South Bank Polytechnic was fortunate in having acquired three years earlier the first GT11 in the country, an early version of a PDP11 with graphics capability. For most of the time this machine could be used as a personal computer, and a very nice computer it was too! Gradually, however, more and more basic teaching had to be done on it and inevitably the machine became time-shared losing all of its early advantages. The powers-that-be also had the curious notion that, once you had bought a computer, there was no need to maintain or enhance it to cope with changing requirements. Even such a simple device as a floppy disk drive could not be added to enable students to look after their own data.

The Pet and TRS (remember?) had just come on the market and Jaap Creutzberg, then in charge of the GT11 which he had secured for the department, put in a request for these micros to off-load the GT11 and make it available again for the research for which it was originally intended. The proposal was not accepted and it became clear that he would have to find his own computing equipment if he wanted to continue his work on computers. At the time (1978) this seemed entirely reasonable because the Z8000 and the 68000 had just been announced and it seemed that these engines would fit the bill quite neatly. Nobody could foresee that the former would come to grief and that it would be another ten years before the latter would be anywhere near general acceptance. No matter, one could make do with the less than ideal Z80 in the meantime. An entirely different matter was, of course, where to get the money to purchase the equipment required.

The solution was a painful one but it had to be taken. Like anyone else who was conscious in the sixties Jaap Creutzberg had a large record collection, perhaps a bit larger than most. He struck up a partnership with a kindred spirit, Nigel Scott, who, in July 1979, pooled his equally large record collection and proceeded to sell them off in the street markets of Southern England under the banner "OFF Records". A Nascom was soon acquired from the Barnet Mafia and soldered together. It worked a treat! A rudimentary accounts package was quickly knocked up and after a year the market traders acquired a tiny shop in Clapham and had saved up enough to start trading in computer equipment: printers, monitors and the Acorn Atom. They had missed the early Nascom boom but even though that company was now in the hands of the receiver they decided to establish contact. In a memorable meeting at the Clapham shop squeezed between the piles of records and posters of Jimmy Hendrix they were signed up by one Ken Jones as official Nascom dealers. They had arrived!

Soon afterwards Lucas took on the Nascom but it became rapidly clear that OFF Records had to be able to sell Gemini product to keep the Nascom viable. Gemini welcomed the partnership as dealers and thus OFF Records became a member of that exclusive club, MicroValue, until then known as the Gang of Six. Despite the name "OFF Records" neither the Gang nor Gemini knew that they had a music business in their midst: witness one of the joint ads which began "We are computer experts. We don't sell cameras and we don't sell records." The customers didn't seem to mind! They responded by buying vast quantities of computers, sufficient to persuade the bank to lend enough money to buy a three storey building in Battersea, the present location of OFF Records.

Times were good, but 1984 saw a sudden decline in computer trade and the business was kept alive by drastic cuts in overheads, by concentrating on consultancy and, of course, by the record business. Thankfully, the crisis was soon overcome, so much so, that a new office could be opened in Streatham to deal mainly with the consultancy side which showed rapid growth. The Battersea site still houses the workshops but the shop itself has reverted to a record shop which has been rebuilt to provide twice its former retail area.

OFF Records continues to support and market 80-BUS equipment very successfully and agrees with Gemini's MD that there appears to be several years' life left in the old Z80 despite the onslaught of the clone market, certainly enough life to form a bridge to the inevitable dominance of 68000 based systems. Fortunately, projects involving the Challenger have turned out to be so large that relatively few sales generate sufficient income to support this superb system. OFF Records are enthusiastic fans of both BOS software and the Mirage operating system. Especially the latter has opened substantial prospects of vertical market applications and its implementation on Atari has been a convincing argument to accept an Atari dealership. Academic institutions can't make a better choice than a Challenger under Mirage with (detachable) Atari terminals. It has taken ten years but it was worth waiting for!

Like all dealers, OFF Records is the beneficiary of the close relationship fostered by Gemini. Such a tightly knit family of dealers is unique in the industry and to the direct benefit of all users of Gemini equipment.

Letters to the Editor

Re. Shugart SA200s.

Dear Sir,

Mr Smeester asks about Shugart SA200 drives in Issue 2 of Scorpio News. Richard Beal says that he thinks they were 8". I have an SA200. It is in fact a single sided double density 5.25" drive.

I have always understood that it was the update of the originally very popular SA400. Unfortunately Shugart took too long to bring it out and when it arrived double sided double densities abounded.

Yours sincerely, Rodney Hannis, Reading, Berks.

IBM OK ! ... and NasNet ?

Dear Sir,

A few thoughts on the future of Scorpio News, as you ask in your Editorial. But first of all, what a pleasure to receive an 80-BUS related magazine on time!

I teach computing at an independent school in Surrey; for my own purposes, I still use my faithful Nascom-2, with both Nas-Dos and CP/M (I'm using Spex to write this). Our main computer lab has a network of Nascoms using Nas-Net, but I am finding the lack of a structured BASIC, and the paucity of languages/applications a handicap, so we have recently obtained an Amstrad PC1512, partly for immediate use for demonstrations, and partly for the possibility of changing over one day. We also have another lab-full of Amstrad 6128s, and various other computers used for Admin purposes.

Although I should like to see something in the magazine on Nascoms with Nas-Dos and/or Nas-net, I recognise that they are a dying breed, so would not attract any great reader interest. However, I should most certainly welcome the idea of an IBM/clone/MS-DOS section. After the Nascom manuals, I find Amstrad very uninformative (though at least they have now found somebody who can prepare a sensible index: in this respect their PC1512 manual is an enormous improvement on their PCW one). I should also be interested in Dave Hunt's suggestion of a section on Modula-2, having just bought the Turbo version.

I certainly hope you continue into the next Volume. I should be very sorry not to have a magazine to carry on the INMC tradition.

Yours faithfully, B.M. Hawkes, Budleigh Salterton, Devon.

Yes to more OSs.

Dear Sir,

I think that Scorpio News should give space to articles on CP/M-86, MS-DOS and PC-DOS and the hardware, as a lot of readers are now using these systems and there is an acute lack of information about any of them. I myself bought an Apricot Portable, but couldn't get on with it. This was mainly due to the awful screen, but also I couldn't get on with COMMAND.COM and I couldn't do anything with it! Now I have the Gemini GM888 board and CP/M-86 which I am slowly pulling apart from CP/M-80 thanks to Digital Research providing CP/M-80 versions of ASM86 and GENCMD.

May I wish Scorpio News many years of success.

Yours sincerely, W.H. Turner, New Malden, Surrey.

IBM ? Only if you have to.

Dear Sir,

You ask for comment about the future content of Scorpio News.

My interest is solely Nascom/Gemini/Map. I have no intention of deserting the ship to buy an IBM contemptible, as the CP/M Users' Group calls them.

I would gladly pay a higher sub to keep the firm afloat, but I realise that others may be less fortunate financially than I.

It is essential that the sole remaining 80-BUS magazine survives. If lowering the standards to include articles on IBM's black box and its relatives will do that then I will come along reluctantly. But if you do so I trust that the articles will really get down to brass tacks and not just review programs.

All of the IBM users I know seem perfectly happy when all goes well. But when things go awry none of them, and that includes the people who market the beast, seem to have the faintest idea what makes the hardware or the software tick.

Yours sincerely, Rodney Hannis, Reading, Berks.

Taking Stock.

Dear Sir

The recent comments in Scorpio News on the subject of content has caused me to do some thinking about my own situation. Just 8 short years ago my job changed from heavy current to electronics. My electronics training had been obtained in the days of valves. The magazines were full of articles on computing, and I thought I had better learn about it too. I bought a NASCOM 1, which worked on switch on, and I was hooked. On the basis of 'todays supplier might be bankrupt tomorrow' (especially after the NASCOM saga), I bought anything new that I could afford, and acquired a NASCOM 2, disks - a massive 140K, DISKPEN etc; and eagerly awaited the '80 Col x 25 Row' screen that a certain DRH told me was coming soon.

Over the years I have continued to expand the system as costs allowed and I have learnt a lot about hardware and software. A lot of the learning was by sheer persistence, and I have tried to help others by sharing my experiences. Now it is 1987 and times are changing. Gemini promise to keep 80-BUS products available for several years yet, and we, the users have every reason to be grateful that they are still in business where so many have failed. The main reason is probably that they have not pursued the popular market, but aimed at specialist applications. This has meant that the products are not cheap, but all comments support the view that they are good, and well supported with data (except BIOS Source Listings!!).

For my part, I have, like several other correspondents, a good knowledge of the hardware and software of the system, and I am able to adapt it in many ways that would be impossible with many modern systems. My Catalog has a large number of programs, many of which I have not yet investigated. I feel that I have only scratched the surface of the potential of the system and I want to do so many things. I have been promising myself to really learn PASCAL, for example, but I seem to spend so much time word processing, modifying the operating system and so on that this, like so many other tasks, just never seems to get nearer.

In a business situation, things are different. The larger TPA and additional speed and facilities of newer machines have obvious advantages. There are a large number of PC's in the organisation where I work, and I will have to start using one 'for compatibility'. I will continue to use my Gemini system at home (and work) for the foreseeable future, but I will, like many, have some interest in the PC. If Gemini do bring out a PC compatible card, I may buy one even at the price of a clone, since it would probably be much better supported, and

perform better, and would I hope allow continued use of my current well understood hardware and software. I hope they hurry up though, before the market vanishes.

So, as long as this magazine continues to support the 80-BUS systems to at least 50% in content, I would go along with the idea of supporting PC's and Clones, on the basis of 'Half a loaf', and 'If you can't beat them'.

Yours faithfully, Clive Bowden, Truro, Cornwall.

Yes to IBM's, plus more on benchmarks

Dear Sir

You invited opinion on the inclusion of IBM-PC related articles in Scorpio News. After reading Vol 1 Issue 2, I feel my situation is fairly common. I bought a Nascom 2 kit in 1978 and built it up to a CP/M system. I work with IBM PC's a lot, however, and was almost able to use the Nascom to help in my work, but not quite. Modem transfer is messy and slow compared to carrying a floppy disk home and back and even with the ability to read and write PC disks on the Nascom, I couldn't use a lot of the PC software as it needs a true PC compatible machine to run (I didn't think a GM888 would help, although I have seen so little about it). I ended up buying a PC AT clone. I do still use the Nascom in areas that I see no merit in upgrading to the PC (I have not yet tried Z80MU, the MS-DOS CP/M-80 emulation program), but the PC has become my mainstream machine.

Other than helping with work, my interests in the PC are just as in the Nascom. I want to know that makes it tick and how I can get it to do what I want. The articles about how and why the system worked (written by people who were genuinely interested) were, for me, a great strength of 80-BUS News. I have not found an equivalent for the PC.

I feel, therefore, that such articles should be included, PC's being the machines the readership seems to be migrating to. As a pure 80-BUS magazine I feel Scorpio News will not survive long term becoming, little by little, less relevant to the readers' activities. Similarly, it should not be a pure IBM PC magazine. I think a combination would best reflect the way the readership appears to be going.

Continuing the benchmark theme of previous issues, I was interested in the relative speeds of the different PC's (and where the Nascom fitted in) and have some results from a simple test program that may be of interest. The program was written in TURBO Pascal (the only compiler I had for both CP/M and PC-DOS) and was designed to take exactly 60 seconds on a 8 MHz 8088.

```
Program Minute;

Var
  Seconds:          Integer;

Procedure OneSec;
Var
  I:                Integer;
  Number:           Real;

Begin
  Number := 0.0;
  For I := 1 to 1815 do
  Begin
    Number := Number+1;
    Number := Number+1;
  End;
End {OneSec};
```

```
Begin
  ClrScr;
  Write('Timings are for an 8 MHz 8088.  ENTER to start - ');
  ReadLn;

For Seconds := 1 to 60 do
Begin
  OneSec;
  GotoXY(1,12);
    Write(Seconds:2,' seconds used so far');
  End;
End.
```

The timings obtained were:-

```
Z80       =  135 Secs at 4 MHz
8088      =  100 Secs at 4.77 MHz
              60 Secs at 8 MHz
8086      =   46 Secs at 8 MHz
80286     =   36 Secs at 6 MHz
              26 Secs at 8 MHz
              18 Secs at 12 MHz
```

These suggest that the effects of a boost in the clock rate are in proportion to the speed boost, an 8088 is 13% more 'powerful' than a Z80, an 8086 is 30% more 'powerful' than an 8088, and an 80286 is 77% more 'powerful' than an 8086 (running 8086 code).

Using these figures, the DOS-Plus (8 MHz 8086) and CP/M-86 (8088) results shown in Vol 1 Issue 2 page 31 give GM888 clock speeds as follows:-

```
ProFortran  -  30 Secs x 8 (1 MHz 8086) x 1.3 (1 MHz 8088) / 54
               (8088 result) = a clock speed of 5.78 MHz
Basic86     -  73 * 8 * 1.3 / 130 = a clock speed of 5.84 MHz
```

The average of the above is 5.81 MHz, very close to the claimed speed and suggesting that the figures above are pretty accurate.

An area I am not sure of the effect of is adding a wait state. Systems I have used which add a wait state when the faster clock speed is used seem to give pretty well the expected results from the clock speed boost alone, suggesting the addition of a wait state matters little. Is this correct?

Yours sincerely, A D Bowler, North Cheam, Surrey.

Coming Out.

Dear Sir,

I have a terrible confession to make: I have purchased an IBM clone. After nearly 10 years being faithful to the Z80 chip I have committed the unthinkable and I am in the process of transferring my allegiance to the IBM standard from faithful CP/M-80. I have owned an Apricot for business use for three years and although it is an excellent machine it did not have the available open architecture that 80-BUS had supported, or the range of public domain software under CP/M-80. With the Apricot having "come out" I decided that it was time to see what was available on the hardware and software front to tempt me.

IBM have done a good job in fixing a standard to which most people adhere eventually. They have brought order to chaos in the past with standard disk formats, and with the PC we have a rudimentary framework to actually use a sophisticated program with graphics and windows that is unknown in the CP/M environment. It gives a new meaning to the phrase "user friendly" with added complexity for all programs and the perceived need to take a Ph.D. to drive the latest commercial offerings.

Since I have only just acquired my clone I have not collected the full range of programs I will need. The public domain programs are of a high standard and there is sure to be one which will suit my needs for the next few years.

For those wanting to explore the 'C' environment there is MINIX, which whilst not in the public domain, is a rewrite of the popular Unix with all the source code being published in a book. Primarily it is designed as a teaching aid so students can alter code and see the effects of so doing.

In short one is entering a rich world where one can do one's own thing. I feel that your excellent journal should include this aspect of computing as well as the Gemini environment. Indeed Gemini have joined the MSDOS world with their adoption of the Costgold card for their range. All that is needed is to provide an adaptor for 80-BUS to have the full range of colour cards on our traditional systems. Scorpio News' contribution could be to establish the database of a selection of public domain software for its readers.

Finally many congratulations on your efforts in launching Scorpio News.

Yours sincerely, John Stuckey, Spartacode Ltd., Bognor Regis, W. Sussex.

Various Points.

Dear Editor

Many thanks for an excellent second issue of Scorpio News. There were many items of interest and I feel obliged to put pen to paper and comment upon some of them.

Dr Dark mentions that he would like to obtain a board which existed to run a Nascom 2 keyboard on a Gemini IVC/SVC. If such a board exists I too would be most interested to obtain one.

He also talks about the ED80 editor from Hisoft. I have used this on occasion and found it unable to read/write files to drive P:, this being the drive MAP 80 assign for the RAM drive. A short investigation reveals that ED80 believes drives end at O:. This can be remedied by altering with DDT etc, one byte:

```
0C3FH is:     FEh, 10h
alter to:     FEh, 11h.
```

If your version does not have FE 10 at this location do not make the changes.

Dave Hunt enquires of us humble subscribers as to whether articles about the IBM and clones would be of interest. I for one would welcome articles about MS-DOS, the innards of MBIs, and 8086 assembler.

For the past month or so I have been in hot pursuit of R. Conns ZCPR system, which C. Bowden writes about. More importantly for me is Richard Conn's SYSLIB. Anyone programming with CP/M and assembler should obtain SYSLIB. It is a collection of some 100+ (I haven't counted) subroutines, which do all the nasty chores for us assembler proggers. I heartily recommend it!

The latest version of SYSLIB is 3.6, and is available from the CPMUG's Software Library in volumes SIG/M 261, 262, 263, which also contain ZCPR 3. Incidentally, the HELP program mentioned by C. Bowden for ZCPR 3 will not work under CP/M. However, R. Conns earlier HELP.COM version 2.4 on CPMUG volumes SIG/M 99 and 103 will with minor mods and re-assembly. I have converted HELP 2.4 for CP/M and converted the .ASM source to .MAC. It also uses SYSLIB. If anyone would like a copy it will be donated back to the CPMUG.

Many thanks for an interesting (and on-time) read.

Yours sincerely, Ian Cullen, Ipswich, Suffolk.

More various points.

Dear Sir,

I'm very pleasantly surprised - Scorpio News was out on time (if not before) for the second occasion. Do keep it up! To judge from your editorial, however, the revival in fortunes of 80-BUS publishing will be short-lived - which would be tragic for all concerned.

Could I thank all those readers who kindly contacted me with offers of circuit diagrams and handbooks in response to my request? So far, 12 people offered help and I have all the information I need. Perhaps I'm biassed, but I can't imagine this level of response in other computer circles.

Regarding the possible extension of the magazine to include PC look-alikes - not a bad idea if people can be persuaded to write suitable articles and for others to purchase the magazine. The main problem is going to be how to extend your circulation once the PC articles come pouring in - and another is going to be how to get articles of general (or even specific) interest to the existing or potential audience which aren't done already by glossier publications (although most of these tend to restrict themselves to software reviews). It's worth trying, however, even if it does offend the purists (shades of an earlier debate in INMC80?).

I have a particular interest since I would like to know how to doctor the BIOS of my Amstrad so that I can fit and use 80 track drives - one gets used to the 784k of the Gemini QDDS and the 360k currently available is pathetic. Obviously, one needs to have one or both drives software switchable to 40 track. Any volunteers?

One of the best features about this magazine and its predecessors is Dr Dark's Diary. Always interesting and usually very enjoyable..... but - the most recent episode seems to have been a bit hard on one or two of us (sob!). So he doesn't like Fortran - well, perhaps he could write a guide to Pascal - which he knows a lot about. I would welcome a decent guide to an excellent language which has so many adherents (and so would others to judge from the letters pages in the same issue). I use Fortran because most of the programs I need are written in it - and it isn't worth the hassle of translating 30,000 or so lines of Fortran IV or Fortran 77 into Pascal. Perhaps I might be able to do it if I sampled some of his anchovy wine!

Of course the Dobbs Benchmark program is silly - no one in their right mind would seriously want to fiddle around with tangents and arctangents around the 90 degree mark - a quick look at a table of tangents would show how rapidly the value changes. The point of the program and the article was to demonstrate how well, or otherwise, various interpreters and compilers coped with such a problem and how fast or slow they ran on a range of machines which might be of interest to readers. The general point is still valid - a machine/language combination which took a long time to produce a result or which produced a poor result could probably be relied upon to give a poor performance in other applications. I know that this is a generalisation but hard experience tends to bear it out. I did appreciate the explanation of the poor result obtained with HiSoft Pascal - which does explain why virtually no other Pascal or Fortran IV compiler has TAN as a supplied function - but it is available in some implementations of Fortran 77.

Well, how about it, Chris? I'm sure that the Editor could find a few pages (or even more) for the first instalment of your guide to Pascal.

Yours sincerely, P.D. Coker, Orpington, Kent.

The ZCPR3 System by C. Bowden G30CB

Part 2 - ZCPR3 Installation - Software Tools

Although Z3 is accompanied by a considerable amount of documentation on its installation it is sometimes difficult to pick out the important points, and often, there are easier or better methods of carrying out the work. This is particularly true in respect of the choice of software tools and methods. In my opinion, the installation information provided on patching the Cold Boot routines is rather too specific to the particular BIOS used by the author of Z3, and made to appear very rigid in application. Those without much experience in machine code and operating systems might find it difficult to understand or implement. Some tips and alternative methods of installation are suggested in this article.

The installation of Z3 falls into several operations.

a) Choosing the commands required in the Z3 CCP and RCP. Choosing names for the named directory. Deciding on the Default 'PATH' and startup commands. Choosing the segment and buffer memory mapping to be utilized.

b) Editing the Z3 .LIB files for the generation of Z3 and each of the required segments, and assembling the associated .ASM files.

c) Creating the Z3 segments from the .HEX files, and renaming them as SYS.xxx, as appropriate.

d) Generating a CP/M image file with the CCP, BDOS and BIOS at the desired addresses.

e) Modifying the BIOS, and reassembling, or patching the CP/M image. Alternatives, avoiding BIOS modifications, are also possible.

f) Overlaying the CCP, BIOS (and BDOS?), and saving this new system, optionally with the Z3 segments and Buffers included. (See Later.)

g) Sysgening or Syswinning this new system, and 'Booting' it.

h) Installing the 70 odd Z3 utilities.

At this stage, Z3 should be fully operational. Further changes can be carried out after experience of the system has been gained, and since the source codes of all of the utilities is provided, these may be customized as well, if desired, although in general, most people will be happy with them as they are.

Some of the above operations will be described in more detail under 'Methods', but it is useful to consider the many options available to the installer in carrying them out.

An essential part of the installation is to assemble the .ASM source files needed to create ZCPR3 and the system segments. These files are written in 8080 mnemonics, and use several macros to define Z80 opcodes.

This means that the DRI Macro Assembler MAC.COM must be utilized to assemble these files as they stand. If one does not have MAC, then one could resort to trying one of several other methods. If M80 is available, it might assemble these files, with minor alterations and removal of the macros. Alternatively the macros could be removed, and the code translated into Z80. I did this with Z2, as an experiment, using the public domain (PD) program XLATE3.

After the first try at assembly with M80 there were about a dozen errors that were easily resolved by editing the source code, and I soon obtained a file that produced code compatible with that from the 8080 source via MAC. As another alternative, the macros in the relevant .ASM file could possibly substituted with some alternative coding, that would allow assembly with ASM.COM. As a last

resort, one could try to find someone with MAC.COM and request him to assemble the files for you.

This is the main problem facing the installer. If M80 and L80 are not available, it means that the utilities cannot be modified to any extent, but this is no real problem since the normal performance is quite acceptable.

For the rest of the installation, everyone will have DDT.COM, which will be sufficient. Since there are several ways of tackling the various problems, however, it is useful to review the alternative 'tools'.

a) **Editors.** It is necessary to carry out editing of the various .LIB files associated with Z3 and the segments, and the BIOS source code as well if available. Since most people will have their favourite editor, there is not much to say here. It should be borne in mind though, that some source files are too long to fit into RAM, so that editors like ED or Wordstar might be needed. (Minor changes can be made with a disk editor like SPZ.COM, which can be quicker and easier than using an editor.)

b) **Assemblers.** If MAC.COM is available, then the problems referred to above will not appear. The availability of M80 and L80 will permit the utilities to be freely altered.

c) **Debug Utilities.** DDT.COM, ZSID.COM, Z8E.COM (PD), GEMDEBUG.COM. The main advantage of DDT and ZSID is that they can 'LOAD' .HEX files. Loading .HEX files under DDT or ZSID is not difficult, but error prone due to the need to use variable offsets. I much prefer to overlay using GEMDEBUG. The latter is not so sophisticated as DDT or ZSID, but has a much better screen display, command structure and several other useful features.

d) **Loading Utilities.** If LOAD.COM is used to load .HEX files, the results are not exactly useful, when the target address is in high RAM!!. If MLOAD23.COM (PD) is available, then it is easier to MLOAD the .HEX file, and to use the result as an overlay or Z3 segment. For example, assume that the 2k Byte Z3 CCP is to reside at C800H, as in my current system. After assembly of ZCPR3.ASM, if MLOAD is applied to ZCPR3.HEX, I am left with a disk file 2K long, with addresses loaded for C800H, that can then easily be used as an overlay.

e) **Disk Editors.** Where a number of bytes might need to be changed, as for example to alter names in a named directory segment, to alter the default path on the system track of the disk, to change the assembly address of BDOSZ (56k long), to patch the startup command in the system image cold boot routines, then I increasingly use a disk editor. My favourite is SPZ.COM (PD). It has a very friendly user interface, and can cursor edit both ASCII and HEX bytes of a file or disk sector, which may be the system tracks, or directory data. DU.COM (PD) in one of its many versions is also very versatile, but I find SPZ so much pleasanter to use. (A version of DU accompanies Z3, and it is more interactive than earlier versions.)

f) **Other Utilities.** One of the several PD utilities such as BDOSLOC, FINDCCP, LOCATE, Z3LOC to discover the operating CP/M system addresses makes it much easier to find out about system addresses, especially if the installer is not used to system work.

ZCPR3 Installation - Some Suggested Methods

Once it has been decided which Z3 segments and buffers will be implemented, a memory map should be drawn up. In the installation information supplied with Z3, it is suggested that certain areas are preset to all 00's. The main reason for this is to avoid system problems that could occur after the system is booted, if spurious data is left in any of the system buffers, and these are accessed before segments are loaded (which is by no means impossible).

During installation of Z3 on an a friend's Alphatronic PC, some very peculiar errors arose. An oversight had resulted in the SYS.NDR segment not having been loaded or initialized, and the locations in question were filled with random rubbish. The data found there by the Z3 CCP was being interpreted as the NDR segment, and select errors were prominent amongst the reported errors.

The installation notes suggest clearing (part of) each discreet segment or buffer area in turn. This works well, but it requires rather more code, since the code required is virtually repeated for each area to be preset. It is much simpler and does not take much longer to clear all of the Z3 memory area. Initialization of startup commands, paths etc, can then follow. This reduces the total amount of code needed for cold boot patches, and could simplify installation where BIOS source is not available.

The standard method of system initialization in the Cold Boot routines is to clear buffers and segment areas, initialize the path, wheel byte, and startup commands, print signon messages, and then boot the system. The startup command uses the supplied utility LDR.COM to load the required segments. On a floppy based system, there does not usually appear to be much alternative to this method of booting due to space limitations on the system tracks of the disk, but if one has spare space on the floppy system track, or a Winchester, there is a better way.

A track on a Winchester disk may be 8k - 8.5k bytes long. If two tracks are reserved for the system, then about 16 - 17k of space is available. After subtracting BDOS, CCP and Cold Boot Loader, over 10k is available for the BIOS and Z3 segments and buffers, and BIOS workspaces. This means that the Z3 extensions can be considered to be part of the system, and loaded with the system at cold boot. Since the buffers can be optionally loaded from the system tracks as well, already initialized to zeroes, or with set messages, commands, etc, the amount of cold boot initialization becomes minimal, with only the default path and wheel byte needing setting up, although these could be defined somewhere in the Z3 memory area and loaded with the system too. This has the further advantage that no BIOS modifications are needed, except in the Screen Edit routine. It then also becomes unnecessary to use LDR.COM, except to change the default segments for special purposes.

If this approach is used, but only some of the areas are loaded from disk (e.g. where on say a floppy system there is some spare room on the system track, but not enough for ALL of Z3), then initialization of non-loaded Z3 segment and buffer areas must be selective, to avoid erasing data that have just arrived in RAM from the system tracks. It then becomes a good idea to group the program segments together just above the BIOS workspace, and to group buffers above this.

Whatever the system adopted, disk space for the system is not a problem. The amount of RAM allocated to the system is a problem however, since all extensions of the BIOS or Z3 eat into the TPA. My extensively customized BIOS, together with workspaces, takes up just under 5k, and the Z3 system uses 3.5k resulting in a total system size of 14k, including CCP, and I am determined not to allow it to get any bigger. Since I am also always trying to add features to the BIOS, I am also looking for ways to economise on space. One thing that I have recently done is to utilize some 80 spare bytes in the TCAP part of the ENV segment, to hold the system message and Clock string that are displayed on the (normally locked) top line of the screen. This has released 80 bytes in the BIOS area for more code, but of course, means that that particular configuration of the BIOS will only work correctly in a Z3 environment.

The BIOS plus Workspace size is one of the important considerations, since this is considerably larger than BIOS plus Cold boot routines. I have no problem in expanding cold boot activities, and in providing extensive signon messages. The amount of workspace area needed is determined by the hardware used, and so is relatively constant for a given system. This means that the BIOS permanent code is the part that needs to be kept in check. I have recently helped two friends instal Z3 on systems based on Nascoms, and the big problem was in the amount of space required by the BIOS code for the keyboard routines. With a Winchester

based system, space was not available in RAM for all of the features that I normally assemble into my BIOS, unless the code was allowed to expand beyond 5k.

Each installer must come to his own decision on what is considered to be the optimum trade off between TPA, BIOS size and Z3 support. Some of the considerations above show how adaptable the installation may have to be.

It is necessary, during installation, to find out the current address at which the CCP, BDOS and BIOS are residing. Before CP/M is relocated lower in memory to make room for Z3, the current operating address is needed, so that, after subtracting the extra RAM to be allocated to Z3, the new addresses are known. BIOS workspaces must also be allowed for, and it would not be safe to just look for the end of the cold boot in the system image (which is normally sign-on messages), subtract 2100H (start of BIOS in standard Gemini system Image), and take the result as indicating the BIOS length. A better guide is to find the BIOS starting address on a running CP/M system and to subtract this from the top of RAM, (or ROM address if ROM based software is used in the system, as in the Alphatronic PC).

Having decided on the addresses of each segment and buffer to be incorporated, the files Z3HDR.LIB and Z3BASE.LIB must be edited. This will allow the correct assembly of the Z3 CCP and will provide the information needed by other assembly operations that also read these .LIB files. Z3BASE defines the system addresses that have been chosen, and Z3HDR defines the commands and parameters available or acceptable to the Z3 CCP. The .LIB files for SYSFCP, SYSRCP, SYSENV, SYSIOP and SYSNDR may then be edited, but if any one or more of these segments is not to be used, edit of those .LIB's may be skipped.

These files may then be assembled (e.g. MAC ZCPR3 $SZ PZ) to produce the relevant .HEX files. The $SZ and $PZ parameters suppress output of the Symbol and .PRN listing files otherwise produced during assembly. If MLOAD is available, the .HEX files may be loaded, and the resulting .COM files should be given the name SYS.RCP, SYS.ENV, SYS.NDR, SYS.FCP and SYS.IOP as appropriate, since the names used MUST be acceptable to LDR.COM. If MLOAD is not available, DDT or ZSID will have to be used to read the .HEX. This is explained in more detail later.

Once SYS.NDR and SYS.ENV have been constructed, any future mods. to these could well be made with SPZ.COM or a debug program, since these files are very short. Reference to the installation manual for the .ENV and TCAP will give the required information. There is little in the SYS.ENV that would need to be altered, once set up, unless a system 'shuffle' is made. Several different named directory segments could quickly be made by copying, renaming and editing with SPZ or DU.

The BIOS cold boot alterations are made to appear unnecessarily complex in the installation manual. If the source code of the BIOS is available, revised signon messages and all sorts of frills are possible, but all that is really needed, just before the jump to warm boot, from cold boot, is:

a) Code to clear Buffers and segments.
b) Code to set the Wheel Byte.
c) Code to set up the Default Path.
d) Code to set up the Startup Command string.

If the BIOS code is available, or an installer is prepared to do some work with a debug program or disassembler, then the necessary code can be 'patched' into the BIOS.

Appendix A lists an example of the code needed in source code form. The code listed assumes that the source code of the BIOS may not be available, and so the listing indicates additional modifications, in mnemonic form, that are required to be patched into the actual BIOS code to redirect the cold boot jump to access the new code before it jumps to the warm boot. It also assumes that there is enough room on the disk system tracks for the extra code. If this is a problem,

discrete trimming of the signon messages might help. These are usually output until a 0 is found.

If the BIOS source code is available, the modifications can be made tidier. The extra code could be added just before signon messages are printed, and the extra data could be placed after the jump to warmboot, but before the actual signon message.

The code listed is really ALL that is needed to implement Z3, but it is rather obscured in the Z3 installation instructions. It consists of about 45 bytes of Z80 code and about 75 bytes of data, which could be trimmed by some 20 bytes if desired by removal of the Z3 welcome message that is included in the startup command string. You can be more elaborate if you wish, but there is no need.

Note that if segments are loaded to memory from disk files by LDR.COM, the SYS.ENV segment MUST be loaded first if it is not already present, since LDR.COM then refers to it to find out where to load the other segments.

Installation without BIOS modifications.

If the Z3 segments and buffers are loaded from the system tracks of a Winchester or Floppy, all that is needed in the cold boot is to set the Wheel byte, and path unless they are also defined in the system track area, for example in the spare space in the TCAP.

In the case where insufficient system track space is available, and BIOS modifications are to be avoided, there seems to be a possible solution. During implementation of Z3 on a friend's Alphatronic PC, I had booted a system that loaded the segments and initialized the buffers, and Z3 was running. A little later I booted, by mistake, a disk that did not have LDR.COM or any of the SYS segment files on it, but did have a Z3 CCP in the system on the system tracks. When this disk booted, I got the error message:

'LDR SYS.ENV,SYS.NDR,SYS,RCP?'

since these files could not be found or loaded. When Z3 commands were given however, they all worked correctly, since all necessary data was in RAM already from the previous BOOT. So, provided they have not been corrupted, it is sufficient to get the Z3 segments, etc, into RAM only once, on the initial Cold boot. They do NOT need to be reloaded unless they are later corrupted for some reason. This suggests that the problem can be solved as follows.

A CP/M system is created that has the CCP, BDOS and BIOS at the same addresses as the Z3 system, but with a CCPZ or ZCPR Vers. 1 CCP. This can be booted, and will it will not use RAM in the area to be used by Z3 segments and buffers. The 'GET' command of this CCP then can be used to load a complete Z3 composite 'SEGMENT.BIT' file from the disk to the Z3 RAM area. (This could be made an auto run command.) Once this file has been loaded, a disk with a Z3 CCP in the system track can be booted, and it will find everything ready in RAM when it starts up. Alternative ways of loading the 'SEGMENT.BIT' might also be devised, such as loading and relocating from the Z3 boot disk itself. The method described above is probably the simplest though. The 'SEGMENT.BIT' file can be made up in a similar way to that used to build up the Z3 part of a complete Winchester based system, as will now be described.

Generating a ZCPR3 System for Direct Booting.

Fig. 1 shows in diagrammatic form a typical Z3 system supporting all features except redirected I/O. The addresses on the left hand side are those that the various parts would start at when the system is in RAM. Those on the right are referenced to a system 'image' file, that is to be written to the system tracks of a Winchester, and will subsequently load to RAM. Note that these addresses and those given below represent only one of many possible configurations of CP/M and Z3.

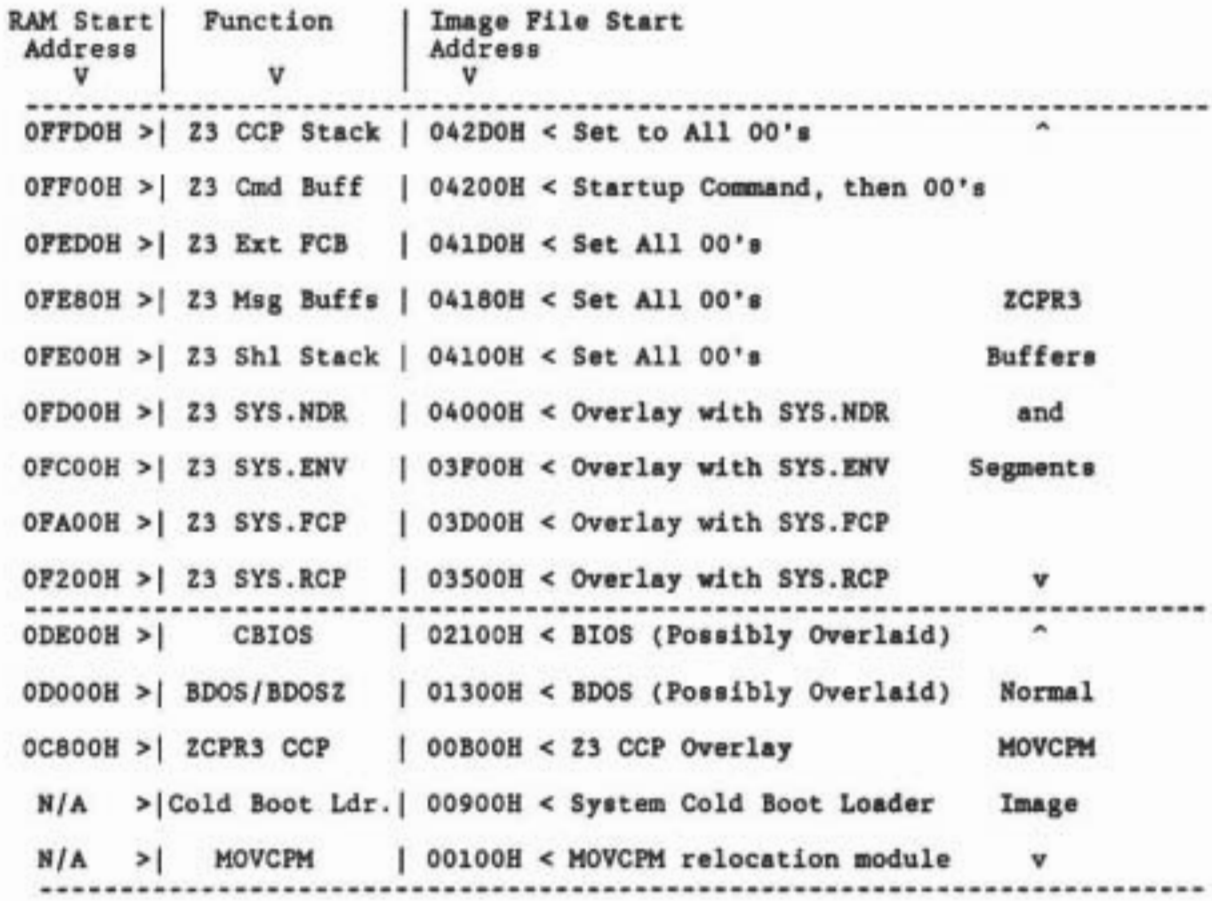


Fig 1. Setting up an entire Z3 System Track Image.

The starting point for this operation is MOVCPM.COM. The first step is to generate a CP/M with CCP starting at 0C800H. This can be achieved by setting the byte at 023DH in MOVCPM to 18h to reserve 24 extra pages, and then issuing the command MOVCPM 64 *. Once MOVCPM has completed its task, the system image in the TPA is saved to a file - SAVE 42 CPMC8.COM. We now have a CP/M system that should run at 0C800H. This can be checked by SYSGENing it, booting and running one of the CPM address display utilities described above. Note that the byte at 023DH in MOVCPM reserves extra memory for a larger or smaller BIOS in 256 byte units. Add 1 for each extra 'page' of memory required.

If you are using Gemini BIOS 3, then MOVCPM is not used. You can still generate your system, CPM64W for example, reserving sufficient workspace pages in the SYSTEMW.CFG file to get CP/M down to the target area.

The important thing to remember is that the system that is being constructed is to boot from a Winchester, so the Cold Boot Loader sector must be suitable for this purpose. The loader in CPM64W is O.K. but those in CPM64F and CPM64FW are not. If it is intended to create a completely new system, (i.e. Z3 CCP, BDOSZ, and a modified or non Gemini BIOS, the Winchester cold boot loader can be transported from a CPM64W file, or the new parts of CP/M can be overlaid on the CPM64W file.

CPMC8.COM or CPM64W can be loaded into memory with a DEBUG program. At this stage, all that will exist in RAM will be that shown in the diagram from 100H to about 3000H. The top address is vague because the exact address will depend on the BIOS present, and there may be other bit map data present. The cold boot loader that loads the rest of CP/M off of the disk into memory will be present at 900H, CCP at B00H, BDOS at 1300H, and BIOS at 2100H. The MOVCPM relocator will occupy the area from 100H to 8FFH.

If a new BIOS is to be used, the first step should be to load it in. Assuming that it is a file called NEWBIOS.COM assembled to run in memory at 0DE00H, then the command INEWBIOS.COM, R2000 should be given. Debug utilities load binary files with a default offset of 100H, so this should result in the NEWBIOS overlaying from 2100H. This can be checked by a D2100 command. The jump table should be present starting at 2100H. At the end of the 'R' command the screen should display the new top of memory of the system image.

If the original BIOS is being retained, then its top should be found if possible, and will usually be just after the signon messages. Once the BIOS top has been found, memory from the top of the BIOS should be filled with 00's to about 5000H, in order to clear all Z3 areas. The command FXXXX,5000,00 will do this. XXXX will be the BIOS ending address.

If BDOSZ is to be used, it can then be loaded to 1300H.

If it is necessary to read in .HEX files, then DDT or ZSID are needed as they can handle .HEX format files. The read OFFSET, once calculated, will be the same for ALL .HEX overlays in a given system, since the address in the memory image is offset from the running address by the same amount. The offset can be calculated by using the H command, (H low,high) where low and high are the image and RAM address of any one segment or buffer. The image address of the RCP is 03500H, and the RAM address is 0F200H. The command H,3500,F200 gives the answer 2700,4300. Now 4300H added to 0F200H gives 013500H. The leading '1' is lost since it cannot be contained in a 16 bit value, leaving 03500H, which is the target address. Thus 04300H is the answer needed. (Since the OFFSET value is added to the address in the .HEX file as it is loaded).

Each .HEX segment may be loaded in turn. (e.g. ISYSRCP.HEX, R4300) or if it is already loaded by MLOAD, (and assuming the file is named as a standard Z3 segment), ISYS.RCP, R3400 (remember the default 100h offset with binary files). Then the ZCPR3 CCP may be read in using one of these two methods, (IZCPR3.HEX, R4300 or IZCPR3.COM, RA00). The D command should be used frequently to check that the results are as required.

A cold boot startup command can now be implemented, by modifying the image memory at 04200H. I prefer to use SPZ.COM for this type of job since it is easier to correct errors and get things exactly right, but this is a matter of preference.

With direct booting of Z3 there is no need to use LDR.COM to load any segments, but a startup command may be required. My startup command is:

```
CSET;ECHO Type ^C for CP/M, or any other key for HELP;SAK;HELP ME
```

which loads a preferred character set to the SVC, and then prints the text message via the RCP ECHO command. Z3 then uses the SAK utility to wait for user input. If this is ^C, then CP/M is entered. Any other key proceeds to the next command in the buffer. This causes Z3 to use its HELP.COM utility, which in turn loads an online help file called ME.HLP that I wrote to summarize the features of Z3.

The process is now complete but for two more small but very important modifications that may have to be be made for the system Boot to work correctly. These concern the Cold Boot loader, and the Winchester system write utility SYSWIN.

The cold boot loader for both Floppy and Winnie systems refers to three vital locations. One is the address to which CP/M will be loaded. One is the Cold Boot Jump address of the BIOS, to which the loader will jump once CP/M is loaded. These two will normally have been adjusted during the MOVCPM or GENSYS operation, but might have to be patched in some circumstances. It is not difficult to find these addresses in the loader, either as conditional or unconditional jumps, e.g. LD HL,xxxx instructions, for a copy operation or for a JP (HL). The third significant item is the number of sectors that the loader will read from the system tracks of the relevant disk. Since the system has

grown by about 1000H-1500H bytes in length, these extra sectors have to be loaded from the Disk. It is necessary to adjust the sector counter. In my Winchester Boot Loader it is located at 093AH in the system image, and at 0909H in the Floppy loader. The value was 13H in both cases. It will need to be increased. The value of 01CH will load a system configured as shown in Fig. 1. Too large a value will load a system over the top of RAM, crashing the Load process. Too small a value will read insufficient data from the disk.

The debug program can now be exited, and the new system saved - SAVE 66 FULLZ3.SYS. This will save all of the TPA from 0100H to 4300H. It only remains to write this to the Winchester. Note that the bytes from 0100H to 08FFH in the system image file are discarded by SYSGEN and SYSWIN. The actual write starts at 0900H. These bytes MUST be save in the system image file however, in order that the required data starts at 0900H.

The utility SYSWIN.COM is used to write a system to a Winchester, but the other remaining problem now becomes apparent. SYSWIN is set to write a normal sized system. The sector counter must again be altered. It is necessary to write all of the much larger system to the Winnie and the Cold Boot Loader sector must be written as well. In my version of SYSWIN this count (at 050FH) was 013H, and I have set it to 01DH.

N.B. I have found that SYSWIN will not work correctly if run from the Winnie itself, or if the system image file is on the Winnie - probably due to hard coded disk drivers. I use both these files from a floppy.

There is another way to get the system written to the Winnie. SYSGEN will carry out a part of the process, but fails part way along - I can't remember where. If you are really masochistic, you can use the scratchpad feature of SPZ to write 128 byte chunks of your FULLZ3.SYS file to the Winnie system tracks.

For the system to boot correctly from the Winnie on reset or startup, SIMON 4 is needed. (Other Versions may be O.K.) or the utility BOOT.COM can be used. These are not PD programs. Your friendly Gemini dealer should be able to supply you with SIMON4, BOOT.COM and SYSWIN.COM, unless Gemini have become as difficult about System Utilities as they have about source code. They might insist on you buying or upgrading to BIOS 3.x.

If Gemini intend to continue supporting 80-BUS products (as they say they intend to do), then it is in their interests to serve their customers requirements. If they intend to let these products lapse, then they have nothing to loose in supplying existing users with the programs and information. Either way I think their attitude is wrong - riding my hobby horse again !!

If the Z3 area from 03500H to 04300H only is needed, in order to load as a complete 'SEGMENT.BIT' so as to avoid BIOS mods. as described above, it is suggested that a debug program is run, but without loading a system image. (e.g. run DDT or ZSID if .HEX files are to be processed). Memory from 100H to about 05000H should be first filled with 00's. The segment parts can then be read in at 3500H, etc; as described above. The default startup command can be set as described above. Wheel and Path could be defined in the spare space in the TCAP of the SYS.ENV segment.

Then the required data area can be moved to 100H - M3500,4300,100. This will move a block 0E00 long down to 100H. Exit from the debug program and 13 pages saved - SAVE 13 SEGMENT.BIT. This file can then be loaded to 0F200H by any appropriate means to furnish Z3 with its requirements. Note that LDR.COM cannot load this file, but a CCPZ GET command could be used as already suggested. (Or maybe LDR.COM modified via the .MAC file so as to be able to load it).

ZCPR3 Utility Installation.

The installation of the Z3 utilities is straightforward. It is necessary to carry out three steps.

a) Set up a correctly configured SYS.ENV file.

b) Create a simple list of files to be installed, using an editor. This file should be given a name with an extension .INS. (A ZCPR3.INS file is included on the Z3 disks, but it does not include the .COM files on Sig. Vol. 200. Neither does it allow for any renaming. e.g. I prefer to call MCOPY.COM - COPY.COM.).

c) Put the files to be installed, SYS.ENV, Z3INS.COM and the .INS file in one user area, and issue the command -Z3INS SYS.ENV XXXX.INS.

The files named in the file XXXX.INS will be installed for the current data as held in SYS.ENV.

This completes the discussion on some of the devious ways in which one can implement the Z3 system. I hope that those of you out there who have not tried Z3 are encouraged to have a go. The results are well worth it. I have heard that there may be further developments of Z3 that may include bank switching and multi tasking, but I have no details.

****

Appendix A. Typical BIOS modifications.

This part could be in the equates section of your BIOS source, if you have it. It assumes Z3 segments and buffers start at 0F200H and occupy RAM to 0FFFFH. Address 0F200H is where the Z3 area starts in my system. Yours could be different. The same applies to the other addresses.

```
;
RAMTOP  equ     0FFFFH          ;Top of memory
SEGST   equ     0F200H          ;Start of Z3 Segment and Buffer areas.
whladr  equ     003BH           ;Address of Wheel Byte
patadr  equ     0040H           ;Start of defined default path
mcladr  equ     0FF00H          ;Multiple Command Buffer
clbuff  equ     200             ;Command Buffer is 200 bytes long
;
;***********

;Patch the Screen edit code to work with DIR: names in the Prompt.
;Find the Screen Edit Buffer and look for some code nearby like -
STRIP:  LD HL,EDBUF+3
        LD B,3
;
;Or its HEX equivalent in the BIOS Image.
;
;Change the code, or patch the BIOS Image to be
STRIP:  LD HL,EDBUF+12
        LD B,12
;

;Now To the Cold Boot area of your BIOS, near the end.
```

The Warm Boot Jump must be found. The code below assumes your BIOS has the Cold Boot routines in the BIOS workspace area, where they will be overlaid when the system is running. The Warm Boot routines are usually located near the start of the BIOS. Examine the First two jumps in the jump table at the start of the BIOS (normally 17 jump instructions) to get the Cold and Warm boot routine start Addresses. Note that the jump to WBOOT1 found near the end of the cold boot routines as shown below will probably not be to the start of the Warm boot routines as found from the second jump table address, but some tens of bytes into the Warm boot routines.

```
;
        LD HL,SIGNON            ;BIOS code to print signon messages
                                ;usually located here.
        CALL PMSG               ;BIOS Print String routine.

[       JP WBOOT1               ;JUMP INTO Warm Boot - CHANGE TO > ]
        JP EXTRA
;
;*********
;
SIGNON: DEFB 'Super Duper...    ;The system signon messages
        DEFB 'and more rot..
        DEFB 0                  ;Signon terminator (Probably 0)
;
        ;NORMAL END OF COLD BOOT CODE
        ;Workspaces usually extend beyond this.
;
;*********
```

Patch in the bytes generated by this code if no source is available. In this case, substitute the equates defined earlier for the symbols below.

```
;This to be in cold boot, just after SIGNON Messages.
;
; First clear memory to all 00's.
;
EXTRA:  ld hl,segst             ;Start Address of ZCPR3 area
        ld de,RAMTOP-SEGST      ;Length of Z3 segment and buffer area
loop:   xor a                   ;Clear Acc.
        ld (hl),a               ;Clear Byte in target area
        inc hl                  ;Next byte
        dec de
        ld a,d
        or e
        jr nz,loop              ;Loop until all bytes cleared
;
; Now set the wheel byte
;
setwhl: ld hl,whladr
        xor a                   ;Clear Acc. to known state
        dec a                   ;Make non zero
        ld (hl),a               ;Set Wheel byte
;
; Now copy the default path to 040H, etc;.
;
        ld hl,extpth            ;Defined path
        ld de,patadr            ;Path address
        ld bc,endpth-extpth     ;Path string length
        ldir                    ;Copy it.
;
; Now copy the startup command to the Multiple Command Line Buffer
;
        ld hl,stcmd             ;Defined startup commands
        ld de,mcladr            ;Command buffer address
        ld bc,endcmd-stcmd      ;Length of command string
        ldir                    ;Copy it
;
; Initialization done, except for loading of segments, which will be done
; after system has booted, via LDR.COM.
;
        jp wboot1               ;Jump into to warm boot - same as the jump
                                ;that was just before the signon messages.
;
;**********
;
```

```
;Data area - The code to be carried into the system.
;
;Define the default path - Typical assignment
;
extpth: defb 1,0                ;A, user 0
        defb 1,15               ;A, user 15
        defb 1,$                ;A, current user
        defb 2,0                ;B, user 0
        defb 2,$                ;B, current user
        if wini
        defb 3,0                ;C, user 0, if C exists.
        endif
        defb 0                  ;Terminating zero - Essential
endpth:
;
;Typical Cold Boot Startup command - 200 Chrs. maximum.
;
stcmd:  defw mcladr+4           ;Point 4 bytes into buffer
        defw clbuff             ;Say how long it is
        defb 0
thecmd: defb 'LDR SYS.ENV,SYS.RCP,SYS.FCP,SYS.NDR;ECHO Welcome to ZCPR3'
        defb 0                  ;Terminator
endcmd:
;
                        **********
```

I should like to thank Peter Bell and Mike Waters for their help. In the first issue of Scorpio News, I referred to a bug in BDOSZ. Apparently I had been using an old version. The problem has now been resolved.

Soft Warm Boot on a Nascom — by R. Mohamed

No, its nothing to do with footwear. The title refers to a software controlled reset sequence or a warm boot (in CP/M) on my Nascom 2.

The idea was developed from an article by David Parkinson (INMC-80 News issue 4) describing the use of the GM803 EPROM board as a ROM-disk (i.e. copying software from ROM to RAM). I developed software (E.O.S. - EPROM Operating System of course) to copy any programs held in ROM into memory at their usual address and execute them or just load the software into RAM at any address.

With my Nascom 2 power-on-jump switch set to A000h and EOS present here on a reset, the system worked well. But if I was using disks, say under CP/M, then pressing reset would load Nas-Sys 3 and display the EOS menu. Not very helpful.

I had the choice of altering the power-on-jump , when working with disks, via hardware or software. Some later articles advocated the use of simple hardware (diodes and switches) to alter the jump-on-reset address. I decided against this approach for two reasons, it is technically inelegant and more importantly, my Nascom 2 is encased in a large steel box. This made is practically impossible to get at the switches, even to the extent of adding a remote switch.

The software approach has been successful and over the years has been enhanced to its current level. The software switch allows for 3 options, warm boot from ROM (standard system), warm boot from disk (non-standard system) or a reset to Nas-Sys and the EOS menu.

Referring to the listing, the software recognises the remains of CP/M code at page 0 by means of the two jump instructions at 0000h and 0005h. It is not particularly rigorous, but I have not had any problems. If both of these jumps are not present then it is assumed that Nas-Sys was running so execution jumps to the rest of the EOS code ('reset').

If the jump instructions are present, the system size is checked to determine whether warm boot should be off ROM (standard system) or disk (non-standard system). In each case a small piece of code, to turn off the ROM board and jump to the correct entry point, is copied to an area of free RAM (0020h) and executed.

The soft switch can be used with any CPU card that has jump-on-reset hardware (e.g. Nascom 2) and a GM803 or GM853 EPROM card. The EPROM card must be configured to be paged in after a reset.

Individual users may wish to add other features to the soft switching mechanism, such as forced jumps to, say, BASIC or Naspen. This could be implemented in a similar fashion to that on the BBC Microcomputer. That is, check for any pressed keys and force jump accordingly, i.e. pressing 'B' at reset time will force BASIC to be loaded and executed.

```
;       OPERATING SYSTEM FOR GM803 EPROM BOARD
;
;       Version 1.3                5/5/85

;       GM803 must select page 0 on system reset

;       Amended as follows.....

;       1) CP/M boot is now at 0B800H
;          (this to be the auto density boot)
;
;       2) CP/M warm boot is recognised
;          on a system reset
;          (with warm boot from EPROM image)
```

```
.Z80
aseg

zcpr    equ   0c400h      ;entry into
ccp     equ   zcpr        ;ccp or zccp
bdos    equ   ccp+0800h   ;bdos start
wboote  equ   0da03h      ;for 56K system
;
; HARDWARE VARIABLES
;
PMODE   EQU   0FFh        ;GM803 control port
BEGIM   EQU   0A000h      ;EPROMs actual address on GM803
        org   0100h

        .phase  BEGIN     ;code for here

start:  JP    BEGIM+3     ;get around reset diversion
;       CP/M warm boot recognition code
;       yours can be more rigourous
        ld    a,(0)       ;start the checks
        cp    0c3h        ;= JP
        jp    nz,reset    ;not cp/m
        ld    a,(5)
        cp    0c3h        ;= JP
        jr    nz,reset    ;not cp/m
        xor   a           ;clear carry flag
        ld    hl,(1)      ;last wboot
        ld    de,0da03h   ;56K wboot
        sbc   hl,de       ;now check size
        jr    nz,trumb    ;NO not 56K system
        ld    hl,6000h    ;EPROM CP/M image from here
        ld    de,zcpr     ;and goes here in RAM
        ld    bc,1600h    ;length
        ldir              ;copy it up
        ld    sp,0080h    ;temporary stack
        call  bpinit      ;initialise workspace
        ld    hl,copy     ;code to switch GM803
        ld    de,0020h    ;page 0 off goes here
        ld    a,c         ;save selected drive
        ld    bc,copye-copy ;code length
        ldir              ;copy to free RAM
        ld    c,a         ;restore drive
        jp    0020h       ;page GM803 off

;       Initialise base page after warm boot from GM803
bpinit: ld    a,0c3h      ;replace JP's
        ld    (0),a       ;here
        ld    (5),a       ;and here
        ld    hl,bdos+6   ;place bdos jump
        ld    (6),hl      ;vector into RAM
        ld    hl,wboote   ;same for 56K warm boot
        ld    (1),hl      ;vector
```

Listing continued on page 41.

A Visit to Io Research by P.A. Greenhalgh

Io Research was one of the first companies producing add-on products for Nasbus/80-BUS. Founded in 1980, they initially produced an EPROM programmer, 8-bit A/D board, and a graphics board that sat between the Nasbus/80-BUS and a Nascom 32K RAM A board, using the RAM on this latter board as its display memory. Io have not produced any of these products for a number of years now, although the A/D board is in fact still available - it is now produced by Gemini as the GM824.

Another of Io's early products was the Pluto 1 colour graphics board, and it is the success of this and its derivatives and descendants that have turned Io into one of the major computer graphics companies in the UK.

A few weeks ago I went along to Io, who are based in Barnet, to have a look at their current product range, and to be given demonstrations of some of these items. What follows is an overview of the events of that visit, with descriptions and prices of many of their products. No apologies are made if it all comes over rather like a sales leaflet, but the Pluto range has had very few mentions in Scorpio News and its predecessors. Io Research have probably been the most successful of the companies producing 80-BUS compatible products, and this article is intended to bring the readership up to date on the company and its product range, which is probably far greater than most people are aware.

Video demo

To get me in a graphics mood, I was first shown a video produced by one of their customers, Diverse Productions. This is a video company that does a lot of work for Channel 4, especially for the "Diverse Reports" program. It would be unreasonable to try and describe in words the contents of a graphics video, so I'll just say that it was very impressive, and even more so when I was told that it was produced a couple of years ago. Io's hardware and software have both moved on considerably since then.

The Boards

Pluto 1A

I was then shown the boards, and other hardware, currently in production. Pluto 1 is no longer available, but Pluto 1A is. This is not a dramatic re-design, but basically a re-engineering job, with a number of minor improvements. Additionally, those who have seen a Pluto 1 board could not fail to agree that the Pluto 1A board LOOKS much better, and much more professional - in keeping with its price. It is now a multi-layer PCB, and various options are controlled by DIP switches rather than by soldered links.

Pluto 1A is an 8" by 8" 80-BUS board, is controlled by an 8088 processor, produces 16 colours, has 256K of memory, RS232 interface (e.g. for connection of a digitizing pad), and all graphics commands are controlled by an on-board ROM. It is available in two different resolution options, plus the option of an on-board colour palette, letting you choose the 16 displayable colours out of a range of 4096.

Pricing of Io's products may be considered expensive compared to other 80-BUS boards from other manufacturers, but against "the competition" in the computer graphics market, Io claim that they are very reasonable. By the time you get to the end of this article you may feel that Pluto 1A prices are positively give-away! Prices of the Pluto 1A range are:

640 x 576 dots in 16 fixed colours	-	£600
768 x 576 " " " " "	-	£675
640 x 576 " " " colours from 4096	-	£900
768 x 576 " " " " " "	-	£975

Although the Pluto 1A boards (and some of the other Pluto products, as you will see below) are 80-BUS compatible, many people choose to use them with other computers. Consequently a case, with power supply, may be bought for Pluto 1A, with various interface boards available for connecting to these other computers.

Pluto 1 case with power supply - £295

Interface cards:

IBM PC,XT,AT,clones	- £85	RML-Nimbus	- £85
Sirius	- £85	Q-BUS	- £450
Apricot	- £85	BBC, VME, VAX	- POA

Pluto 2

Pluto 2 is Io's best seller. Still an 80-BUS compatible board, it has grown to 12" x 8" to contain all of the electronics. It retains the same 8088 processor and RS232 interface as Pluto 1A, but includes 512K of memory (optionally 1M), has hardware pan and zoom, 256 colours from a palette of 16.7 MILLION (!), and optional real time frame grabber. This latter option is very impressive, as you would believe that the colour camera was connected directly to the colour monitor, not going into and out of Pluto's memory. Pricing is equally impressive!

768 x 576 dots in 256 colours from 16.7M	-	£2500
Options - Frame grabber	-	£300
Memory expansion (gives 2 screens of 768x576, or virtual 768x1152)	-	£950
Case, including power supplies, fans, video amplifiers, etc	-	£500

Pluto 2 - 24 bit system

Only being able to display 256 colours at a time on the screen is a severe limitation to many people. At that level it isn't possible to have very smooth shading from one colour to another, and therefore subtle toning and other such effects are just not possible. Io therefore sell a system based on three Pluto 2 boards synchronised together. This means that any screen pixel can be any of the 16.7 MILLION colours available.

24-bit system, in case with power supplies etc.	-	£8750
Options - 24-bit colour frame grabber	-	£900
1.5MB Memory expansion (2 screen memory)	-	£2850

Pluto Megares

Pluto Megares is aimed at CAD applications. It is a 10"x8" 80-BUS board and is available in two versions. The first uses a non-interlaced display to give flicker-free output; the second provides much higher resolution than the other 80-BUS Plutos already covered. A palette is fitted to these boards, and they also feature hardware pan and zoom and RS232 interface.

Non-interlaced, 768x576, 16 colours from 4096	-	£2050
Interlaced, 1024x768, " " " "	-	£2150
Case, including power supplies, fans, etc	-	£500

Non 80-BUS

All of the boards mentioned so far are 80-BUS compatible, although they may all be used with a wide range of other computers, as described under "Pluto 1A".

Io also produced a Pluto 1 equivalent that fitted internally in the Sirius, although I believe that this is no longer in production. On top of all this, they also manufacture two ranges of Pluto boards for IBM PCs and compatibles.

Pluto IBM Range

This board is a standard IBM format board for use in PC, XT, AT and compatibles. The usual Pluto features and options are included in the range:

768x576,	interlaced,	16 fixed colours,	1 screen	- £950
768x576,	non-interlaced,	8 fixed colours,	2 screens	- £1250
1024x768,	interlaced,	8 fixed colours,	1 screen	- £1250
768x576,	interlaced,	16 fixed colours,	4 screens	- £1200
768x576,	non-interlaced,	8 fixed colours,	8 screens	- £1500
1024x768,	interlaced,	8 fixed colours,	4 screens	- £1500

On-board Processors

So far all of the boards described have had on-board processors and control ROMs. This approach has allowed Io to give all of the boards the same base command set, and at the same time has allowed them to use different on-board CPUs and graphics display controllers. For example, Pluto 1 and 1A use an 8088 plus 6845, Pluto 2 an 8088 plus NEC7220, and Pluto IBM uses a 68000 CPU. Despite these hardware differences this technique means that many programs written to drive one Pluto model will also drive other models, and means that programmers don't have to totally relearn commands, just modifying their programs to take advantage of more colours, more screens, or whatever.

The arrival of the Hitachi HD63484 ACRTC graphics processor caused a major re-think of this approach. This chip is much more powerful than the other graphics chips so far mentioned, and to add a CPU plus ROM between this and the host system could prove very limiting. So the new Pluto Pennant range, based on the 63484, have this chip interfaced directly to the bus.

(Incidental information: The Gemini 68K-BUS graphics boards also use the 63484, again directly interfaced to the bus.)

To maintain software compatibility with other Pluto boards, Io have written the PGI (Pluto Graphics Interface) driver. Basically this accepts the standard Pluto command set, and converts it as appropriate to drive the actual board attached. Again, this means that as far as the programmer is concerned there is consistency across the entire Pluto range, a very attractive feature when you have spent man-months writing software for a board that gets replaced by another!

Pluto Pennant

As already mentioned, this board range is for PC, XT, AT, and compatibles. All provide flicker-free non-interlaced displays. Hardware pan and zoom, colour palette, and DMA access are supported.

1024x768,	16 colours from 4096,	2 screens	-	£1650
1024x768,	256 colours from 16.7M,	1 screen	-	£1650
1280x960,	16 colours from 4096,	1 screen	-	£1650
1280x1024,	16 colours from 4096,	1 screen	-	£1650

Software Demo

So far I've just run through the various hardware available. (Oh, I should add that Io also supply colour monitors, video cameras, film recorders, printers, etc.) But all this fancy hardware is obviously of little use without software. Many OEM customers (Original Equipment Manufacturers) take Pluto products, and write their own software for their own specific applications. But Io also have various software packages available, and I was shown one of these.

Designer Paint

This is "a sophisticated art and design package aimed at almost everyone in the business of visual arts". Like many packages of this type it uses two monitors. The (usually monochrome) display of the host computer is used to allow entry of

text, display various error messages, show disk files available, etc, whilst the Pluto monitor displays the picture being created. A digitizer tablet is used to control the drawing process. Moving the pen so that the cursor goes off the right hand edge of the display results in a menu appearing down the right hand side of the colour display. Then, moving the pen up and down highlights the various options, and pushing the pen down selects the chosen option. Various menus can appear, depending on what you are doing.

This method of operation allows for fairly rapid working after only a short familiarisation time.

A detailed review of this package could run to many pages, plus it is almost impossible to describe in words what I saw, so I shall just outline some of the options and possibilities.

This package was demonstrated to me on a Pluto 24 bit system with two screen memory and full colour frame grab facilities. A colour photograph was put under the camera and "grabbed". By grabbing into one screen memory and working in the other, a new picture can be built up using operator drawn graphics plus portions taken of the "grab".

The Distort command is fascinating. Any area of a picture may be selected and copied to any other area. However, the two areas selected may be of different shapes, and the software will distort the original area to fit into the new one. As an example of this, various portions of the grabbed image were copied/distorted to form a three dimensional cube, with pictures on each of the three visible faces.

Portions of a picture may be picked up and rotated. Circles, lines, boxes, ellipses, etc may all be drawn. Brushes may be selected for free-hand painting, where the brush may be of any colour and size. More fascinating, the brush may actually be a portion of a picture.

Text facilities are comprehensive. Various fonts (type styles) may be selected and positioned. If required they can have a drop shadow (i.e. the text is repeated again just behind the original text and slightly offset, in any direction, by any amount and in any selected colour, so that the original text appears to be offset from the "page" with its shadow behind). Box text will take any entered text and stretch/shrink it as necessary so that it exactly fits a box that has been drawn on the screen.

The whole thing is extremely impressive. When a drawing is finished it can obviously be saved to disk. From this it is then possible to have slides or photographs produced, or even dump it to a printer. Io have a Mitsubishi colour thermal printer that can produce A3 colour prints on paper or acetate, and around the show-room they have framed examples of this.

The system demonstrated was obviously using all top of the range items and options. Prices of all this technology follow, but starting prices can be much lower, and of course all of the items I was demonstrated aren't essential.

Pluto 24 bit system with frame grabber and memory expansion options, £12500; Designer Paint for Pluto 24 bit system, £2500; Mitsubishi printer driver software, £250; Mitsubishi A3 printer, £6470; Digitizing tablet, £750; 20" colour monitor, £1164; colour camera, £4510; camera stand and lights, £500. Further software options also available (there is a 3-D version of Designer for £3995). (P.S. don't forget you need a host computer too!) Total c. £30000 (+ VAT).

For the not so well off

Arctic Computers of Wetherby, a long established Gemini dealer, and now a main Io Research one also, have realised that many people would like some or all of these facilities, but do not necessarily have the throughput to warrant purchasing them. They have therefore just completed setting up a bureau, with full facilities for producing slides, videos, colour prints, etc, etc. People

may go there and rent time on any of the equipment, or may send disk files in, for processing by Arctic to the desired form of output. Arctic will also supply any of the relevant equipment.

The Company and its Market

From the above it is fairly obvious that Io doesn't sell just to the enthusiastic amateur any more, although some still buy Pluto 1A, which you will now all agree IS at give-away prices!

As I said earlier, Io sell to OEMs who often produce their own software, and until relatively recently Io did not really bother with software. However, they now realise that many totally diverse applications have many common requirements, and they have a number of programmers who specialise in producing customised versions of Designer for these different markets.

Italy has seen the light, and have 300-400 Pluto 2 installations in the textile industry. Here they are supplied, for example, by a knitting machine manufacturer to design stockings, tights and socks. By connecting the Pluto/host computer directly to the knitting machine, once a design is completed a sample sock can appear in 7 SECONDS! Alternatively, the pattern can be dumped to cassettes, which can then be placed into the other knitting machines on the shop floor.

Another use is in carpet design. The complete Pluto system/software in this application is about £20,000. An American competitor offers a similar product for £120,000. You see, Io are cheap!

Further systems are in use in floor tile, wallpaper, and kitchen unit design. Io have also recently become involved in a major publishing project.

The Future

Io have certainly come a long way since their original Nascom add-ons. And as to the future? Well, they say that they have defined 16 specific market areas (e.g. video, textiles, etc) and will be pursuing most of these. And their hardware and software products will continue to move up-market, to provide higher resolutions and higher performance.

Listing continued from page 36.

```
ld      a,(4)           ;get last selected drive
ld      c,a             ;store in 'C'
and     0fh             ;strip user off
cp      4               ;test for drives A-D
ret     c               ;ok, one of these
ld      c,0             ;else set to drive A
ret

        Not standard 56K system so boot off
        disk as usual - a little bit slower
        than off EPROM !
trumb:  ld      a,(0004h)       ;? drives A-D
        and     0fh             ;strip user
        cp      4               ;drive E ?
        jr      nc,gotoa        ;invalid drive
omitc:  cp      2               ;? drive C
        jr      nz,gotoit       ;valid drive A or B
gotoa:  xor     a               ;set to drive A
        ld      (4),a           ;store it
gotoit: ld      hl,wbret        ;GM803 switch off code
        ld      de,00020h       ;empty space
        ld      bc,wbrete-wbret ;code length
        ldir                    ;copy it down
        ld      de,0f000h       ;don't ask me why ?
        ld      hl,0020h        ;for later jump
        ld      a,(0004)        ;set selected drive
        ld      c,a             ;in C register
        jp      (hl)            ;JP to RAM exit

;       RAM exits for return to warm boot and ccp
wbret:  xor     a               ;page off and
        out     (PMODE),a       ;Jp to 0000h
        jp      0               ;warm boot as usual
wbrete  equ     $

copy:   xor     a               ;page off and
        out     (PMODE),a       ;jump to ccp
        jp      zcpr            ;as cp/m pre-fetched
copye   equ     $

;       Code for Nascom system
reset:: LD      SP,01000h
        LD      HL,09000h       ;LOAD N3 INTO RAM @ 09H
        LD      DE,0
        LD      BC,800H
        LDIR
        CALL    00000h
        . .                     ***********************
```

Upgrading a GM813 to 256K by W.H. Turner

Introduction

This article describes how to add more memory to a Gemini GM813 CPU board and what to do with it.

There was not a single good reason for wanting to put more memory on my Gemini Galaxy 2 computer! Instead there were several vague possibilities - the possibility of using it for a RAM disk, the possibility of using it for a disk cache (as in CP/M Plus), and the possibility of adding a GM888 board (8088 co-processor), that made me do it.

Whatever the reason, the additional memory boards looked very expensive and used valuable bus slots which made the whole exercise a lot less attractive. My solution was to replace the existing 64k bit RAMs with the pin compatible 256k bit RAMs which quadrupled the memory for an outlay of only £55.

The GM813 board, at the heart of the Gemini Galaxy series of computers, is able to address up to 512k bytes of memory via 19 address lines but only uses 16 to address the 64k bytes of onboard memory. Access to the additional address lines is controlled by two 74LS189 TTL RAMs used as sixteen 8 bit mapping registers between the top four Z80 address lines and the top seven bus address lines. Each mapping register controls a 4k byte block of the 64k byte Z80 address space. Each address generated by the Z80 selects a 4k byte block of memory via one of these registers and the 8-bit value written into the mapping register determines which 4k block out of the 512k bytes is actually accessed. The result is that any 4k byte block of memory can be mapped at any 4k byte boundary in the 64k byte address space of the Z80. In passing I should mention that the bus signal RSFU (reserved for future use) can be redefined and connected as A20 giving a colossal megabyte of address space should anyone ever find a use for it.

I first became interested in extending my memory when everyone else was enjoying RAM disks except me. But the cost of buying an additional memory board and alternatively, the complexity of producing my own board, meant that things didn't get very far. It was only after an engineering friend suggested replacing the existing 64k bit RAMs on the processor board with 256k bit RAMs that I began to look at the problem more seriously.

For a long time I procrastinated, believing that if I attempted even minor surgery on the processor board I could end up with a very dead computer. Eventually I took a firm grip on myself and removed the GM813 board from the computer to look at the possibilities, besides at that point I had already bought the 256k bit chips! Looking at the board and the circuit diagrams, so thoughtfully supplied by Gemini I worked out a plan of action.

Not being very imaginative when it comes to hardware I felt that I should, as far as possible, duplicate the circuits already in use on the processor board to address the additional memory. Further, by mounting the additional chips directly above the existing chips, the new ones could be easily connected to the power supply and the necessary signals.

The modifications were to be done in a series of short steps after each of which the board would be slid back into its slot in the card cage, the cables reconnected and the system booted back up to check that it still worked. If it suddenly stopped working then I would only have to check the changes performed in the last stage, looking for pins not inserted in sockets, dry joints, solder bridges etc. Further I would not cut any tracks so everything could hopefully be returned to its original state if necessary.

The Theory

The theory starts with the 256k chips being pin compatible with 64k ones except that pin 1 is now used as A8 (the additional address line necessary to address all the extra memory). By connecting this A8 via a multiplexer to A16 and A17 it should be possible to address the whole of the 256k. It doesn't matter that A8 on the RAM chip should be multiplexed between A8 and A17 bus address lines and all the other address lines moved up to match as it is not necessary to use consecutive cells in the RAM, the only requirement being to ensure that the same location is addressed consistently.

Looking at the circuit diagram revealed that I would still only be able to use the lower 64k as the chips are only selected when the top three address lines are all zero. Working backwards from the PAL, through which all onboard addressing is done, the onboard RAM is selected via AE0 (pin 18) which is high whenever A16 - A18 are all low (i.e. the address is in the lower 64k bytes). To select the onboard RAM for any address within the lower 256k bytes, AE0 has to go high regardless of the state of A16 and A17 so it becomes the inverse of A18. Unfortunately the signal AE0 is also used to determine when the boot EPROM is selected as this is expected to be at 0F000H - 0FFFFH. The way round this is to sever the connection between A16 - A18 and AE0, leaving A16 - A18 to be used to select the EPROM and take A18 via an inverter to AE0 on the PAL to select the onboard RAM.

The Modifications

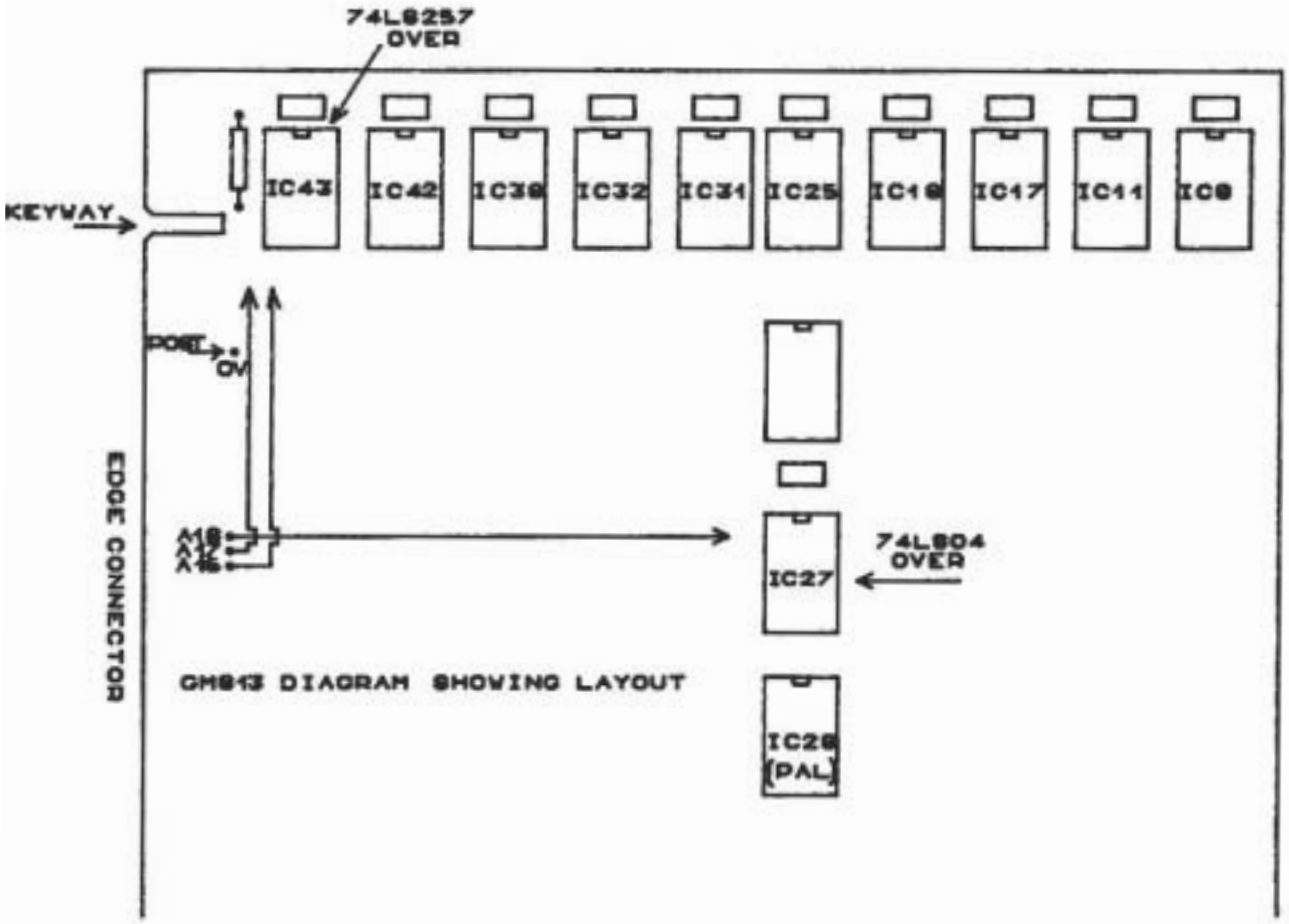
Now to the actual modifications (refer to diagram). The first step is to remove the eight 64k bit RAMs from their sockets (ICs 8,11,17,18,25,31,32,38). Very carefully take the eight 256k bit RAMs and bend up pin 1 on them all. Then insert them into the sockets making sure that they are the correct way round. On my board, manufactured in 1983, all the RAM chips were socketed making this step very easy. However, photographs in the current Gemini catalogue show the RAM chips soldered directly into the board which means a lot more work for some people. Wire all the lifted pins together and to earth which I found at a post near IC43. Then check that the board still works before proceeding.

The second step is to take a new 74LS257A quad multiplexer chip and bend up pins 2 - 7 and 9 - 14. Then solder it directly above IC 43, also a 74LS257A, with the same orientation so that it can pick up power and some control signals from it (i.e. pins 1, 8, 15 and 16 are all soldered onto the same pins on the chip below). Unsolder the wire from the earth pin connected in step two and solder it to pin 9 of the new 74LS257A. Connect pin 10 to A16 which can be conveniently found near the edge connector, and pin 11 to A17 which is next to A16. Now check the board before proceeding.

The last hardware modification is to enable all the memory to be addressed. Take a new 74LS04 hex inverter chip and bend up pins 1 - 6 and 8 - 13, then solder it above IC 27 (the same way round) connecting the power pins 7 and 14 to the chip below. Solder pin 13 to A18 (next to A17 near the edge connector). Carefully remove the PAL chip, IC 28, and bend up pin 18 and replace it in its socket. Connect this pin to pin 12 on the new 74LS04 and check that the board still works before finally closing up the case.

Software

When it comes to using the mapping registers, these can be altered by use of the Z80 OUT instruction "out (c),r" where the memory mapping port (0feh) is in the "c" register and during execution, the "b" register contents are also placed on the address bus where the most significant four bits select the mapping register. The "r" register contains the 4k block number 0 - 07fh which is written into the mapping register. The boot EPROM ensures that the mapping registers are initially set up to map the bottom 64k bytes in a one to one mapping (i.e. the registers contain 00h, 01h, 02h, up to 0fh).



CM813 DIAGRAM SHOWING LAYOUT

The test program (listing no. 1) is not an exhaustive test but it does verify that the mapping works. In it all the forty eight 4k blocks from addresses 10000H to 3FFFFH are mapped into the 4k block starting at 1000H and the block number is written into the first byte of the block. The process is then repeated mapping the blocks back and checking that the first byte is still what had been written. It outputs 'Success' or 'Failed' depending on the result. I didn't have any problems so I haven't bothered to extend the program to explore which blocks failed.

Having all this additional memory, what did I do with it and was it worth it? The first thing I discovered was that it was no use for programs as there is no way of linking sections of code or data at a particular address. Therefore the first thing I did with the extra RAM was to use it as a 192k byte RAM disk (see listing no. 2). The sector size was made 128 bytes to avoid the need for deblocking and the track size was made 4k bytes so that the track number for a read or write gave me the 4k block number to map.

To make full use of it I changed SUBMIT and the CCP to use the RAM disk for the "$$$.SUB" file. Then I reduced the disk size by 4k and reloaded the CCP and BDOS from it at each warm boot. Finally I patched Wordstar to look on drive M: for its overlay files. What an improvement (provided I remembered to copy them across first), particularly the speed up in editing. But I do regret that the disk will never be big enough to use as a source disk.

For some time I considered the possibility of writing disk cache software to use the memory more efficiently, but eventually opted to try a banked version of CP/M Plus which already includes all the disk cache software. This meant that I had to allocate 4k blocks of memory to banks, a 4k block to common memory and then write my own bank switching routine (listing no. 3).

The secret of banked CP/M Plus is to make the bank switching software as efficient as possible as a lot of bank switching goes on, but even though I have to map 8 blocks on every bank switch, there is no visible degradation. Of the memory not allocated to banks or common, one block is used to reload CCP from on warm starts and the rest is a RAM disk again. The disk cache does speed things up a bit, but making drive M: the temporary disk and making CCP search drive M: first for any programs is even better!

Conclusions

Was it all worth it? In terms of size, the RAM disk I have created is too small to use for program development and I may still buy the Gemini 2Mb RAM disk when it appears, but on the other hand using some of the additional memory to run a banked CP/M Plus does seem to produce the system I am looking for! Now I've added a GM888 board, and CP/M-86 comes up with a TPA of 243k, but that's another story!

Listing 1

```
;
;       banked memory tester for GM813
;
        .z80
;
start:  ld      hl,1000h        ;start of free 4k block
        ld      a,10h           ;start at 10000h
        ld      b,48            ;number of 4k blocks
wloop:  push    bc              ;save
        ld      b,10h           ;mapping reg 1
        ld      c,0feh          ;port
        out     (c),a           ;do it
        pop     bc              ;
        ld      (hl),b          ;save number
        inc     a               ;next 4k block
        djnz    wloop           ;repeat
;
        ld      b,48            ;number of 4k blocks
        ld      c,0             ;success count
        ld      a,10h           ;start at 10000h
rloop:  push    bc              ;save
        ld      b,10h           ;mapping reg 1
        ld      c,0feh          ;port
        out     (c),a           ;do it
        pop     bc              ;
        push    af              ;save a for compare
        ld      a,(hl)          ;recover number
        cp      b               ;save as written?
        jr      nz,bad          ;if not
        inc     c               ;another good block
bad:    pop     af              ;restore a
        inc     a               ;next 4k block
        djnz    rloop           ;repeat
;
        push    bc              ;save c
        ld      a,01h           ;restore to 1000h
        ld      b,10h           ;mapping reg 1
        ld      c,0feh          ;port
        out     (c),a           ;do it
        pop     bc              ;restore c
;
        ld      a,c             ;count of good blocks
        cp      48              ;number of 4k blocks
        ld      de,success
        jr      z,print
        ld      de,failed
print:  ld      c,09h
        jp      0005h
;
success:defb    'Success$'
failed: defb    'Failed$'
;
        end     start
```

Listing 2

```
;
; Additional code for BIOS to support RAM disc on GM813
;
mmp     equ     0feh            ;memory management port
;*****************************************************************
;
; dph for memory drive
;
drivem: defw    0               ;no sector translation
        defw    0               ;
        defw    0               ;
        defw    0               ;
        defw    dirbuf          ;common directory buffer
        defw    dpbm            ;dpb address
        defw    0               ;no CSV as permanently mounted
        defw    alv12           ;allocation vector ((192/8)+1 bytes)
dpbm:   defw    32              ;sectors per track (128 byte sectors)
        defb    03              ;bls=1024 thus  bsh
        defb    07h             ;               blm
        defb    0               ;               exm
        defw    191             ;192k drive
        defw    30              ;only 31 directory entries
        defb    80h             ;al0
        defb    00h             ;al1
        defw    0               ;cks=0 (permanently mounted)
        defw    0               ;no reserved tracks
;*****************************************************************
;
; Select track zero
;
home:   ld      a,(sekdsk)      ;memory disc?
        cp      80h             ;test
        jr      z,home1         ;if yes no need to use host buffer
;*****************************************************************
;
; Select disk (number in 'c')
;
seldsk: ld      hl,0000h        ;ready to return 0000
        ld      a,c             ;get drive number
        cp      'M'-'A'         ;drive M:?
        jr      nz,sel1         ;if not
        ld      a,80h           ;show it is memory disc
        ld      (sekdsk),a      ;store (for read/write)
        ld      hl,drivem       ;return DPH for drive M:
        ret                     ;return
```

```
;*****************************************************************
;
; READ - Read sector from disk
;
read:   call    swstk           ;switch to bios stack
        ld      a,(sekdsk)      ;check if read on drive M:
        cp      80h             ;test
        jp      z,mread         ;perform read on drive M:
;*****************************************************************
;
; WRITE - Write sector to disk
;
write:  call    swstk           ;switch to bios stack
        ld      a,(sekdsk)      ;check if write on drive M:
        cp      80h             ;test
        jp      z,mwrite        ;perform write on drive M:
;*****************************************************************
;
; Memory disc access routines
;
mread:  ld      a,1             ;indicate read operation
        jr      mcom            ;common code
;
mwrite: xor     a               ;indicate write operation
;
mcom:   ld      (mflags),a      ;save direction of transfer
        ld      hl,1000h        ;assume track mapped at 1000h
        ld      a,(dmaadr+1)    ;get DMA page address
        and     0f0h            ;convert to 4k block number
        cp      h               ;DMA address within 4K block at 1000h
        jr      nz,mcom1        ;if not
        ld      hl,3000h        ;track mapped at 3000h instead
mcom1:  ld      b,h             ;h is mapping register number
        ld      a,(sektrk)      ;this is 4K block to map 0 - 63
        add     a,16            ;step over first 64k
        ld      c,mmp           ;load port address
        out     (c),a           ;map track into memory
        push    bc              ;save to map back
        ld      a,(seksec)      ;get sector in track
        ld      bc,128          ;sector size
        or      a               ;sector 0?
        jr      z,msec0         ;if yes
nxtsec: add     hl,bc           ;bump address to sector
        dec     a               ;count sectors
        jr      nz,nxtsec       ;repeat until done
```

```
msec0:  ld      de,(dmaadr)     ;get dma address
        ld      a,(mflags)      ;get transfer direction
        or      a               ;test
        jr      nz,mcom2        ;if read
        ex      de,hl           ;exchange source and destination for write
mcom2:  ldir                    ;transfer sector
        pop     bc              ;recover mapping register / port number
        ld      a,b             ;get 4k block number in 'a'
        rra                     ;from high nibble
        rra                     ;
        rra                     ;
        rra                     ;
        and     0fh             ;mask to be in range 0 - 15
        out     (c),a           ;restore normal mapping
        xor     a               ;return good status
        ret                     ;
;
mflags: defs    1               ;transfer direction

;*****************************************************************
;
; Initialise memory disc. Check if RAM disc already initialised by
; contents of direcory entry 32 which is not used. In this way the
; disc contents remain unaltered in the event of a reset.
;
initm:  ld      bc,1000h+mmp    ;mapping register 1 / port number
        ld      a,10h           ;4k block at 10000h
        out     (c),a           ;map at 1000h
        ld      a,(13ffh)       ;if already initialised then this is 0e5h
        cp      0e5h            ;test
        jr      z,initm1        ;if correct then already initialised
        ld      hl,1000h        ;fill track with e5h
        ld      (hl),0e5h       ;
        ld      de,1001h        ;
        ld      bc,1023         ;1k-1
        ldir                    ;
initm1: ld      bc,1000h+mmp    ;mapping register 1 / port number
        ld      a,01h           ;4k block at 01000h
        out     (c),a           ;restore original block to 1000h
        ret                     ;and return
```

Listing 3

```
;
; CP/M Plus Bank Switching routine for GM813
;
mmp     equ     0feh            ;memory management port
?bank:
        push    hl              ;save regs
        push    bc              ;
        ld      c,mmp           ;memory management port
        ld      hl,bank0        ;assume bank 0
        or      a               ;test
        jr      z,map           ;must be bank 0
        ld      hl,bank1        ;otherwise must be bank 1
map:    ld      a,(hl)          ;get register contents
        inc     hl              ;point to next byte
        or      a               ;if zero done
        jr      z,map$done      ;exit
        ld      b,(hl)          ;get register number
        inc     hl              ;point to next byte
        out     (c),a           ;map 4k block
        jr      map             ;again
map$done:
        pop     bc              ;restore regs
        pop     hl              ;
        ret
;                               bank 0 consists of:
bank0:  defb    06h,060h        ;06000h-06fffh at 6000h
        defb    07h,070h        ;07000h-07fffh at 7000h
        defb    08h,080h        ;08000h-08fffh at 8000h
        defb    09h,090h        ;09000h-09fffh at 9000h
        defb    0ah,0a0h        ;0a000h-0afffh at a000h
        defb    0bh,0b0h        ;0b000h-0bfffh at b000h
        defb    0ch,0c0h        ;0c000h-0cfffh at c000h
        defb    0dh,0d0h        ;0d000h-0dfffh at d000h
        defb    0eh,0e0h        ;0e000h-0efffh at e000h
        defb    0
;                               bank 1 consists of:
bank1:  defb    10h,060h        ;10000h-10fffh at 6000h
        defb    11h,070h        ;11000h-11fffh at 7000h
        defb    12h,080h        ;12000h-12fffh at 8000h
        defb    13h,090h        ;13000h-13fffh at 9000h
        defb    14h,0a0h        ;14000h-14fffh at a000h
        defb    15h,0b0h        ;15000h-15fffh at b000h
        defb    16h,0c0h        ;16000h-16fffh at c000h
        defb    17h,0d0h        ;17000h-17fffh at d000h
        defb    18h,0e0h        ;18000h-18fffh at e000h
        defb    0
;                               common consists of:
;                               0f000h-0ffffh at f000h
;                               00000h-05fffh is always at 0000h-5fffh
        end
```

Scorpio Systems Goes "Compatible" by P.A. Greenhalgh

Bad to Good

If, a couple of years ago, you had said that at some future date I would be looking at buying an "IBM" you would have received a fairly abusive reply. When IBM first introduced the PC there were, in my opinion, a number of major shortcomings. They were expensive, PC-DOS was fairly ropey, the video left a lot to be desired, there was little software around, and they were very slow. At that particular stage I was using a Gemini Galaxy system with Winchester and RAM-DISK, mainly for WordStar work, and this combination was vastly faster than WordStar on the PC, the software apparently being a "straight-forward" code conversion from CP/M / 8080 to PC-DOS / 8088.

But, by having the very name IBM, this monstrosity almost overnight became a standard. Subsequently developments have raced ahead, with more powerful, faster processors; many improvements to the Operating System; a vast reduction in prices; faster, better and more colourful displays; and huge quantities of very good software.

Why Change ?

There were a number of reasons for deciding to buy an "IBM", but there was one major one:

Here at Scorpio Systems, as well as producing this glorious publication, a lot of time is spent producing technical documentation. A laser printer is used to produce master copy. This printer is capable of very many fancy tricks, and incorporates a large number of different fonts, its most attractive being certain proportional ones. But WordStar does not understand the concept of proportional fonts, and that lines of equal length can contain considerably different numbers of letters. Take the words lilly and mummy for example, and you can quickly appreciate that in a proportional font, where each letter has its own width rather than a fixed width for all letters, the former word occupies a much shorter line length than the latter.

So WordStar can not be sensibly used to take full advantage of the laser's facilities, and it therefore currently tends to be used, in effect, as a very high speed high quality daisy-wheel, with the advantage of there being various fixed-spacing fonts available for various applications. (For example, a fairly small font is used for Scorpio News in order to get as much information as possible into a limited number of pages, whilst still maintaining good legibility. It can go MUCH smaller!)

Having a printer that is as capable as this one, but being so restricted in the number of features that can be used, must be akin to owning a Lambourghini Countach (capable of c. 180mph) and living in Jersey (maximum speed limit 40mph). There doesn't appear to be any CP/M-80 software around to use it to its full, but under MS-DOS there are various programs, particularly Xerox Ventura Publisher, which are capable of some pretty (sic) amazing things.

What To Buy ?

Anyway, having decided that it was time to go "compatible", the big question was with what? When you start to look around the IBM / Clone market place it all very quickly becomes very confusing. 8088, 8086, 80186, 80286, 4.77MHz, 6MHz, 8MHz, 10MHz, 12MHz, 0 wait states, 1 wait state, slow seek, fast seek, 10MB, 20MB, 30MB, 320K, 360K, 720K, 1.2MB, MS-DOS, PC-DOS, 3.5", 5.25", MDA, CGA, EGA, PGA, Hercules ..... these are among the different things that are thrown at you. And that was BEFORE Personal System 2, 80386, and OS/2 started to get in on the act too!

I decided that the best policy was to adopt a "maximum bangs per optimum bucks" philosophy, i.e. to go for the best possible specification prior to prices starting to go through the ceiling. On this basis it very soon became apparent

(in my interpretation, anyway) that this would be an IBM AT clone with EGA (Enhanced Graphics Adaptor) colour board and display. The number of alternatives was still large, but definitely narrowing.

Which AT ?

The next decision is really whether you want a known brand name or not? The less well known, the cheaper the product. At the top of the scale you have IBM itself. Then you move down through Compaq, Epson, Tandon, etc, and eventually arrive at the Ness, Elonex, Packard Bell, etc. Initially I felt that I should go for an Epson or Tandon. They are recognised names in the industry, have large numbers of dealers, and are unlikely to disappear tomorrow (hopefully!).

One day, during my deliberations, I was talking on the 'phone to someone ("Mike"), and mentioned that I was looking at AT-clones. They responded that they had recently purchased several from Elonex, based on the North Circular Road, London, and that they were very pleased with them. This started me wondering whether I should in fact go for one of these lesser known breeds. A bit more digging established that most of the low cost Taiwanese clones are based on one of about three different main boards, and that servicing should therefore not be a particular problem if the chosen supplier suddenly "disappeared". As there is a considerable saving to be made by going this route I became interested.

A Look at Elonex

Several days later I had to go into London, and on my way back called in at Elonex. They occupy about half a dozen rooms in a larger building, and there were boxes of computers everywhere. I was given a demo of their AT clone, that they call the PC-286. Various well-known packages were run to demonstrate its compatibility, all without hitch. It seemed to be well built, and was certainly competitively priced. I commented that I had read that some packages did not like running at the higher clock speeds that many of these clones offer, something that I had never quite understood anyway. The response was that they always ran their systems at 10MHz, 0 wait states, and had no problems.

I decided to go away and consider the situation further, but before leaving asked about the delivery and payment situation. I was told that delivery would be about three weeks. I queried what all the piles of computers were around the rooms, and was told that these were all sold, plus the next shipment that was due in from Taiwan! High demand like this seemed a good sign (unless they were all empty boxes to fool people like me!). On the payment side, this could be made in advance, or a Banker's draft, Building Society cheque, or cash, brought along on the day of collection, if indeed you were collecting yourself. Again this seemed a good sign, as like many people I am very wary about payment in advance to any company, particular a relatively large sum to an unfamiliar one in such a high-risk industry.

As luck would have it an issue of Practical Computing came out a few days later, and carried a comparative review of the Elonex with two other similar machines. The comments were sufficiently favourable to strengthen my interest. A further 'phone call was made back to Mike - what was his opinion now that he had had the machines in use for a few weeks? Very good, no real complaints, other than a keyboard that had stopped working after being given a cup of coffee to drink!

Decision made

I phoned up Elonex and placed my order. Two weeks' delivery I was told, and as I was going to collect they would call when it was ready. Mike had said to give them their stated time, but then to call them as it seemed to prompt swifter reaction. I called after two weeks (a Thursday) and was told it would be available Monday. Along I went, handed over the specified cheque, and left with my purchase, like a kid with a new toy! [Ed.'s wife - yes, a big kid with a big new toy!]

Review of the Elonex PC-286 by P.A. Greenhalgh

Introduction

The Elonex PC-286 is one of the many Taiwanese IBM-AT clones currently available. As now appears to be the norm, it is actually faster than the IBM, as well as a great deal cheaper. Here follows a review of this system. It should be noted that, unlike most magazine reviewers, I have actually shelled out hard cash for this! Consequently the very fact that I decided to buy this machine over all the others must mean that I must think it is pretty good. However, I shall try to remain impartial!

Specification

The specification is:

* Intel 80286 processor, software switchable between 6 MHz, 8 Mhz and 10 MHz. 0 or 1 wait state, also software switchable.
* Socket for 80287 maths co-processor
* 640K bytes RAM
* 32K bytes ROM BIOS
* 20MByte Winchester hard disk drive (65mSec access)
* 1.2 MByte floppy disk drive (can also read/write 360K disks)
* Battery backed Real Time Clock/Calendar
* MS-DOS 3.2
* Two RS-232 serial interface
* One parallel printer interface
* 6 16-bit expansion slots, 2 8-bit expansion slots (4 occupied)
* 200W Power Supply with fan
* "AT-style" Keyboard

* Hercules compatible monochrome text/graphics card
* 14" monochrome (green) monitor

The above is priced at a very competitive £1295 (+ VAT).

Options

There are a variety of options available, including various larger Winchester sizes. The ones that I took were:

* EGA compatible colour display card
* 14" colour monitor
* Additional floppy disk drive (360K)

The EGA board/colour monitor is available, in place of the Hercules board/monochrome monitor, for an additional £395 (+ VAT). The additional floppy disk drive is £80 (+ VAT).

Total price of the system reviewed here, therefore, was £1770 (+ VAT).

Unpacking

On getting the system to the office I unpacked it. The monitor box contained purely that, plus manual. The computer box contained the computer, keyboard, manuals, master disks and relevant connecting cables. The only additional item required to hook it all up and get it running was a mains plug.

Appearance

The Elonex is a reasonably attractive clone, finished in a cream colour. The front panel has space for two floppy drives, both filled on my system. The standard system has just a 1.2M drive, but I decided to also order the 360K drive. The reason for this was that 360K disks created on the 1.2M drive cannot always be read by systems with just 360K drives. I also reasoned that by

ordering this extra drive at the same time as the system, it too would be in beige, like the 1.2M drive. Well, I was wrong, it was a black drive! When I queried this with Elonex their sales-style reply (i.e. making a "feature" out of a "bug") was that "many people prefer it as they then know the drives are different". Well, I'm not one of those people!

There are two recesses in the front panel. One contains a reset button. This is a very useful facility; many clones have no reset, and if you somehow manage to totally hang the system you have to resort to powering it off and back on again. The second recess, immediately above the first, contains a power-on indicator, Winchester access indicator, CPU speed indicator, and key-switch. The CPU speed indicator is off for 6MHz, green for 8MHz, and red for 10MHz. The keyswitch, when locked, disables the keyboard and physically locks the case lid to the system. The logic to the keyboard disabling is that (a) you can go away from your system in the middle of some work and no-one can disrupt it, or (b) no-one can access your private files. Presumably locking the lid on is to prevent people from stealing boards from inside your system - but they could still take the whole thing!

The only other items on the front are an "Elonex" label, and some designer grooves.

Towards the rear on the right hand side is the power-on switch. On the rear of the system itself is the mains-in socket, as well as a mains-out socket. This latter socket is intended for the connection of mains to the display monitor, and is controlled by the main system's on/off switch so that you have to throw just the one switch to power the whole thing on and off. Just above the mains sockets is the cooling fan vent. The only other thing on the rear, other than access to the rear of the expansion boards, is the keyboard socket.

There is a rear panel for each of the eight expansion slots. On one (the colour board) is a 9-pin D-type, for connecting to the display monitor, and a DB-25(F) for the parallel port. There are two other panels "occupied", each with a DB-25(M) for the serial ports.

Inside

I had no intention to go inside the unit until two things happened:

1) I decided to try out the key-switch. It turned part way round, and then met some resistance. I assumed that it wasn't mating up too precisely with the catch to lock the lid on, and turned it a little harder. The resistance disappeared. I later discovered that the Reset button no longer functioned, put 2+2 together, and realised these things may well be associated with one another!

2) The Norton Utility programs include one called SYSINFO, that gives you various information about your system, like how much memory it has, plus the results of a timing test. The timing test is supposed to indicate the number of times faster the system is than a standard IBM PC. I know various people dispute the actual figures given, but I also know that a 10MHz, 0 wait state system should give a result of approx. 11.5. My system said 10.3, regardless of whether I software switched the wait states on or off. Referring to the manual I discovered that there is a link block inside the system that can be set to select either (i) always one wait state, or (ii) software selectable zero or one wait state.

I therefore decided to dive into the system. To open it just requires the removal of five screws on the rear panel (four on mine, one was missing). Then the whole lid and front panel slides off to the front, with the drives and recessed items described earlier left behind with the main chassis.

I must say that the whole thing seems to be very well built, with excellent access to everything. To remove any of the expansion boards requires the release of just one screw per board. To remove a disk drive requires the release of two screws per drive at the front, disconnecting the power and data cables, and then the drive just slides out to the front on runners.

The assembly contains the reset switch, indicators, key switch, etc was also released by two screws. I then discovered that my suspicions were correct. One of the cables to the reset switch was too long, and had got in the way of the keyswitch. When I had turned the latter I had ripped off the former! A quick dab of the soldering iron and all was well.

I was also right about the wait-state link, and changed that to software-selectable.

I noted that the 1.2M floppy drive was a Teac FD55FG unit, and that the Winchester was a half-height Seagate unit. No un-heard of Taiwanese rubbish here! Further, the unit was all equipped to easily take a second Winchester, the power cable sitting there waiting.

The main board on the base of the unit contains the 640K RAM (18 x 256K chips, plus 18 x 64K - 18? Yes, parity RAM.), the 80286, socket for 80287, real time clock/calendar and battery, keyboard support, and general logic. Like most AT boards, the CHIP's chip set is used to provide most of the general purpose logic.

There are three "expansion" boards fitted. A full length Winchester / floppy controller board; a full length EGA colour board, also containing the parallel port; and a small dual serial-board, one DB25 being on the same board, plus a cable going to the second DB25, mounted on one of the rear expansion panels corresponding to an unused expansion slot.

The Monitor

The monitor is a 14" TVM one (although it actually has a "Copam" label on the front). It is a dual-mode one, switching automatically between CGA and EGA modes as required by the graphics board. On the rear is the mains-in socket, plus a cable going to a 9-way D-type for plugging into the EGA board. There are also four pots, two "Vertical size" and two "Horizontal Centre", for adjusting the display in each of its modes.

On the front is an on/off switch, power-on indicator, brightness control, and colour balance control. There is also a three position Display Mode switch that gives a full-colour picture, or a green only or amber only one. I can't really consider this to be an awful lot of use, as it is easy to software select text of your own colour choice anyway. Unless anyone else knows of a good reason? I bet some people can't work out where their colour has gone, or don't even know they've got it!

The Keyboard

The keyboard is "AT-style", whatever that means. It has three groups of keys. On the left are 10 function keys, then there is the main alpha-numeric and control cluster, and on the right is the numeric pad, screen control cluster. The main cluster is slightly non-standard in that it has the four cursor keys repeated in a straight line at the bottom right hand corner.

There are three LEDs to indicate Caps Lock, Num Lock and Scroll Lock. A couple of swing-out plastic legs under the keyboard can be used to tilt it, and this is the way that I find it most comfortable.

Key action is a la IBM, in that each key contains a "click" mechanism. It's not a bad keyboard at all.

The Manual

An A5 ring binder containing the documentation comes with the system. As seems to be happening quite often with both hardware and software these days, there is JUST too much manual for the binder, making it awkward to use unless you remove one of the smaller sections.

In the binder, as supplied, there are several sections:

EGA Users' Manual
Serial Card User's Guide
PC-286 Turbo Personal Computer User's Guide
MS-DOS 3.2 Operating System User's Guide
MS-DOS 3.2 Operating System User's Reference

Plus you could TRY to add the "Colour Monitor Operating Instructions" manual into the binder if you wish, as it is also A5 and is correctly punched.

I have to confess that I have not read the manual in any great detail yet. I have, however, managed to find everything that I have wanted to know so far. I have shown the manual to a few people who have bought various other clones, and the general reaction has been very favourable, as some of these other systems have apparently come with next to nothing. However, as I have said, I cannot really vouch for the quality yet. Here are a few impressions:

The EGA Manual doesn't appear to be very good at all. There are various link and switch options on the board, and the manual shows you where they should be in normal use. However, in general it doesn't say why or when you may wish to alter them.

The Serial Card Manual is better, although still rather terse. It does have the advantage of containing a circuit diagram, and as it's a very simple board, this could be a good way of confirming that you've interpreted switch positions, etc, correctly.

The PC-286 Manual is better still. It too comes with full circuit diagrams. Various drawings in it of the system show that the front panel has had things moved around, but this shouldn't matter. The manual does NOT assume that you have a Winchester, but that you may have either one or two floppy drives only. Therefore various sections are more complicated than they need be, as it describes loading programs from the master floppy disk, then removing this disk and inserting another, etc.

I have so far spotted two errors in this manual - one gave me a good chuckle, but other readers may not have been so impressed if they actually followed the instructions! The section went:

> Formatting Using Two Drives
>
> If you have two disk drives, follow the steps below:
>
> 1. Insert the DOS diskette into Drive A. (Ed. - your MASTER disk)
> 2. Insert the new diskette into Drive B.
> 3. Type:
>
> A> FORMAT A: [enter] (Ed. - your MASTER disk!)

The other error is in the section on the wait-state link. It refers to a diagram, but the diagram shown is the wrong one and is to do with connecting an external battery. I then used the circuit diagrams to work out where the correct link was. I later discovered that there was a drawing of the main board at the very beginning of the manual that showed the correct link anyway.

The MS-DOS manuals actually have "Microsoft" on the cover, and inside "Printed in Taiwan under the licence from Microsoft Corporation". Between them they are a good couple of inches thick, and I have yet to wade through them, but they do look fairly comprehensive. Unlike the other manuals, though, they do not appear to suffer from any Taiwanese English.

Using the system

As stated earlier, there is a program available that purports to show the speed of the system relative to the PC. Once the wait-states had been made software switchable, I got the following results:

6MHz, 1 wait state	-	5.7
6MHz, 0 wait states	-	7.0
8MHz, 1 wait state	-	7.7
8MHz, 0 wait states	-	9.2
10MHz, 1 wait state	-	10.2
10MHz, 0 wait states	-	11.5

A program supplied with the system, SPEED, was used to switch between these. You can also switch clock speeds directly from the keyboard, but not wait states. The default power-on state is set via SETUP, which stores your choice into battery backed memory.

In general I can see of no reason why you should want to switch clock speeds, and normally leave the system at 10MHz. However, a few of the programs that I have are games, and these are far too fast at 6MHz, let alone at 10! It is in fact interesting to note that some of the games don't seem to alter, even when you change the clock speed. Presumably they are using one of the programmable timers that are a standard part of the AT spec.? As far as I am aware the only time you HAVE to change clock speed is with some clever formatting and backup programs that drive various chips directly, and have very specific and critical timing loops in the code. As I don't seem to have any of these programs this doesn't worry me.

I tried a variety of software on the system, and it all ran without problem, with the exception of BASIC! By trial and error I discovered that BASIC was fine at all speeds except 10MHz, 0 wait states. I started to wonder if BASIC was one of these programs that falls over at high speeds - I couldn't understand how it could be for a moment, but that was how it looked. What would happen was that a BASIC program that would run fine at 10MHz, 1 wait state (from now on 10/1, etc), would give various syntax errors at 10/0. I went with my system to see Dave Hunt. These funny programs ran at 10/0 on his system - aha, a hardware fault on mine! I then said "Look how they fall over on mine". They didn't! Hmmm! Then several minutes later, at 10/0, they started to fail. A temperature dependent fault, temporarily cured by a cold spell in the car? OK, let's leave the system on for a bit longer. Lo and behold, the same problem at 10/1.

Back to Elonex

I took the system back to Elonex, and they disappeared into a back room. When they reappeared they said "changed some chips - try that". I did, leaving it on for some time with no problem. Then, running Norton, I noticed that it was back at 10/1 again - they'd reset the link! I queried this, and was told that all the initial systems that they had had had run at 10/0 OK. But then they had started to get problems, so they were all now sold set to 1 wait state.

I pointed out that the advert said 0 wait states, and was told "yes it will run with 0 wait states, but not at 10 MHz. We don't say it will do both at the same time". For some reason I let this pass, but have subsequently decided that I may raise this again. After all, the SETUP program allows 10/0 to be set as a default, and the manual makes various references to it. In fact, I have subsequently met someone else who has recently bought an Elonex system and he has noticed a Norton of only 10.3, yet his SETUP, AS DELIVERED, shows it is set to 10/0. Obviously the system is supposed to be capable of it, but I expect you'll see Elonex's adverts changed in the near future.

At the time of writing, I don't think you'll find another 10MHz system at the same price, in fact I'm not aware of even an 8MHz system that's within £100, and so I suppose it can be argued that one shouldn't quibble about the lack of this last bit of possible performance. If you are looking for a system with that extra edge, do check carefully. Some suppliers make no comment at all about wait states, and more often than not this means that the system has them permanently set. Of course, if you're wealthy, there's always the 12MHz systems which have recently become available, and if you're absolutely stinking rich you can get an 80386 based system!

Colour Display

Having used a monochrome Gemini system for many years, I felt that there would be little or no benefit in having a colour display as far as most programs are concerned. I have now changed my mind, I think! The reason for my hesitation is that whilst I am very impressed by different portions of the display in certain programs being in different colours, without seeing them in monochrome too I can't definitely say that it is a major leap forward. I do, however, have the definite impression that it may be. How's that for a definitive statement!

In Dave Hunt's bit, elsewhere in this issue, he goes on about selecting a monitor with 0.28mm dot pitch. Well, according to the manual, my monitor has a 0.31mm dot pitch, which doesn't seem much bigger to me. And yet, looking at characters on his screen, and those on mine, mine are much chunkier. We didn't get chance to plug his EGA into my monitor and vice versa, but I suspect that different EGAs must have slightly different character sets, and that Dave's has got narrower ones. Which is better for long term use I obviously don't know, but it looks to me that this is possibly yet another parameter to be taken into account when choosing a system!

With Dave's EGA and monitor he got a demo program. This displays various graphics screens. Running this on my system I noticed that on some of the screens there was a short line of pixels in the wrong colour, and that each time you ran the program it may appear in a slightly different place. This was not the case with Dave's. I reported this too when I went back to Elonex. They put a replacement EGA in my system, but it was still the same. They had another system there with a different EGA, and that didn't show the fault. As I have not yet noticed this problem with any other software on my system, I just asked that they acknowledge that they have seen it, and if I ever have any funnies with anything else I'll be straight back to see them. Meanwhile I have decided to assume that the demo program was written very specifically for that specific EGA, and there's some specific quirk that makes it do a slight funny on my specific type of EGA!

Finally, the display on my monitor is not quite square to the tube. It isn't much out, but when I mentioned it to Elonex they said it would need pulling apart and the magnets, coils, etc adjusting. For the moment I think I'll live with it as it is rather than risk what may happen once fiddling begins!

Summary

For A very reasonably priced system, to a very good specification. Well designed for easy maintenance. "Known" drive manufacturers (except 360K add-on by Copal?). Fairly good looks. Appears to be very "compatible". Good feel (in my opinion) to keyboard. Very fast for the price.

Against Maybe I was unlucky, but apparent Quality Control let down on main board, key-switch/reset switch assembly, missing screw, monitor alignment. Possible potential problem with EGA. Still won't run at 10MHz, 0 wait states for more than a few minutes.

Conclusions

I think you pay your money and take your choice. You will certainly find it difficult to find the same specification for the same money (at the time of writing, anyway). And, of course, if you can live without colour the price is quite a bit less. Remember, however, that my machine did have to go back, although this may well not be typical. If you don't live within easy reach of London this could be a problem, as all sales are direct, and so you can't go to a local dealer.

If you are cost conscious and interested in an AT, I do think that this has to be on your list. And if you're not in a rush, keep reading Scorpio News to see how I get on with it in the future.

The Dave Hunt Pages - PCs & MS-DOS

Introduction

"In the depths of darkest Harrow there seem to be signs of movement. From the dark, damp quagmire, a creature, in perhaps it's most primitive guise stirs, and may be seen to be emerging from the mists which for many months have clouded the clear insight of the vision of that same life form. The creature, if it is such, has its first small glimmerings of intellect, a small vision of that which for such a long period has misted its perception of the trial it set itself, a trial set, it seems, so long ago."

Now if that sounds like a cross between pure David Attenborough (try re-reading it with a David Attenborough tone of voice) and some of the more recent, wordy, science fantasy, then you'd be wrong; it's pure me, only I can't keep writing junk like that for long. Not that I think David Attenborough is junk, I have a high regard for the man, but some of the books I've got from the library recently in the guise of science fiction are so indirect and waffly that I despair of reading more than the first chapter. Either recent science fiction has changed or my tastes have changed; come on Asimov, let's have the fifth part of the Foundation story.

IBM Attack

Anyway, I feel a bit like the first paragraph. About nine months ago my passing acquaintance with IBM type machines burgeoned (note I don't say blossomed) into a full blown, in depth attack. That, viewed in the light of the letters column of the last issue, where many writers confessed to more than passing acquaintance (and in some cases actual ownership) of machines of the IBM ilk; coupled with the fact that I have nothing exciting or even passing clever to say about Z80 based machines at the moment; leads me sit down and write what will no doubt turn out to be a somewhat scathing, but hopefully informative chat about that recent phenomenon, IBMism.

Slowly I've started to see the wood for the trees, the complete and utter ignorance brought on by total lack of information, coupled with the frustration in trying to discover that information, has slowly abated as things have gradually fallen into place. Not everything, whole great areas of ignorance remain, but slowly it's getting itself together. We've been spoiled, you know, from the earliest days of Nascom and a tradition carried on by Gemini, information has been available if you knew where to look, and failing that the manufacturer could point you in the direction of a dealer who would be willing to spend the time to explain, usually without charging. Not so in this mysterious nether world of IBMism. Never have I encountered such a blank wall of unhelpfulness, assumed superiority, misleading and incorrect information; in short know-nothing salesmen.

Twaddle Departments

If you don't believe me, for the almost certainly worst examples and the easiest to find, go to your nearest D*x*ns (preferably a big one) and hang around the computer department for half an hour. Just listen to the twaddle trotted out by your over eager computer salesman. At least D*x*ns has no pretensions, nor is it renowned for the quality of its sales staff. You might think you'd be better off consulting your specialist computer dealer.

These (presumably highly paid) people might know something about business software, and are ever so keen to sell you a system to solve a problem (with what degree of competence I am unable to judge), but when it comes to pure hardware technicalities, simple things (once you know the answers) like what does EGA stand for (Enhanced Graphics Adapter), or what is the line timebase frequency of an EGA card in EGA mode (21.45KHz), then you hit a total blank. They seem totally incapable of comprehending that someone might actually be interested in the machine for the machine's sake; their total lack of knowledge, apart from a few technical sounding buzz phrases is almost unbelievable.

Take, for specific instance, the selection of a suitable colour monitor for my new toy. In every case (I visited five shops) I had to explain to the salesman the difference between the dot pitch of the colour tube (an intrinsic feature of the tube) and the number of pixels (computer generated output) that the monitor was capable of displaying. In every case the assumption had been made that because 'such and such' graphics card was capable of generating 720 pixels per line, then there must be a matching 720 colour dots per line. Not true of course, the more dots the better, in fact a usable minimum should be better than two dot triads (a triad is one blue dot, one green dot and one red dot) per pixel, but tube manufacturers don't measure dot pitch that way anyway. They give the dot pitch as being 'so many per millimetre' or, more usually, 'so many decimal fractions of a millimetre per dot'. The finer the dot pitch, the better; unfortunately, also the more expensive. I finally opted for a 12" tube with a dot pitch of 0.23mm. I think some of the salesmen thought I was slightly dotty myself for being so insistent.

Now it might seem I am maligning some very competent and knowledgeable computer salesmen. Not a bit of it, it just seems to me that computer salesmen are on the whole useless. Those few (I haven't found one yet) who are knowledgeable and competent will know who they are (and likely share my low regard for their compatriots), my advice to them is get out now, you're probably more worthwhile somewhere else.

So having dismissed salesmen as a source of information, where to next. Well a number of friends have recently defected towards IBMism, usually along the clone route, but in the main they are as much in the dark as I am. A couple of them have had the time to tinker and therefore have sussed certain bits of interest to them, but for an overall knowledge, I haven't found anyone yet. (If I did find the 'Fount of all Wisdom', I doubt that I would be able to afford the considerable amount of time required to absorb the information, and for that matter, I doubt that the 'Fount of all Wisdom' would have the time for li'll ole me.)

Books and Colleges

That leaves the technical colleges and books. Well, I do the night school teacher bit at Paddington college (not a computer subject) and I know they don't do short courses (or long ones either) on 'The Innards of MS-DOS', '8086 (et al) Assembler Programming' and 'IBM type Architecture', which is what I need. The best they aspire to is to keep asking me if I'd like to take an introductory course on Z80 assembler, and I keep saying no! Slough Tech did send me a glossy leaflet on just those sorts of subjects, but the very expensive quality of the leaflet made me very cautious about the price before I enquired, and I found my caution justified. The courses were obviously aimed at business and were priced commensurately, quite up to the price of some of the private rip-off classes I see advertised in the computer mags. Anyway, colleges are out, they either don't do the courses, or if they do, the charge is OTT, at least on a private interest basis.

Books, well books abound, ranging from good to Mickey Mouse. The choice is almost too much. These sort of technical books are expensive, but not as much as the college courses. They may be hard to get, except by mail order, or to those within easy reach of the Modern Book Shop in Praed Street, Paddington, but at least they are obtainable. I have acquired two or three which suffice for the software side, and I have the IBM AT Technical Manual coming which might go some way to answering some of the more obscure hardware permutation problems I've hit.

Of the books, "Programming the 8086/8088" assembler manual by Sybex suits for the internal structure of the 8086 family of processors with a full breakdown of what makes it tick, the instruction set and examples of little programs. The book covers the 8088 and 8086, being very similar processors, it doesn't cover the cleverer and newer processors, the 80186, 80286 and 80386, but as the sort of things I'm called upon to write have to cover the whole gamut of IBM PC's, XT's, AT's and all the various breeds of clone, I have to write things at the lowest level, so 8086 assembler suffices. For those who are not aware,

the newer processors all contain the 8086 instruction set as a sub-set of their own instructions, and the new instructions are primarily (but not solely) concerned with extended (over 1 Mbyte) addressing modes. The book falls down on the subject of the segment registers and how these are programmed (I still occasionally have trouble with these as the numbers which magically turn up in the registers tend to have no relation to numbers I told them, or expected them to contain). I've mostly got this sorted out with the aid of the examples in the Microsoft MASM (Assembler) Manuals and the Norton book, but they still catch me by surprise every now and then. When I've got this aspect fully sorted, then maybe I'll revamp the 'Dodo's Guide to Z80 Assembler', you know, the thing I wrote all those years back, turn it into the 'Dodo's Guide to 8086 Assembler', and see if this mag. will republish it.

IBM Compatibility Problems

The problem with IBM compatibility all stems from the fact that IBM from the outset allowed direct memory addressing to the screen RAM and the use of direct BIOS calls. This, of course, completely blows the idea that all I/O should be done through the DOS and therefore be machine independent, as it firmly fixes the hardware to a certain form. I'm quite sure this was not Microsoft's intention, and seems a step backwards.

Many of us are aware on the incompatibilities which come about by trying to transport CP/M software from machine Brand X, where the software rules have been broken and direct screen and BIOS addressing has been allowed, to machine Brand Y, where the I/O and BIOS are at different addresses. One of the sneakier tricks of a certain piece of early software, which was written for the IBM alone, was to make it look at the beginning of the BIOS, where the message 'IBM CORP. (C) 1981' is held. If the software didn't find the message, it wouldn't go. One entrepreneurial type soon got round that one in his clone by including the message 'NOT IBM CORP. (C) 1981' in his BIOS starting four bytes to the left. After that, attempts to make software machine exclusive seem to have ceased. The new Microsoft operating system, OS2 has the same publicly declared intention that all I/O will take place through the DOS itself, and Microsoft say that they have made it next to impossible to use direct addressing effectively. We shall see; and also wait to see how long it takes Big Blue to get round that one.

Back to the Books

So given that the DOS, the BIOS and the hardware are all much more closely woven together than in a CP/M machine, what are the most useful references? MS-DOS, the BIOS and hardware architecture related software points are very adequately covered by Peter Norton in his book "The Programmer's Guide to the IBM PC". This book unashamedly covers IBM machines only, but points out that most of what is said applies to a good clone. The book covers all versions of DOS from version 2.0 (with passing reference to DOS 1) to version 3.2, the differences, the enhancements and the bugs. IBM appear to have made fewer BIOS's than Microsoft have made versions of MS-DOS, and the book covers the different versions adequately. It also has thorough extensions related to the greater facilities available within the XT and AT BIOS's. It also notes the differences and implications of different hardware / DOS / BIOS permutations, although these can get quite hairy. It can not and does not cover all possible permutations, but suggests a 'try it and see' approach to some of the weirder perms. The information on the BIOS and DOS is adequate to allow you to spot 'compatibility' problems without resorting to the source listings in the IBM Technical Manuals.

Another book, the title of which I forget, as I've loaned my copy to someone - something like 'Advanced MS-DOS' by the Microsoft Press, covers the inner naughties of the various versions of DOS, including lumps of source code and bug lists (very useful). This book is confined to MS-DOS and as such is not IBM dependent.

Compiled This and That

One thing is apparent, and seems to owe something to the growing 'maturity' (if that's the word) of the software writers for PC type machines, or, perhaps it's main frame practices creeping in. That is, as little as possible is written at assembler level. Most applications encountered seem to be compiled this, or compiled that, or they don't say they are compiled, but the sheer size of the object code says they must be. Now, I've got nothing against compilers, in fact most of what I've been doing over the last few months has been compiled, so I don't knock it. But when I look at compiled object, its size says that it can't possibly be as efficient as good old hand written assembler. In fact of course, this could not be, but it depends on where you place your benchmarks as to what determines efficiency. Taking hand written assembler as a base line, one could argue that a compiler is several hundred percent more efficient when compared with the time taken to write an equivalent program in assembler. Or it could be argued that compiled code is only some tens of percent efficient when it comes to the size of the programs created, the same applies to operating speed. Now I don't intend to get into any arguments about benchmarks, I'll leave that to Paddy Coker; but it seems to me that comparing the 'goodness' of anything must take into account all these things (and probably a few more parameters I don't see right now) and not just confine itself to the operating speed of a program! Just how and where you adjust the trade-off, I don't know.

Now there are compilers for this language, and that language and each has its advantages and disadvantages. Of the compilers around, I favour those languages which have matching interpreters; as an interpreted program is usually much easier to edit than one you have to recompile every time you have tweaked it. An interpreter is invaluable in the early stages of learning a language on your own, in getting round the inevitable syntax errors caused by such things as leaving off semicolons on the ends of lines in Pascal and C programs, very frustrating as the compiler stops every time it hits an error. Of the languages available, I use a very few, being a very pedestrian type who hates having to learn anything new; you can certainly list them on the fingers of one hand, BASIC - fairly proficient, C - pretty hopeless, DBASE - very good (although I say so myself, I get lots of practice), and PASCAL - forget it. I've recently heard of a C interpreter, so I'm keen to get my hands on that and brush up my C a bit. But in the main, armed with the two languages I feel competent with and a bit of assembler to 'paper over the cracks' caused by the deficiencies of the compilers, I feel ready to tackle most things.

Software Packages

As to applications/utility software I use, I have found that SIDEKICK is perhaps the most invaluable. The major features are that it is memory resident, so it's always available (and always knows exactly what you were doing last) and consists most usefully of a calculator in decimal, binary and Hex, a calendar with appointments dairy, and a mini word processor, well a context editor really. The context editor endears itself particularly as it is very similar in the way it works to a convenient cross between GEMPEN and WORDSTAR. The text is RAM resident (so it won't handle large files) whilst most of the control codes are WORDSTAR compatible. There are a lot of other frills as well, including a phone directory and dialler, but I don't use them. I've tried several different versions of SIDEKICK-like things, but still I always come back to SIDEKICK.

Memory Resident Quirks

Mind you, memory resident software has to be treated with a certain caution. Strange things happen. I'm told, not actually seen myself, that SIDEKICK hooks itself into the keyboard, looking for a certain keyboard key press combination to see if it's required. To that end, it redirects the keyboard interrupt vector to itself for examination prior to sending on the key press to where-ever it was going in the first place. Ok, now that's logical, but SIDEKICK could fall over if another program redirected the keyboard interrupt vector to ITSELF. SIDEKICK is sneaky, to ensure this doesn't happen, SIDEKICK has also hooked itself into the 'clock tick' interrupt, so that SIDEKICK gets woken up about 18 times a second. On waking up, SIDEKICK looks at

the keyboard interrupt vector to see if something has nicked it, if the application has changed it, SIDEKICK changes it back. Result, the program which 'took' the keyboard has now lost it, so SIDEKICK works, but the application doesn't. Moral, if you've acquired a piece of 'foreign' or 'liberated' software which doesn't seem to work, and SIDEKICK is resident, don't give up on it until you've turned SIDEKICK off and tried again.

A worse situation than that is where two or more memory resident programs are asked to co-exist together, perhaps SIDEKICK and something else, say FRED. Can you imagine the fight if they are both polling for keyboard interrupts. Problems of this sort can often be solved by the order in which the programs are loaded. For instance, we already know that SIDEKICK picks up the original keyboard interrupt vector and replaces it with its own, having processed the keyboard interrupt it passes it on to the original vector. So if we loaded SIDEKICK first and then FRED, SIDEKICK would know that the keyboard vector had been changed by FRED and so change it back. However, if FRED were loaded first, it would change the vector to suit itself, then SIDEKICK comes along second, sees the new vector to FRED, picks that up as the output direction vector and replaces it with its own. All should work Ok, as SIDEKICK will see any keypress first, and if it's not for itself, it will then pass it on to FRED. Of course, all that presupposes that FRED doesn't indulge in the same tricks as SIDEKICK in the first place.

Mystery Time

Here's one for the people who like mysteries. The utility called MODE, supplied with DOS, is used amongst other things to set the speed of, and to direct output to, the serial I/O. When executed for the first time, it leaves part of itself memory resident. Now the general way memory resident programs are handled are that they are loaded into low memory and a pointer is set to the top of the program pointing to the next 'paragraph' of memory usable as transient program area. Not that it concerns us here, but all programs for 8086 et al are relocatable in 'paragraphs' (16 byte boundaries). Suffice to say, DOS knows where to starting loading the next program. All this takes place with MODE. One of my proprietary programs makes use of this relocatable bit, as there are times when I want to change the UART parameters from inside the program, and it was convenient to 'RUN' MODE from inside the program rather than hack some assembler together to do the job properly. It just so happened that there's oodles of room available on top of the program to accommodate MODE, so no problem - or at least I thought there wasn't!

After a while, it was found that the applications program ran out of memory if someone indulged in a lot of UART parameter swapping. The answer was obvious, every time MODE was run, it didn't recognize that it had been run before, and so left the resident part behind, reset the load paragraph pointer upwards, and so slowly filled the memory until the application hiccupped and copped out. Ok? But how come MODE knew if it had already been run from DOS and didn't leave the resident part behind then? Well I didn't twig that quite so soon, as I didn't initialize the serial I/O from the AUTOEXEC.BAT, but I did fire up SIDEKICK in the AUTOEXEC. I checked with a couple of other customers to see if this problem had occurred, it hadn't, the only apparent difference was that they initialized the serial I/O in the AUTOEXEC and weren't using SIDEKICK. Prewarned that SIDEKICK did naughty things, I turned it off, but it no difference, the memory still filled up. So it looked as if SIDEKICK could be eliminated.

Then I tried executing MODE before executing the program, and the problem went away. In other words, I guess (but I'm not sure) this is what is happening. By executing MODE first, the resident part is left in the rock bottom lowest part of RAM. The next time MODE is executed, it checks to see if it already exists, but it only checks at the lowest address in RAM. If it finds itself, it doesn't bother to leave the resident part behind, if it's not found, then the resident part gets left. So now MODE gets executed in the AUTOEXEC, followed by SIDEKICK - end of problem - only I'm not entirely happy about my explanation - they don't say anything about that in the manual, it seems right, but does anyone know for sure?

More Useful Programs

A course I use a number of other utilities and programs, some of which I find highly recommendable.

One I find invaluable is XTREE version 2.0 by Executive Systems. This is a sort of super SWEEP, very useful for burrowing about in the subdirectories on my winnie and generally shovelling stuff from one place to another. It has all the usual facilities including the ability to execute and run programs from within itself, and to 'view' text files, whilst removing all the funny control flags from most of the more usual test processors. About the only thing I don't like about it is that it uses the f1 key to exit. In my experience, the f1 key is used by most programs for 'HELP'.

Another, a public domain one this time, is the file archiving suite by Phil Katz, PKARC and PKXARC. It has all the advantages of LU and NULU with none of the disadvantages. A major feature is that it 'squeezes' the files as it goes. As it analyses the files for squashing, it automatically makes a choice of one of four different 'squeeze' algorithms to achieve the greatest packing density. Having re-squeezed a number of CP/M text files which were previously squashed with NSWEEP2 and then archived under NULU I found that PKARC made very significant savings indeed. The only problem is that it is command line driven and to that extent, awkward and inflexible.

Directory programs are ten a penny. Why Microsoft didn't make the DIR command do an alphabetical sort I don't know, except perhaps to keep all those people who delight in writing directory programs happy. The one I use is ZSPD by an anonymous author. It does what I want including the ability to look at the .ARC files produced by the archiver to see what's inside. I made use of this program to do the hard bits in my rather cobbled together 'DAVE'S MESSY, SLOW, OVER LARGE CATALOGUE PROGRAM' which I wrote over last Christmas to relieve boredom and an overdose of mother-in-law.

The DRH Catalog Program

On the subject of my cataloguer, I wrote it because I couldn't find anything half decent knocking around in the public domain (I'm not saying there isn't one, simply that I haven't found one). I based the idea on a cross between the best parts of the original Ward Christensen CAT program and the later FATCAT. The original CAT idea is a little too simplistic for use with a machine which can use subdirectories and archives, whilst I found FATCAT so difficult to use (to mention nothing of its size) that you might as well not bother. So my cataloguer is simple to use, whilst doing everything I want it to do. As to its size - that's a different matter. I recognized that cataloguing is a database management problem, so I wrote in compiled dBASE. There's only one problem with the Clipper compiler I use, and that is it generates 120K of code just to say:

```
@ 10,5 SAY "Hello"
```

If you tell the compiler to generate native code (which goes faster), it's even bigger! So the cataloguer stands at about 150K. That's uncomfortably big for a disk based machine unless you keep the cataloguer and its database on a separate disk, but is comfortably lost in amongst the other 2.5 Mbyte of system utilities I've got knocking about on my winnie. The main thing is that it does what I want. It's still got couple of bugs that need fixing when I get round to it, but it goes.

A nice thing about fully compiling dBASE is that once you've paid for the compiler (which I have, fully legit - honest), there's no 'kick-back' to Nantucket, who wrote it. Far better than Ashton-Tate's pseudo compiler which is not only incredibly slow, but they'll charge you £65.00 for every copy of dBRUN which you would need if you sold the finished product.

There is some talk of Scorpio Systems going into the public domain program supply market, if they do, then perhaps they can have my CAT program, and its companion ADDRESS program to see if anyone wants them. If you don't fancy sending loot in the direction of Scorpio, then you can pick up those two, and the other public domain programs I've mentioned on the CBBSLW bulletin board for the cost of a (long) phone call.

Adding 720K drives to an Amstrad

One last (long) thought, passed on to me by Brian Hayward of Colchester, for those who own Amstrad 1512's with one drive and haven't forked out the necessary for the Technical Manual. Statement: If you've owned a Gemini or Nascom with 800K disks, bet you're pretty peeved to discover that the maximum disk capacity of the the Amstrad (and most other clones) is only 360K per!! Now Brian has both; that is, a one drive Amstrad and the Technical Manual; and he's found a new command for the CONFIG.SYS file (it actually ends up in the 'environment string space') and it goes like this:

```
DRIVPARM=/D:dd[/F:ff/T:tt/S:ss/N/C/H:hh]

     where /D:dd is the drive number (0-255)

     and optionally:
          /T:tt is the number of tracks/side (1-999)
          /S:ss is the number of sectors/track (1-99)
          /H:hh is the maximum head number (1-99)
          /C indicates changeline (door lock) support required
          /N indicates non-removable block device
          /F:ff indicates the format factor, where:
               0 = 5.25"
               1 = 5.25", 1.2 Mbyte
               2 = 720K
               3 = 8" Single density
               4 = 8" Double density
               5 = Hard disk
               6 = Tape drive
               7 = Other
```

Zap an 80 track drive in the spare hole and, set DRIVPARM correctly, and lo, you have 720K on that drive. Nice, isn't it.

How do you get 720K per disk on an AT?

Well I run DOS 3.2, so I tried the above on both my AT clone and one of the real genuine IBM AT's at work with peculiar results! Firstly an AT is fitted with a double sided 80 track dual speed drive, giving 1.2 Mbyte on that drive, made up of 15 sectors of 512 bytes/track. It can also double step the drive and forget about the double speed bit and give the 'IBM standard' 360K, double sided 40 tracks, 9 sectors of 512 bytes/track. The snag is that at 1.2 Mbyte, the disks are expensive, whilst 360K per disk is hardly enough to be usable. Wouldn't be nice to keep drive A as a 1.2 Mbyte drive, whilst at the same time using the drive as 720K logical drive at a different logical address, mainly because I've got lots of ordinary disks and very few high density disks. My clone is also fitted with a 40 track Teac as drive B.

So first I set the drive parameters for drive A, 80 track 9 sectors/track. I used the Amstrad FORMAT.EXE, because a 'file compare' showed it was different to my original format program, and as the DRIVPARM ends up in the 'environment string space' I figured that the program might actually look there to see what drives it had got. The darned thing paid no notice, it still formatted the drive as 80 track, 15 sectors/track, that is 1.2 Mbyte. So then I set drive parameters to drive B, the Teac 40 track, and lo and behold, the dear old head moved 80 steps (Ok, so as it was a 40 track drive, the last 40 steps were knocking against the end stop), but the disk reported 720K when it had finished. Still it proved that removing the 40 track drive and replacing it with an 80 track would work, but that defeated my aim.

So next I thought I'd try using the ASSIGN command from DOS, guessing that the BIOS always picks up the drive parameters from drive A from the ROM BIOS and forces it to do the double speed bit if the drive is 80 track. I set up the parameters for drive E and then assigned drive A to drive E. I got 80 steps alright, but still 15 sectors/track. Disgusting!!

Then I took the 40 track drive out and shoved one of the 80 track Teac drives out of my Gemini in the hole, all should work now - or so I thought. I set up the parameters to set drive B to 80 track and told it go. Can you imagine my amazement when the 80 track drive proceeded to double step, do 40 tracks, spend the next 40 steps banging away at the end stop and then have the nerve to tell me the disk had 720K.

At that point I assume I had got the select links on the Teac mixed up, but for the life of me I can't see what I had got wrong as it still worked as an 80 track drive when it found its way back into the Gemini. Seeing it was about 3.30 in the morning I left it, and haven't come back to it yet. But then again, someone out there has probably achieved my aim of using the 1.2 Mbyte drive on an AT as a different logical 80 track 720K drive, sitting there assuming that everyone knows the trick, so perhaps if they would like to drop me a line .....

Next Time

Well I guess that about wraps it up for this time round. Next time I'll chat about the fairly painless conversion from dBASE II to dBASE III Plus, some of the not so nice surprises it has in store, and how to go about compiling it using Clipper, Quicksilver or Foxbase, none of which is entirely compatible with each other.

Private Adverts

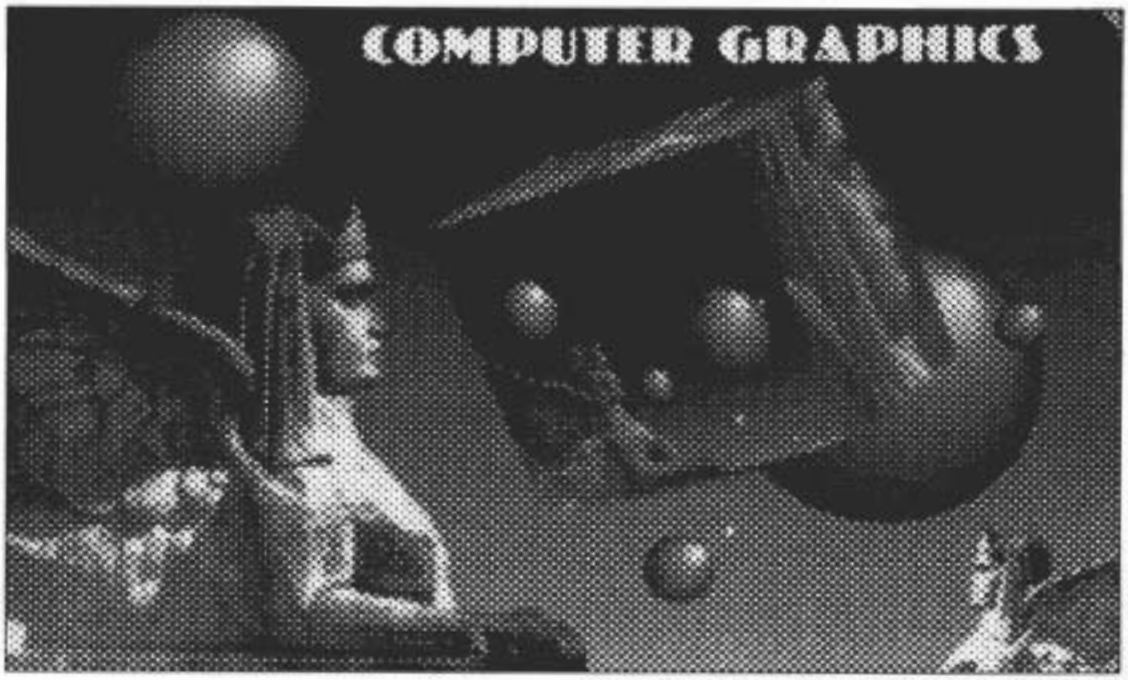
Nascom 3 system for sale. Full working order with manuals. Complete with 48K RAM, input board with counter timer, floppy disk controller, dual disk drive and case, Kaga 12" monitor. Also fetal monitor system interface! Offers for whole or parts. Phone Andrew Wheble on Reading (0734) 663349.

HALF PRICE! GM833 512K RAM-DISK board, £197.50. Being sold on behalf of emigrated owner, therefore can not accept "offers"! Also GM816 PIO/CTC/RTC board, £90. Phone 0296 624868.

HELP! Has anyone got a working version of a patch for WordStar and the Smith-Corona TP1 daisy-wheel printer? Having struggled without success for several days, I'd be grateful if anyone can help. Dr P.D.Coker, 23 Darwin Close, Orpington, Kent BR6 7EP (0689 58510)

Gemini Galaxy 3 system for sale. Grey case, low profile keyboard. 5MB Winchester, 800k floppy. Complete with GM816 PIO/CTC/RTC board and GM818 serial daughter board. All maunuals, etc. £1300 + VAT. Contact Finlay Microfilm, 18 Woodside Road, Amersham, Bucks., HP6 6PA. Amersham (02403) 22126.

For sale: 130+ x 8" scarcely used SSSD disks, good condition. £4.75 for 10 inc. postage. Dr P. Coker, 23 Darwin Close, Orpington, Kent BR6 7EP. (0689 58510).



ARCTIC COMPUTER GRAPHICS

System Sales
Full Bureau Services

PCR	ARCTIC	Slides
Laser Prints		Video
Pen Plots		Colour Prints
2D & 3D		Frame Grabs

Arctic Computers Ltd.

6 Church Street, Wetherby, West Yorkshire, LS22 4LP

Telephone (0937) 61644

XT-Compatible 20 Megabytes 4.77/8 MHz

£795

- The PC88 Turbo
- Legal BIOS
- NEC V-20 Super Processor
- Switchable 4.77/8 MHz Clock
- 640K RAM
- 20MB Half Height Hard Disk
- 360K Half Height Floppy
- Parallel Printer + RS232 Serial ports
- Clock/Calendar with Battery Back-up.
- Monographic/Printer card
- 130W Power Supply
- Professional Keyboard
- "Hi-Res 12"" Green Monitor
- MS-DOS 3.2
- Full Set of Manuals
- 8 Expansion Slots
- 12 Months Warranty

AT-Compatible 20 Megabytes 6/8/10 MHz

0 or 1 Wait States

£1295

- The PC-286 Turbo
- Legal Bios
- Switchable 6/8/10 MHz
- 640K RAM
- 20 MB Half Height Hard Disk
- 1.2 MB Half Height Floppy
- Parallel Printer + Two RS232 Serial Ports
- Clock/Calendar with Battery Back-up
- Monographic/Printer card
- 200W Power Supply
- Professional Keyboard
- "Hi Res 14"" Green Monitor
- MS-DOS 3.2
- Full Set of Manuals
- 8 Expansion Slots
- 12 Months Warranty

OPTIONS

- 30 MB Hard Disk Drive Upgrade £265
- 40MB Hard Disk Drive Upgrade £365
- 14" Colour Monitor Upgrade £175
- 14" Hi Res Colour Monitor and EGA Card Upgrade £395
- EGA Card £165

'ON SITE MAINTENANCE AVAILABLE'

ELONEX (UK) Ltd.

Please feel free to visit our showroom for demonstration

RAYS HOUSE, NORTH CIRCULAR ROAD, STONEBRIDGE PARK, LONDON NW10 7XB TELEPHONE: 01-965 3225

The Best! from HiSoft

Devpac80 Ver. 2

Devpac has always been *the* popular Z80 development system used by Software Houses all over the world since we first introduced it for CP/M over 6 years ago. Many regard it as a standard to work from. Now we have extended the standard to produce the most powerful development package yet and still at the same price as before!

- ✓ fully-integrated screen editor, remembers your assembly errors and takes you straight there for correction.
- ✓ standard Zilog macros with strings, nested conditionals & text include, full expression handling with absolute and relative values.
- ✓ produces standard .REL files or .COM files. .REL files allow you to link the output with the output of most other languages e.g. CBASIC™, ProPascal™, ProFORTRAN™, all MicroSoft languages etc.
- ✓ assembler produces standard .SYM symbol table files for use with the symbolic debugger or with debuggers from MicroSoft and Digital Research.
- ✓ the debugger is symbolic i.e. you can see the labels you used when assembling in the code that you are debugging.
- ✓ debugger uses full conditional breakpoints and watchpoints so that you can even profile your program to highlight slow areas.

(Version 2 of the debugger requires 80 column display).

All this power with an extensive, quality ring-bound User's Manual for the incredible price of

£39.95

Existing Devpac80 owners can upgrade for only £15, just send back your old disc and manual pages.

HiSoft Pascal80

Pascal80 is our fast, standard compiler for CP/M Z80 systems. It provides an almost complete implementation of Standard Pascal as defined by Wirth. We have been careful to adhere to the definition while providing extra facilities to exploit your computer. The compiler is written in Z80 and produces Z80 object code directly to a COM file - no P-codes or linkage.

- ✓ fully-integrated screen editor, with menu system for easy compilation and program testing.
- ✓ Fast one-pass compiler producing compact, optimised and very fast code.
- ✓ Version 2 has variant RECORDS and FILES of any type, register variables, and upper or lower case reserved words.
- ✓ Compiled programs can CHAIN other programs with common variables.
- ✓ Low-level functions for BDOS and m/c calls, PEEK & POKE, INLINE and ADDR, INP and OUT.

Pascal80 comes complete with an extensive ring-bound User Manual and sample programs for only

£39.95

FTL Modula-2

Great value-for-money, this up-to-date and full implementation of the successor to Pascal is easy-to-use with an integrated editor, assembler and linker together with the Modula source code for most of the predefined modules.

Versions available for both CP/M (48K TPA) and for MSDOS to give upward compatibility.

The full source of the split-screen, macro editor is available separately at £39.95 or for £35 when purchased with the compiler. Go for Modula-2 today!

£54.95

HiSoft C / Aztec C

C is now a very popular and important language. We offer two entry points, one for beginners or casual users and one for the experienced C programmer. Both can be supplied on request with our extensive GSX library for graphic programming on CP/M Plus.

HiSoft C £39.95

- ✓ fully-integrated editor, reports your error and takes you to the rogue line, ready for correction.
- ✓ super-fast and compact code.
- ✓ Kernighan/Ritchie without floating-point or linkage.
- ✓ **Special Offer!** Supplied with tutorial book **C at a Glance** ***free! Offer ends July 1st 1987, so hurry!***

Aztec C' Prime £79.95

- ✓ industry-standard compiler with upgrade path to developer and commercial versions.
- ✓ assembler, linker and object modules all supplied, produces efficient, fast code.
- ✓ full Kernighan/Ritchie with floating-point.
- ✓ extensive manual with tutorial.

Nevada FORTRAN

The language for science and engineering. Nevada FORTRAN compares favourably to mini- and mainframe FORTRAN compilers. Based upon the ANSI-66 standard it has many 'real-life' extensions (IF..THEN..ELSE, COPY (include), CHAINing with COMMON, TRACE debugging etc.) making it perfect for the beginner wanting to learn the language or for the experienced programmer who wants to test out routines at home or develop a program quickly away from the VAX queue.

Dynamic module loading lets you load modules in seconds. No linking is necessary - just compile and run. Very small object code modules conserve valuable disc space, and there is a run-time package - not an interpreter - for additional space savings. Also included is an assembler and 8080 assembler mnemonics may be included in your FORTRAN program.

Nevada FORTRAN comes complete with HiSoft's full screen editor, ED80, examples programs and a 200 page manual for only

£39.95

Please rush me the following HiSoft Software for my Gemini/Nascom:

Product **price**

Disc format **Day-time phone no.**

Please charge my Access/Visa card:

Card No. Expiry Date

I enclose a Cheque / Postal Order to the value of £

Name:

Address: Sca/6/87

Post Code: **Signature:**

HiSoft The Old School, Greenfield, Bedford MK45 5DE ☎ (0525) 718181