October-December 1987 Volume 1. Issue 4

Scorpio News

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

NEWBURN

NEWBURN ELECTRONICS, 58 MANSE ROAD, BALLYCARRY, Co. ANTRIM
BT38 9LF Tel. 09603-78330

NE889 512K STATIC RAM-DISK BOARD

- The Newburn NE889 battery-backed static RAM-disk board is an 80-BUS compatible board for use as a high-speed virtual disk ("M" drive") in all 80-BUS systems. The board is compatible with all versions of CP/M that support one or more "M" drives via I/O ports.
- Being a battery-backed board, data is not lost on power-down. On repowering the system, data is immediately available.
- The board is supplied with 512K of battery-backed static RAM. Versions with 2MBytes CMOS RAM, or for EPROM are also available. Memory sockets are 32 pin, allowing an eventual upgrade to 16 MBytes per board.
- The board can be write-protected by both an on-board dil-switch, or by grounding an edge pin (for keyswitches etc.).
- Together with a modified CP/M, this board can be switched to 4 separate I/O ports giving four "M" drives on the same system (Up to 32MBytes).
- The operating system can be optionally cold-booted from this board, producing a fast, rugged, disk-less CP/M system. A boot EPROM is available for all 80-BUS CPU boards.
- This board provides an ideal environment for fast, ruggedised control systems, where program chaining, overlays etc. are fast enough for real-time control.

Price £395 + VAT.

GM890 Z280 PROCESSOR BOARD

- Turbo charge your Z80 system. Uses the 10MHz, Z280 processor to speed up all existing Z80 software by a factor of between 2 and 12 times.
- The board contains 256K RAM, (up to 4MBytes), four (32 pin) EPROM/CMOS sockets (up to 4MBytes), real-time clock, Gemini compatible PIO and UART and battery backup.
- This board is 100% compatible with existing Z80 software and is a direct plug-in replacement for an GM811/GM813 CPU board in any 80-BUS systems. A preferred CPU board for any new designs.

- For advanced operating systems and user-written code, read on....
  Separate system/user modes. Separate memory for program, data, operating system. Interrupt routines and traps. 7 interrupt levels. 8 vectored traps including privileged instruction, invalid address, write protect violations, stack o/flow, div. by zero, halt in user mode, etc. MMU for any 4K blocks within 16 Mbytes. 256 byte fast cache or memory. 64K (paged) I/O ports, additional addressing modes. Three 16-Bit counter timers. Additional full duplex UART. Multi-CPU handshakes. Four DMA channels. Auto 256 Byte bootstrap via UART. Dual stack pointers. Single-step operation. Multiply & divide instructions. System call instruction with 16 MByte displacement. 64K displacements on IX, IY, SP & PC relative instructions. Auto I/O and address translation.

Price £450 + VAT. Limited Period Only... £410 + VAT.

NEWBURN

October - December 1987 SCORPIO NEWS Volume 1. Issue 4.

Contents

Editorial

Nascom - 10 Years On

Thank you to everyone who wrote in about the proposed *Nascom - 10 Years On* event. There was an avalanche of mail on the topic, and you have given us a lot to consider. Whatever we do I am sure that it will be an event to remember. In due course we will be writing directly to all of you who have expressed an interest, saying exactly what the arrangements are going to be. If you haven't yet been in touch with us, it still isn't too late.

To re-cap, this November will be the 10th anniversary of the launch of Nascom 1, the machine that started (well, almost) the 80-BUS that we all know and love today. We are going to arrange a "happening" and have asked readers to write in and let us know what they would like the event to be (see last issue's *Editorial*). We are then going to take an "average" of the suggestions. So, if you too are interested, then please let us know soon (in writing).

What's New

Every so often someone will say to me - "It's a pity that 80-BUS is dying out - I think it's great. Why are there no new developments taking place?" Well, maybe Gemini's laid-back policy to publicity (i.e. they really don't tend to do very much) is the reason, but there has never been a let-up in 80-BUS developments, and I think that this issue of *Scorpio News* shows that there is still much going on.

In the following pages you will find details of THREE new 80-BUS CPU boards. As I write, none of the boards are actually here yet (what's new?), but they all exist in pre-production form and so should be with us RSN (Real Soon Now). I was going to offer some advice on how to choose between the 64180 and Z280 based boards, but I haven't really had chance to way up the pros-and-cons myself yet. Please do read my notes on page 53.

You will also find a brief technical specification of Gemini's 80286 CPU/RAM board. Now do you see another reason why we thought that it was becoming appropriate to include some coverage on MS-DOS in *Scorpio News*? And there may be yet another purpose soon!

Dealer Profile - Kenilworth Computers

Kenilworth Computers Limited started life in the computer retailing market as Business & Leisure Microcomputers in October 1979.

David Searle served his time as an apprentice with The Rover Company Limited, as a Mechanical Engineer, and became involved in computing by mistake (don't we all!) about nineteen years ago. He was involved in Project Management at Rover and required a network analysis program. In the absence of anything suitable he had to design and install his own, which proved highly successful.

After the Rover Co. became British Leyland (or was it Rover Triumph? - the name changes came thick and fast) the whole structure of management altered and he found that he was being paid more and more for doing less and less which is fine for a time, but boredom and frustration rapidly take their toll.

David took voluntary redundancy and invested that and the proceeds from the sale of his house in a motel and pub in the middle of Wales backing onto the River Wye. Idyllic setting, but after you have installed all mod. cons. for the guests and perfected the art of keeping real ale, the old grey cells are apt to wither or even worse drown in alcohol!

So to stimulate the grey matter he purchased a Nascom 1 kit and was excited when he discovered that this primitive looking microcomputer had as much processor power as the IBM machine that started his computing career. It must have been the first pub in Wales, or possibly anywhere in the U.K., to have a computer board and keyboard, unboxed, sitting in the back bar with games running from tape to entertain the locals. The locals loved it, and not too much beer got spilt over it - and in any case the Nascom did not seem to mind!

However, after three and a half years of slave labour in the business, his wife Sue decided she had had enough. That, combined with the possibilities for starting a computing business, based on Nascom 1 kits, and the need, for family reasons, to return to his home town of Kenilworth, lead to the setting up of Business and Leisure Microcomputers' shop on the main street in Kenilworth in October 1979.

What exciting days those were, with Nascom 1 then 2 kits being purchased in hundreds for a myriad of different applications by one man bands or the giants like GEC, British Telecom, Plessey, etc.. Then came the avalanche of cheap home computers, starting with Sinclair, and B & L Micros found themselves being used as an information bureau, with customers gleaning all the knowledge needed to make their choice of home computer to purchase from one of the multiple giants whose prices were ridiculous.

By early 1982 David was intrigued with the possibility of producing a portable computer, a full featured machine in a briefcase (some people had mischievously suggested a suitcase was more appropriate!) and a prototype was constructed. It was on display at Compec '82, the same year that the prototype Osborne Portable was shown.

Shortly after this, David was invited down to London to meet Adam Osborne with a view to becoming a dealer for his portable. For various reasons, mostly to do with non-standard screen and inadequate disk size, he decided not to take on a dealership, but to pursue more vigorously his own design of portable, which by this time had changed to the present upright shape. Sadly, although the project came to fruition, it was so bugged with manufacturing quality problems of the case, which delayed its introduction, that it missed its slot in the market. Nevertheless a goodly quantity of Kenilworth Portables have been sold.

As B & L Micros gradually turned their back on the home and hobbyist end of the market and started to concentrate more and more on specialist and educational applications, it seemed that office accommodation would suit the style of business much better than a shop front. Also they needed to drop the Leisure from the B & L title, therefore what better than to absorb B & L into the newly formed Kenilworth Computers Ltd., set up to produce the portable.

During the past eight years the business has taken on various dealerships including Commodore, Comart, Sinclair, Dragon, but has been consistently faithful to Nascom and Gemini 80-BUS systems.

In fact, serious thought was given at the time of Nascom's initial demise as to the viability of purchasing the business. But when Lucas stepped in KCL felt that Lucas had considerably more

financial muscle, which would be advantageous for developing Nascom's potential. How disappointing the outcome proved to be! It has however ended up by providing KCL with hundreds of poorly Nascoms to repair over the years.

Other than their industrial customers, Kenilworth Computers Limited's business has ranged from standard word processing systems, spread sheet templates designed to the customer's specific requirements, a point of sale program, a borehole analysis program to detect land movement due to settling, and a label overprinting package. In fact any area where specialised software is required.

As has happened with many dealers, survival has lead to the need to specialise more and more. Kenilworth Computers' strength is in solving industrial application problems, be they hardware or software, or preferably both. Of course, the diversity and flexibility of the 80-BUS boards have enabled them to offer cost effective solutions to their customers, and are the mainstay of the hardware side of the business.

Gemini GM880 Hitachi 64180 CPU/RAM Board

Gemini have announced the imminent availability of a new 80-BUS CPU board, the GM880. The following description has been extracted from pre-release information sent by them to their dealers.

PRODUCT OVERVIEW

Gemini are currently testing a prototype of a new MultiBoard CPU which uses the Hitachi 64180. This board has been designed to replace the GM813, but please note that Gemini have NO intention of discontinuing the existing product.

The Hitachi 64180 (Zilog call it the Z180) is fully compatible with the Z-80. All object codes are also fully compatible unlike the Z280 which despite advertising "hype" is only "almost" compatible.

The chip itself has the following specification:

- Operating frequency: 4, 6, 8, 10 and 12MHz versions
- MMU supports 512k bytes of memory
- 2 channel DMA for fast memory-memory, memory-I/O, etc.
- 2 channel full duplex asynchronous serial communication (ASCI)
- Clocked serial I/O port with high speed interface (200k bits/sec.)
- 2 channel 16-bit programmable re-load timer

HARDWARE SPECIFICATION

This design will interface to the MultiBoard range of products in the same manner as the GM813. The basic specification of the board includes the following:

- A 64180 running at 10 MHz with no wait states (12MHz option)
- 1 MByte of memory
- A Z-80 4 MHz PIO
- Support of the GM833 512k RAM disk and GM873 2 MByte RAM disk
- Drive the BUS at 4 MHz but with the CPU itself running at 10 MHz
- All existing I/O boards to function correctly
- Full Z-80 interrupt operation

SOFTWARE

As there is no specific operating system for this second generation Z-80 device, the 64180 user can currently only use 'good old CP/M'. One could opt for CP/M PLUS, but the only advantage this would give is faster access to files. However, it would cost a lot more than the existing version 2.2. The other alternative is ZSYS which the user can get free from one of the user groups. In some ways it is a more user-friendly product but people used to CP/M might find it unacceptable.

It is also worth noting that CP/M 2.2 takes up less CCP memory than the alternatives which, therefore, gives the programmer the largest TPA possible thus cutting down the need for overlay files.

The on-board memory (1 MByte for a standard card) above 64k will be used as a RAM disk called 'O'. This will be much faster than the 'M' equivalent because the CPU will be running at 10 MHz versus 4 MHz on the GM813. The new BIOS, however, will still support 'M' drives. Another possible application for the large amount of on-board memory could be the track buffering of floppies or Winchesters, thereby giving very fast access to files, or accelerating spooler type applications.

MARKETS

Since the design concept embraces total compatibility with the GM813 Gemini see the following markets for the product:

1) Existing and future MulitNet users for both Fileservers and Workstations.

2) CAD-8 users.

3) Disk conversion products such as the DX-3.

4) ODIN development systems.

5) Computer enthusiasts who have traditionally worked in a Z-80 environment.

6) Turbo charging of packages based upon original Z-80 based software.

With an anticipated speed acceleration of over 2.5 times of the existing Z-80, new systems based around the GM880 are going to be able to run almost as fast as an AT clone, and if you take into account the speed advantages that are added to a system by the SVC and the RAM disk the combined effect will be to provide a very competitive alternative to even the latest 80386 type products.

Look out IBM, Compaq, et all

PRICING

The end-use price of a board with 1 MByte of RAM will be £450.00.

AVAILABILITY

The design should be completed by the end of July, and Gemini would anticipate a further 3 months will elapse before it can go into production.

BENCHMARKS

	GM813	GM880/4	GM880/8	GM880/10
SWEEP w. verify from M0: to M1:	71.22	65.40	33.04	26.68
CRC *.*	30.54	28.53	14.42	11.67
M80 - BIOSF	78.94	70.79	35.51	28.46
PCW BM1	1.12	1.01	0.52	0.41
PCW BM2	3.90	3.54	1.82	1.51
PCW BM3	10.10	9.17	4.69	3.78
PCW BM4	9.00	8.12	4.20	3.39
PCW BM5	9.68	8.66	4.46	3.66
PCW BM6	18.07	16.15	8.22	6.66
PCW BM7	29.29	25.04	13.13	10.57
PCW BM8	5.42	4.87	2.52	2.13

Preview of the GM890 Z280 Processor Board by Gerry Dickson

This is a preview of the GM890 processor board, from Gemini. It uses the state of the art Zilog Z280 microprocessor, a super-updated version of the Zilog Z80.

I have used the board for 3 weeks. I persuaded Gemini to let me have an advance version of the board. The board is an 8" x 8" 80-BUS board. The board I have does not have the usual solder resist and contains some wire link modifications that Gemini say is for multi-processing.

I am told that the final version, due in August, may be slightly different from this one, but I must say that I was really impressed. The speed at which the board operates is so much faster than the Z80 and there are numerous extra facilities. One thing I must point out straight away is that this board is fully compatible with the existing Z80 boards, so it can be plugged straight into your 80-BUS system, in place of a GM811 or GM813, and be in use immediately.

WHAT IS ON THE BOARD ?

- The board was supplied with 256K of Dynamic RAM and a Boot EPROM. . There are four 32 pin sockets on the board designed to accept EPROMs or static RAM ICs. Link options allow the eventual use of the complete range of memory ICs up to 1Megabyte (278192 ?). Each socket can optionally be battery-backed from the on-board battery.
- The board can be supplied fitted with 1MByte DRAMs, if required. Being 18-Pin DRAM sockets, 4MByte DRAMs may be fitted when available.
- Battery backed real-time clock, using the 146818 clock IC.
- Gemini compatible PIO, using a fast Z80 PIO. Pin outs are identical with the GM811/GM813 and it is both hardware & software compatible.
- Gemini compatible UART using the WD8250 UART. Pin outs are identical with the GM811/GM813 and it is both hardware & software compatible.
- A second UART, within the Z280.
- Four, DMA Channels, Three 16-bit counter timers, 256 Byte cache, Dual (System/User) addressing modes, separate 64K code/data/Op system.

THE Z280 PROCESSOR

The GM890 processor board contains a "state of the art" Z280 processor. This CPU, from Zilog is 100%, upwards code compatible with the Zilog Z80 and the Hitachi HD64180, (also called Z180). The Z280 supplied on the board was a 10MHz version. Versions up to 25MHz are also supported on the board. The Arithmetic Logic Unit (ALU) is now completely 16-Bit.

I found that "pure processing", Z80 code ran approximately 3 times faster than the GM813. A speed improvement of 300% !!!!

USING THE GM890

The board was plugged into a standard Gemini Galaxy in place of the GM813. I found that all my existing applications software worked at vastly increased speed. Pascal compilation speed was amazingly fast. The screen displays come & go immediately. Compilation to the "M" drive was around 3 times faster than the same operations on the GM813.

Both printer & serial ports were checked OK to both types of printers.

NEW INSTRUCTIONS

I tried out a few of the enhancements, using both Turbo Pascal and the GemZap assembler. The additional instructions were easy to try out, using "DB" in Gemzap and "In-line" code for Turbo Pascal.

- MULTIPLY and DIVIDE instructions are included, for both signed and un-signed bytes & words.

- 16-BIT INDEXED where an instruction such as [IX+d] has d as 16 bits, in addition to the 8-bit usual Z80 instruction. This instruction works with the HL, IX and IY registers and is useful for indexing larger (over 256byte) arrays of data. Indexing on HL is new.

- PROGRAM COUNTER RELATIVE addressing, for example,
  Ld A,[PC+d] where d is 8 or 16 bits.

- STACK POINTER RELATIVE addressing is useful where the stack contains a data array of known format, passing parameters on stack etc. Previously to access a position on the stack, the SP had to be added to, or subtracted to point to a byte, modified, then the reverse operation to the stack was required. The same operation now takes one instruction to load or save to somewhere in the stack (8 or 16 bit displacements). This addressing mode alone will dramatically speed-up any (Z280 specific) compiler, where the method of parameter passing is usually on the stack.

- BASE INDEX addressing allows the effective address of the operand to be computed by adding the contents of any 2 of the HL, IX or IY registers. For example Ld A,[HL+IX].

- EXCHANGE H with L. The contents of the H and L registers can be exchanged with one instruction (EX H,L). Possible uses include correcting the significance of the MS & LS bytes of a word. Without this added instruction, three instructions are required and a third register.

- WORD COMPARE instructions e.g. CP [HL],BC simplifies this operation from 5 instructions to one.

- EXTEND SIGN extends the sign of a byte into another register, correctly converts a signed byte to signed word.

- SYSTEM CALL NN instruction. The CPU jumps straight to the address found at the contents of the "Interrupt/Trap vector Table pointer" +50H. the call number (NN) is left on the stack. Return to the user program via the new "RETIL" (return from Interrupt long) instruction. This is similar to the software interrupt on 8088 based machines. It enables an operating system to have a standard set of calls for all Op. System functions.

- WORD I/O allows IN[W] HL,(C) and OUT[W] (C),HL to input/output 2 consecutive bytes to/from the HL register. Also INDW & INIW, input word to (HL) and decrement (or increment) HL twice.

- INDRW, INIRW, OUTDRW & OUTIRW input/output word to/from (HL), HL+/-2, B-1, until B=0.

- TEST I/O DATA TSTI (C) allows an input port to be tested, without actually inputting data. This is very convenient, and fast, for say "Test keypress" type instructions, where the input data can be (non destructively) tested. With the Z80, we had to do a real input of the data, then store it somewhere.

There are other commands for "Extended processor Architecture" (EPA), which apparently extend the CPU instruction set, but I have not looked into these yet.

MEMORY MANAGEMENT

The Z280 supports 16MBytes of memory. This is fully usable for new programs because of the memory management unit (MMU). The MMU uses 12 bits to map each 4K within the 16MBytes addressing range.

The MMU translated "logical" addresses into "physical" addresses. A concept used in all the modern, advanced operating systems.

On startup the MMU passes logical addresses to physical addresses without translation. The boot EPROM is at physical location 000000 onwards. On boot-up it maps DRAM into the bottom 256KBytes, allowing all standard CP/M programs to execute immediately.

Being of 4K block size, it is conceivable that it could "emulate" the existing Gemini page mode structure, if required.

SEPARATE USER/SUPERVISOR MODES

This is very important when advanced, fully protected operating systems are used. The User program is usually run in "User" mode. All the special ("Privileged") instructions are only allowed in "System" mode. This prevents a user writing code that causes a "whoopsie" in the system. In user mode, the programmer is fully protected from modifying someone else's data, or code. A program jump to somewhere unknown does not crash the system. Instead, the system can return a message such as "Not allowed, please try again... etc.).

System mode is used for all operating system code, where the special instructions are all available (and hopefully de-bugged). Any transgressions by the user program are reported to the system by "Trap" (see below).

An interesting by-product of this is being able to translate memory & I/O port addresses, all completely transparent to the user. If bit 2 of the trap control register is set, then any direct I/O in user mode causes a "Privileged Instruction Trap". The CPU jumps straight to the address found at the contents of the "Interrupt/Trap vector Table pointer" +54H. Here we do the direct I/O using an I/O conversion table, then return to the user program, using the new "RETIL" (return from Interrupt long) instruction.

Similarly, the MMU can be "write protected" at a block of memory. Any access to this block of memory causes a "Page fault Trap". The CPU jumps straight to the address found at the contents of the "Interrupt/Trap vector Table pointer" +4CH. Here we do the required address translation, or anything else required, all transparent to the user, then return to the user program using the new "RETIL" (return from Interrupt long) instruction.

It should be possible, using this method, to simulate any Z80 system. For example, Nascom 1/2 code can be made to run directly on the GM890 board, using the required I/O translation table and screen driver.

256 BYTE CACHE

This I found was an extremely useful feature which considerably speeded up the running of the program when it was in a short repeating loop. The cache is transparent to the user, it continually loads instructions and data from the main memory during any free time the processor might have. This means that when the processor is ready to process its next instruction it takes it from the cache rather than main memory thus cutting down on the number of external bus accesses.

The cache may be enabled for data, instructions or both, depending on bits 5 and 6 respectively of the "Cache Control register". On reset, the cache is enabled for instructions only. The PCACHE instruction (purge Cache) invalidates the 256 byte on-chip cache. Instead of using the fast memory as a cache, the 256 bytes can be used as 256 bytes of extremely fast physical memory, no bus access is required.

SEPARATE BLOCKS FOR PROGRAM, DATA & OPERATING SYSTEM

I have found when programming that due to the size and amount of variables I use that the data takes up quite a lot of the 64K into which my compiled code has to fit. Sometimes the data takes up as much as half and then I still have to allow room for the run-time package. This then does not leave a great deal of room for the program and what I generally have to do is split the program into numerous parts and use overlays or chain to separate programs. How much better I have found this new board, which allows me to have separate blocks each of 64K for the program, data and operating system. This board also allows me to have more than one program block each using the same data block. This feature is not enabled on reset. It must be enabled on the MMU.

THE INTERRUPT/TRAP VECTOR TABLE

The I/T vector table consists of pairs of Master status register and Program counter words, one pair for each interrupt or trap source.

The CPU contains an additional register, the "Interrupt/Trap Vector Table Pointer". This contains the top 12 bits of the physical address of the contents of the "Interrupt/Trap Vector Table". Any interrupt or trap, finds the address of its service routine from this table.

INTERRUPTS

The Z280 processor has a compatible interrupt structure with the Z80, in modes 0,1 and 2. Both maskeable and non-maskeable interrupts are completely compatible with the Z80.

In addition to these original modes, Zilog have added a fourth, more advanced interrupt mode, IM3. This is upwards compatible with the Z80. This mode is similar to mode 2, except that instead of saving only the return address, also the "Master Status Register" and 16-Bit "Reason code" are saved. A new "Master Status Register" and Program counter are fetched from the "Interrupt/Trap vector table". The "Reason code" is the vector address of the interrupting device. The contents of the "Master Status Register" decides, amongst other things, the interrupt levels that are enabled and whether the interrupt routine is user or System mode.

A further enhancement is that there are 3 (Maskeable) interrupt lines to the CPU. This is a binary coding of 7 different interrupt levels. Each level can be separately enabled or disabled (by 7 bits in the MSR). This means that certain interrupt devices can optionally interrupt lower-level interrupt routines. This provides a nested interrupt structure, nice.

There is a separate stack used to store the return addresses of an interrupted code. This keeps the user stack free for use within the program. No problem any more with user stack space, its just not used at all.

I remember the old Nascom monitor (Nas-Sys 1) had a problem, where data was left vulnerably (bad programming) on the stack. Any interrupt overwrote this data producing a bombed-out system. This would not happen on the Z280. Perhaps Nascom should have used the Z280 for the Nascom (you can interrupt me if you like, but don't expect me still to be there when you return) 2 monitor.

Also, unlike the Z80, Block moves, Block compare and Block I/O can all be interrupted before the end of block.

TRAPS

Traps are Software Interrupts, similar to the external interrupts, except that they originate from the code or condition code. Eight trap are implemented for Single-step, Breakpoint-on-halt, Divide by zero, Stack overflow, memory transegression, system call and privileged instruction.

64K OF I/O PORTS

Paged blocks of I/O addresses, giving 256x256 or 64K ports. That's probably enough for most of us. Again, cleverly completely compatible with existing Z80 code.

Access to more than the usual 256 ports is (yes..you guessed it) via the processor "I/O Page register". Pity that 80-BUS boards only decode the bottom 8 bits. What about decoding whole racks though !!!.

AND THE REST...

In this short (?) overview of this board, the Editor would not let me mention in detail the 3 16-bit counter/timers, Full duplex UART, 10 bit DRAM refresh, Multi CPU design, 4 DMA channels, Auto program bootstrap via UART, Dual Stack pointers (User & system), Single Step operation, etc.

CONCLUSION

I have found the board to be an excellent addition to my system, akin to putting a Concorde afterburner on my ageing MG.

Gemini tell me that the new version will probably have multi-processing capabilities, allowing up to 4 GM890s on the same bus..

I have just read that a Z280 assembler is available. Named the XZ280, it supports all Z80 instructions, together with all additional Z280 instructions. The assembler is available from Real Time Systems, in the Isle Of Man. Phone 0624-26021.

Letters to the Editor

In my opinion

Dear Sir,

With due regard for those of the old Nascom/80-BUS congregation who have 'defected' (evocative word, that!) to the IBM style camp, I feel I must finally rouse myself to air my views on some of the points raised, and state my views on the future direction of this magazine.

I have come up through the full range (10 years you say?!) from hand-built Nascom 1 running Nasbug T2 and tapes without either reliability or named files (let alone a directory), up to Nascom 2 running CP/M 2.2 and CP/M Plus with dual floppies and 512K RAM. OK, so that's not a lot nowadays, but I have come to appreciate both the general quality of the hardware / software and the fact that information is actually available on the products - e.g. circuits and listings - at least from Nascom, Gemini and Map-80. This has allowed me to both repair the system, and also modify it to my own needs. This ain't generally possible with IBM or compatibles.

CP/M was written off years ago by the pundits - before I could actually afford it - and it is still criticised today as being unfriendly. This can be laid firmly at the door of the relevant CBIOS author, BUT Gemini and Map-80 CBIOSs are excellent. I wish others were half as good. Even so, if CP/M is unfriendly, these critics have never lived !!! In comparison to some alternative OSs it is worth its weight in gold not to overwrite the next file on disk without warning because saving a longer file with its old name results in a rewrite... or require that a disk be 'compacted' to remove 'holes' between files because files must be saved contiguously. With CP/Ms security and the added friendliness of the Map-80 or Gemini CBIOSs, it's great!

Whether or not the Amstrad CP/M machines are any good or not I cannot say, except that their effect has been to reinject life into CP/M. This can do nothing but good. I have a large (for me) investment in Z80 (ask the wife!) and CP/M in particular, and do not see any reason why I should scrap my investment in hardware, software, time and knowledge while it can do 90% of what I require. The other 10% unfortunately has to be done on the office IBM-XT (256K) - I don't have dBASE II - and experience has shown that it will require an awful lot of work and money to 'upgrade' to IBM and still be able to do the same things. I know it's supposed to be faster, but it needs to be with the overheads.

By all means (if there is no alternative), let us include details of how MS-DOS etc. works (we may all have to go that way one day) but why stop there? MS-DOS is from Microsoft, but IBM has announced the advent of OS-2 which will carry their PCs into the future. Will MS-DOS emulate the Dodo ? Whatever is decided in the editorial ivory halls - well ceramic anyway - PLEASE PLEASE PLEASE DON'T let it become a foot in the door to oust coverage (in depth) of 80-BUS products, systems and software (especially CP/M) from any manufacturer. This *Scorpio News* is the latest in a long line of Nascom/80-BUS related publications, and I, for one, would be most sorry to see it die.

I would also like to see more mention of other 80-BUS manufacturers in your pages - Map-80 for instance have always given me most courteous and helpful service, but in the previous publications it seems that their name has only appeared almost by accident... are they in from the cold at last??

So much for the past and the future. As for the present, it is a refreshing change to get a regular magazine, with hard information not waffle - Mr. Waters series on disk directories is excellent, plus the notes on ZCPR3 installation. Try finding that lot in any book with such detail!!!

As you say, the magazine can only survive if more material is contributed by others. To this end, please find a couple of enclosures for your consideration. Please keep up the high standards and don't desert the Z80 / CP/M population - even if I AM biased. I would be prepared (just!) to pay more for a magazine which suits my requirements, but only if more loot results in more info....? It seems that in the previous experiment, printing two pages on one was not popular, but I didn't find it any problem - as long as the print quality remains high. (It could have been printed slightly larger if the surrounding picture frame were narrowed.) The publication frequency is OK. Don't fall into the trap of over-extension and trying to compete with the 'weeklies' - disaster lies down that path. Quality counts. But keep it regular (More prunes, vicar?).

Michael Newson, Banbury, Oxon.

Yes - let's disassemble clones.

Dear Sir,

On the general point of whether to move the mag. to support the IBM PC world, as Dave Hunt suggests, I would agree with him, though I should state my bias as an owner of both clone and Gemini/Nascom hybrid.

The expertise of the early group of enthusiasts who built their own machines and managed to squeeze programs into 1k of memory is still valid, and perhaps more so in the clone world where information is sadly lacking - no circuit diagrams or port maps in the Taiwanese PC world (or even with the Greenock real ones), or very expensive - the DOS technical reference manual cost a staggering £70 or thereabouts. The skill of the average MultiBoard enthusiast, most being hardware-software hybrids, should, put together, rapidly disassemble the hidden store of information on these machines.

Finally, our beloved 8 bit CP/M machines are long in the tooth, and though superior to 16 bit machines when dealing with text applications, we will have to admit they are coming to the end of their golden age.

Doug Taylor, London.

Ed. - some "compatibles" DO come with circuit diagrams, see the review in the last Scorpio News, although I understand that most do not. As to your beloved machine coming to the end of its age, perhaps the new product information in this issue will change your mind?

A few points

Dear Sir,

I thought I should write to inform you (and the readership) of a few points which have come to my notice, but before I start moaning I must say how much I enjoy reading *Scorpio News*. It must be just about the only computer / electronics related magazine which does not send me barmy with technical errors.

I have had a rather funny week since receiving the Scorpio Systems SVC-03 disk that I ordered. During the week I found bugs in MLINKS, PCB and finally in my newly acquired copy of HiSoft C, and all this with my untrusty old GM809 FDC board giving up on me. No doubt you will be glad to know that the MLINKS bug is very easily fixed. It relates to the PSI and GSI commands which, on my copy at least, write and read garbage. This is because they assume the default CP/M DMA at 80H without checking that MBASIC has not moved it. MBASIC, of course, has moved it. The solution is quite simple, just use the BDOS function "Set DMA address" (code 26). This can be done at the end of the disk common routines thus only needing to be written once.

Find "discom:" , move down to "endlin:" and find the "ret" instruction at the end (3 lines down) and insert:

```
ld   de,tbuff      ; This is 80h
ld   c,26          ; Set DMA address
call bdos
```

This occupies extra space, but the space can be recovered by removing two unnecessary BDOS calls in PSI. Shortly after creating the new file with:

```
ld   de,fcb
r
ld   c,16h         ; create file
call bdos
```

you will find:

```
ld   de,fcb
r
ld   c,0fh         ; open file
call bdos
```

This is entirely superfluous as the file is opened automatically by the call to create. This occurs twice (very inefficient programming).

The bug in PCB is also easily fixed if you have a PASCAL compiler, which I don't. I did get round this as I shall explain later. The bug here is far more obscure. In track delete mode, mark one end of a track twice and you will find that the track is deleted (although it stays on the screen). If several tracks meet at the point you mark, you may expect them all to go, but actually only the first one defined is removed. This can be fixed simply by rearranging the list of conditions in a very long "if" statement, which I have no intention of copying out here.

I reported the HiSoft C bug to HiSoft, who replied quickly explaining that it is CP/M which is actually at fault. It relates to the *keyhit()* function which does not work correctly with *rawin()* due to the lack of direct console status checking in CP/M 2.2. Other than that I have found this to be a very good product despite the fact that I don't possess Dr Dark's library. In fact I have no books on C at all. This is my normal modus operandi, having learned BASIC from some example programs (and the Nascom manual), Z80 machine code from the Mostek technical manual (an excellent document, but they would never claim it to be tutorial) and PASCAL from nothing (we didn't even have the manual (Oops!)). I must admit I failed to learn ALGOL this way but I blame that on not having the runtime package. I am learning C by the process of translating the PCB program from PASCAL. This is not simply to learn C, nor is it to fix the bug (though I have done that). I find it entirely inadequate that it has no hard-copy options, so I am writing those into my C version as well as catering for larger boards. When I have finished it I may well make it available for a small sum.

While on the subject of learning to use things, I must disagree with Dr Dark regarding Gempen and Wordstar. I have used Gempen a bit but found it most inconvenient. Wordstar, however, seems fairly self-explanatory by comparison, and the main reason for the two key commands such as ^KQ is to allow you to type text. As I recall it is necessary to exit from typing in (with ^Z) before Q will work on Gempen, thus making that a two key sequence also. Besides which, on machines like recent Galaxy's there are function keys which can be set to do that. And as regards 32 Megabyte files, I have frequently exceeded 50K or so and more than once I have gone over 64K with both documents and source files. I agree WordStar costs a lot, but it is a top quality product and you get what you pay for.

Regarding the Nascom anniversary "happening", I would like to see something "proper" (i.e. a bit more than a few people in a pub) but I would not wish to see the cost get beyond the less wealthy among us. Equipment demos. would be nice.

As to I.B.M.s in *Scorpio News*, it's probably a good idea providing they are treated differently from the "colourful comics" (i.e. like you already do with 80-BUS stuff). The I.B.M. is rather less of a "black box" than most but generally still rather more of one than 80-BUS equipment, it would be nice to see it "exposed".

Yours faithfully, R. Pearce, Chadlington, Oxon.

RAM-disk makes life bearable

Dear Sir,

I totally disagree with Dave Hunt's comments in Issue 1, p35 about the general uselessness of extended memory - either as pageable executable RAM, or as virtual disk (RAM-disk). I happen use two MAP 256K RAM cards (=512K) as virtual disk with a Nascom-2 (32K paging), and find it almost indispensable.

I have never written - or intend to write (for the Z80) single programs which exceed the 62K TPA, but its primary function IS as a RAM-disk. It's not a case of having something 'bigger than you' (I don't know any other Nascom users around here), it's the fact that when editing large text files e.g. source code or data lists, the whole file can be held in RAM at once, allowing editing to proceed without disk waits. Wordstar's swapping overlays in and out is unnoticable. I find waiting for disk access tiresome nowadays - I suppose it wouldn't be such a problem if I was running a winchester like D.H.

The speed of assembly and compilation is also greatly improved (with the output to RAM-disk, naturally), and making multiple copies of a disk is a snap when the whole disk can be held in

memory. As for software support, this is taken care of by the CP/M BIOS - I don't have to do anything extra to use drive P:.

The only problem with RAM instead of disk is that it is volatile. You must obviously remember to re-save to disk all files of interest before switching off or suffering a power cut !!!

D.H. may hardly ever use his virtual disk, but for me it makes life bearable.

Michael Newson, Banbury, Oxon.

BDOSZ Fixes

Dear Sir,

Many thanks for publishing my last letter in which I offered free upgrades for BDOSZ. The response to the letter was absolutely overwhelming, and all four people should have their updated copy by now. Since that last letter, BDOSZ has now had a couple of minor alterations and one small bug fix, and the version now stands at 1.92.

The bug caused data written to a previously allocated sector to be forgotten when the file was closed. A typical example was under ZAP or SPZ where the first record of a file was patched and ZAP exited. Some users found that the changes hadn't been written to disk. The problem was that although the data was sent to the BIOS, it remained in the blocking/deblocking buffer. The BDOS wasn't causing the buffer to be flushed when the file was closed, and the buffer was discarded on warm boot.

The bug fix is published below and is valid, certainly, for all versions from 1.6 onwards. The unmodified code is shown in full. The change involves deleting those lines indicated '<--- '. My thanks to Philip Sherlock of Kenilworth Computers for bringing this bug to my attention.

```
WROUTS:
        CALL    CVISTS          ; CONVERT TO INTERNAL SECTOR NUMBER
        POP     BC              ; RESTORE ALLOC FLAG
        PUSH    BC              ; SAVE IT               <---
        CALL    WRSECT          ; WRITE NEW ONE OUT
        POP     BC              ; RESTORE ALLOC         <---
        LD      A,(IX+F.CR)     ; GET THE CURRENT RECORD NUMBER
        CP      (IX+F.RC)       ; EXCEEDED?
        JR      C,TSETWR        ; J IF NOT
        LD      (IX+F.RC),A     ; SET THE NEW VALUE
        INC     (IX+F.RC)       ; UP BY ONE
        LD      C,002H          ; SET DUMMY UNALLOC     <---
TSETWR:
        DEC     C               ; TEST IF UNALLOC       <---
        DEC     C               ;                       <---
        JR      NZ,TMAXRC       ; J IF NOT ALLOCATED    <---
        RES     7,(IX+F.S2)     ; FLAG A PENDING WRITE
TMAXRC:                                                 <---
```

For those who obtained an upgrade to version 1.8 (the last version issued to Henry's Radio for distribution) and who are wondering whether they have missed out, versions 1.9 and 1.91 only added options to reverse the effect of the Backspace and Delete keys or to make them both Backspace. They also offered improved Control-S handling so that the Control-S is no longer ignored if there is already a character waiting in the BDOS character store (try pausing M80 if you have it sending the listing file to the screen if you have already typed another character). The penalty for this second improvement is that the BDOS will now steal type-ahead characters - this is why the feature is optional.

My offer is still open for those who have yet to upgrade their versions of BDOSZ. Please send your BDOSZ distribution disk to the address below, enclosing your name and address together with U.K. postage stamps to cover the cost of returning it to you. Overseas readers may send a cheque payable to M.W.T. Waters.

Yours sincerely, M.W.T. Waters, 7 Trenchard Close, Stanmore, Middx. HA7 3SS.

Disk Formats and CP/M Disk Routines - Part 4 by M.W.T. Waters

This is the final part in this series. But first of all here is a correction to the last part, part 3, published in Volume 1, Issue 3, page 11:

In the article, the author had a brainstorm and said that the directory buffer (DIRBUF) under CP/M 2.2 is one physical sector long. This, of course, is quite incorrect. The length of this buffer is one logical record (128 bytes) long. The description of the directory buffers controlled by the Buffer Control Blocks under CP/M Plus IS correct, however. CP/M Plus, as stated in the article, controls buffers, each of which is one physical sector long.

CP/M 3 Buffer Control Block format

The format of a Buffer Control Block is given below. The format is identical for both directory and data BCB's.

```
DRV     ;  8 bits - Source drive of buffered data
REC     ; 24 bits - Record Position on disk
WFLG    ;  8 bits - Flag indicating unwritten data
00      ;  8 bits - BDOS scratchpad
TRACK   ; 16 bits - Track number for buffered data
SECTOR  ; 16 bits - Sector number for buffered data
BUFFAD  ; 16 bits - Address of buffer for this BCB
BANK    ;  8 bits - Bank no of buffer for this BCB
LINK    ; 16 bits - Address of next BCB in list
```

DRV indicates the source drive from which the contents of the buffer located at BUFFAD was obtained or, alternatively, the destination drive for the data in the buffer.

REC contains the record position of the current buffer contents. This figure is an absolute physical sector number relative to the first sector in the data tracks of the disk. For example, the first sector in block 0 (i.e. the first directory sector) is numbered zero, the next is number one and so on for all data blocks on the disk.

WFLG is set by the BDOS when the buffer associated with the BCB contained new data that has yet to be written to the disk. When the data has been written, this flag is then set to zero by the BDOS.

TRACK is the physical track location of the contents of the buffer.

SECTOR contains the physical sector location of the contents of the buffer.

BUFFAD contains a 16 bit address showing the location of the physical sector buffer associated with this BCB.

BANK contains the memory bank number holding the buffer. Naturally, this field is not present in non-banked systems.

LINK contains the address of the next BCB in the linked list. If this is the last BCB in the list, this field will contain zero. This field isn't present on non-banked systems as only one buffer and associated BCB exist.

Having covered the DPH, XDPH and associated data structures, we can now examine the Disk Parameter Block in detail.

Disk Parameter Blocks

The CP/M 2.2 DPB

The disk parameter blocks in the BIOS tell the BDOS all it needs to know about the physical characteristics of the disk. The CP/M 2.2 DPB for a Gemini QDDS format disk looks like this:

```
Hex    Dec    Interpretation
0028H   40  ; SPT - CP/M Sectors (128 bytes) per track
  05H    5  ; BSH - Block Shift Factor
  1FH   31  ; BLM - Block Mask
  03H    3  ; EXM - Extent Mask
00C4H  196  ; DSM - Disk Size in blocks-1
007FH  127  ; DRM - Number of directory entries-1
  80H       ; ALO - Reserved directory..
  00H       ; AL1 - ..blocks
0020H   32  ; CKS - Checksum Size
0002H    2  ; OFF - Number of system tracks on disk
```

The abbreviations used are those used by Digital Research in the CP/M 2 Alteration Guide. Decimal values are given where appropriate. Note that CP/M 1.4 doesn't use Disk Parameter Blocks and that the DPB for CP/M 3 contains extra information about physical sector size.

CP/M uses the value SPT to calculate the track and sector numbers for a particular disk data block. The block number and an offset showing where in the block the required record lies need to be converted so that the appropriate track and sector can be accessed. SPT is determined by the physical disk format. Gemini in the QDDS format have 10 sectors of 512 bytes on each track. CP/M uses 128 byte records and so 40 CP/M sectors will fit on a track. The Gemini SDDS format disks use a figure of 18 for SPT while the Gemini DDDS format uses 80. The latter figure is possible because of the method of access to the disk for the DDDS format. Some explanation seems in order.

Cylindrical and Track methods of disk access

There are two methods of accessing disks. The first method is to start on side 1 of the disk and number the tracks from say 0 to 79 (QDDS) then flip sides and call the tracks 80 to 159. Note that this is just for convenience as the tracks on side 2 of the disk are also physically numbered 0-79. As far as CP/M is concerned, it is seeing a single sided disk with 160 tracks. This is known as the Track method of accessing a disk.

The second method is to have the tracks on both sides of the disk numbered from say 0 to 34 (DDDS) and having filled track 0 on side 1 we flip sides and continue writing to track 0 on side 2 of the disk. In this case CP/M thinks it seeing a single sided disk with 35 tracks but the tracks are twice as long as before (i.e. 80 SPT). This method of access is known as the Cylinder method. You should note that the Gemini Quad density format is a misnomer used by a number of companies. In reality, it should be called 80 track double density or double density with double track density.

BLM, BSH and EXM

The block mask (BLM) is basically one less than the number of 128 byte records that will fit in one block. Our figure of 31 means that 32 records will fit in the block. This enables the BDOS to calculate the block size as, you will have noticed, it isn't given in the DPB. If we take the current record count from the FCB and logically AND it with the block mask, we will obtain a number in the range 0-31 which tells us where in a block the particular record may be found. Take, for example, that the current record count is 92 or 01011100. If we AND that figure with 31 or 00011111 we get the result 00011100 or 28. This indicates that the record occupies the 29th (0-31 remember) record position in the block.

The extent mask (EXM) is one less than the number of extents that may be controlled by one directory entry. This depends upon the maximum disk size as, if 16 bit values are recorded in the directory, then each directory entry will refer to half the number of extents than it would if 8 bit entries were being used. In the case of the QDDS format, this value is 3 which indicates that 4 extents may be recorded in a single directory entry. When used as a mask by ANDing the extent byte of the directory entry with EXM, we produce a number in the range 0-3 which tells the BDOS which extent we are referring to in the entry.

The next field is the Block Shift Factor (BSH), and is used by the BDOS when converting CP/M records and extents into block positions in a directory entry and vice versa. BSH is defined as the logarithm base two of the block size in 128 byte records or, LOG2(BLS/128) where LOG2 represents the binary logarithm function. That last bit is more or less a direct quote from Digital Research and probably means as much to you now as it did to me initially.

A beginners guide to base 2 logarithms

For those of you who, like me, left school with Maths 'O' level or alternatively finished school a long time ago (or both), here is a quick refresher. The notation that I shall use to represent powers is borrowed from Microsoft MBASIC. e.g. ten squared will be shown as 10^2 while ten cubed will be shown as 10^3.

Consider the following:

Decimal	Binary	Equivalent
1 = 10^0	1 = 2^0	= 1
10 = 10^1	10 = 2^1	= 2
100 = 10^2	100 = 2^2	= 4
1000 = 10^3	1000 = 2^3	= 8
10000 = 10^4	10000 = 2^4	= 16
100000 = 10^5	100000 = 2^5	= 32
1000000 = 10^6	1000000 = 2^6	= 64

These are simply powers of 10 and powers of 2. The similarities between the two number systems are quite evident and most, if not all, assembly language programmers should have seen individual bits in a byte referred to as powers of 2. The useage of powers of two in all probability explains why the bits in a byte are numbered from 0 to 7 rather than 1 to 8.

The relationship between logarithms and powers is extremely close as can be seen from the examples below:

log10 of 1 = 0	log2 of 1 = 0
log10 of 10 = 1	log2 of 2 = 1
log10 of 100 = 2	log2 of 4 = 2
log10 of 1000 = 3	log2 of 8 = 3
log10 of 10000 = 4	log2 of 16 = 4
log10 of 100000 = 5	log2 of 32 = 5
log10 of 1000000 = 6	log2 of 64 = 6

Logarithms of intermediate values do not fall on a linear scale. i.e. log10 of 50 is not 1.5 but is, in fact, 1.69897 to five decimal places. That is to say that 10^1.69897 = 50. Base 2 logarithms follow exactly the same rules as for base 10 logs. Log2 of 3 is 1.58496 to five decimal places. i.e. 2^1.58496= 3.

The integral part of a logarithm is known as the characteristic and the fractional part is known as the mantissa. For the purposes of explaining BSH, only the characteristic of the base 2 logarithm is required. To simplify matters further, Digital Research have kindly done the calculations for us so if you came out of the last couple of paragraphs with a headache, don't worry.

How BSH, BLM and EXM are used

For completeness, I have copied the tables for the values of BLM and BSH provided by Digital Research. The values, as may be expected, are dependant upon block size (BLS).

BLS	BSH	BLM
1024	3	7
2048	4	15
4096	5	31
8192	6	63
16384	7	127

Having obtained values for BSH from the table (or by calculation), what are they used for? Below is a simple program in BASIC that will calculate X/256 with a remainder. This program is normally used to calculate values for a 16 bit POKE (i.e. to two consecutive memory locations) where X may contain any value between 0 and 65535.

```
10 INPUT "Address to be POKED";ADDRESS
20 INPUT "Enter value to POKE";X
30 P1 = X AND 255
40 P2 = INT(X/256)
50 POKE ADDRESS,P1
60 POKE ADDRESS+1,P2
70 END
```

If we analyse this program and take the binary equivalents of the numbers, this program explains, in precise terms, the relationship between BLM and BSH.

Imagine that the number entered for X is 273. Follow the sequence below:

```
0000000100000111        ; 273 in binary (16 bit)
0000000011111111        ; 255 in binary
0000000000000111        ; 7 (Result of logically ANDing the two
                        ;    values together) = P1

0000000100000111        ; 273 again
       "                ; This bit has the value 256 (2^8)
0000000000000001        ; 1 (Result of integer dividing by
                        ;    256) = P2
```

In the division, the 256 bit (2^8) has moved to the 1 position (2^0). In effect we have right shifted the number 8 places. Another way of looking at it is that we have right shifted the number log2 of 256 places.

In the example DPB given, BLM has a value of 31 and BSH has a value of 5. Their use is exactly the same as in the BASIC program.

Imagine that we wish to read a 128 byte record from disk and that record is number 43 in extent 18 of the file we are reading. Assume that the correct directory entry is available and that it looks something like the one given below:

```
00544553 54202020 20545854 13000080   *.TEST    COM....*
202E2F30 31323334 35363738 393A3B3C   *-./0123456789:;<*
```

We shall use the values for the Gemini QDDS format in this example and so we can see that the directory entry controls four extents. For convenience we shall number the extents 0 to 3. Extent 0 has blocks 2D, 2E, 2F and 30 allocated to it; extent 1 has blocks 31, 32, 33 and 34 and so on for the remaining two extents. Within each extent, we shall also number the block positions from 0 to 3 for convenience.

Since we are using a 4K block size, each block will contain 32 CP/M records of 128 bytes each. We shall number those records 0 to 31.

If we logically AND the required extent (18) with the extent mask (3) we obtain the value 2. This means that the record we require is contained in the extent numbered 2 in our directory entry. i.e. the record is held in one of blocks 35, 36, 37 or 38.

If we now take the record number (43) and right shift it BSH (5) times, we are left with the value 1. The block in position 1 is in fact block number 36 on the disk.

All we need to do now is to determine where in block 36 the particular record we require is located. This is achieved by ANDing the record number with the block mask. The result of ANDing the record number (43) with BLM (31) is 11. Therefore, the record we require is the 12th record in the block.

Back to the DPB

The value DSM contains the number of blocks on the disk less one. Another way of looking at it is that it holds the number of the highest block since the blocks are numbered from 0 to DSM or, in our case 0 to 196 (0-0C4H).

DRM contains the number of directory entries on the disk less one. Again, the directory entries are numbered from 0 to DRM (0-127 or 0-7FH for the QDDS format).

AL0 and AL1 contain the first two bytes of the allocation vector for the drive for which the DPB applies. These two bytes are copied into the allocation vector when the drive is logged in. Remembering that the directory occupies the first block or two on the disk, we need to reserve the required number of blocks by filling in the appropriate bits in the allocation vector. In our case, we only need to reserve one block and this has been done by setting bit 7 of AL0, hence the value 80H. Clearly, we can now see that a maximum of 16 blocks may be reserved for directory entries (by setting all 16 bits in AL0 and AL1). With a 1K block size, a maximum of 512 directory entries may be allocated at 32 directory entries in each block. If a 16K block size is used, the maximum number of directory entries will now be 8192.

CKS is known as the check size and simply tells the BDOS the length of the checksum vector for the currently logged in drive. If fixed media drives are in use, the value of CKS may be set to zero to prevent directory checking.

OFF is the number of system tracks on the disk. This figure depends upon the physical disk format and the CP/M system size. Clearly, a single density disk will require more system tracks than a double density disk in order to save the system. As an example, the SDDS format has 3 system tracks, the QDDS and QDSS formats have 2 and the DDDS format has 1. The discrepancy between the QD and DD formats is again because of the apparent length of the tracks in the DD format which uses the cylindrical method of disk access. CP/M uses the OFF parameter when calculating track numbers for a pending read or write so that it effectively ignores the system tracks. This is particularly useful when partitioning a high capacity drive (i.e. a Winchester) into a number of logical drives. If, say, the first 50 tracks are defined as logical drive A, logical drive B could have a value of 50 for the OFF parameter so that it would start at the 51st track of the physical drive.

The CP/M 3 DPB

Now that we have exhausted the CP/M 2.2 Disk Parameter Block, we can now examine the CP/M 3 DPB. An example of the latter is given below:

```
Hex    Dec    Interpretation
0028H   40  ; SPT - CP/M Sectors (128 bytes) per track
  05H    5  ; BSH - Block Shift Factor
  1FH   31  ; BLM - Block Mask
  03H    3  ; EXM - Extent Mask
00C4H  196  ; DSM - Disk Size in blocks-1
007FH  127  ; DRM - Number of directory entries-1
  80H       ; AL0 - Reserved directory..
  00H       ; AL1 - ..blocks
0020H   32  ; CKS - Checksum Size
0002H    2  ; OFF - Number of system tracks on disk
  02H    2  ; PSH - Physical Record Shift Factor
  03H    3  ; PHM - Physical Record Mask
```

A brief examination of the CP/M 3 Disk Parameter Block will reveal that it is identical to the CP/M 2.2 DPB apart from the last two entries so I shall only explain these additions.

PSH is the physical record shift factor and is described by Digital Research as being LOG2(physical sector size/128). For the Gemini QDDS format which uses 512 byte physical sectors, this value is 2.

PHM is the physical record mask and is calculated to be (physical sector size/128)-1.

PSH and PHM are used in exactly the same way as BLM and BSH except that instead of converting CP/M records and extents to block numbers, they are now converting CP/M records to physical sector numbers. Imagine that we have extracted a block and CP/M record number from the directory using BSH, BLM and EXM. The value that we obtained when explaining BSH and BLM were block number 36 on the disk and record number 11 within that block.

There are 32 CP/M records in the block numbered from 0 to 31 as described before. Additionally, we now know that there are eight physical sectors (of 512 bytes each) in the block, each containing

four CP/M records. We can number these physical sectors from 0 to 7 and the CP/M records within them from 0 to 3.

If we shift the record number (11) right PSH (2) times we get the value 2. It is a simple matter to read the third physical sector into the deblocking buffer ready to extract the required 128 byte record.

Finally, by ANDing the record number (11) with PHM (3) we get the value 3. Consequently, the required record occupies the last 128 bytes in the physical sector, i.e. the fourth record in the sector.

Blocking and Deblocking

Isolating separate 128 byte CP/M records in a physical sector is central to the concept involved with blocking and deblocking. All versions of CP/M transfer data to and from the disk in 128 byte chunks (including CP/M 3). We have already seen that disks may be formatted into sectors of varying size and if a sector size of 128 bytes is used then no problems are experienced. However, many disk formats use sector sizes of 256, 512 and 1024 bytes. As we have seen, Gemini use a 512 byte sector size for their DDDS, QDSS and QDDS disk formats. Since CP/M requires 128 byte records, surely it is easier to format the disk in 128 byte sectors.

The answer to this is yes. It is easier to use 128 byte sectors but oddly enough it is generally faster to use larger sector sizes. Why this is will become apparent as we proceed but suffice to say that since we are using sectors that are larger than 128 bytes, we need a method of providing CP/M with 128 bytes at a time and allowing CP/M to write 128 bytes at a time. The processes for writing and reading in this manner are known as blocking and deblocking respectively.

Blocking and deblocking are implemented in the BIOS under CP/M 2.2 and (usually) in the BDOS under CP/M 3 although facilities exist within CP/M 3 to allow the BIOS to do this. The method used requires a buffer in RAM that is one physical sector long (512 bytes for the Gemini DD and QD formats). This buffer is simply an area of RAM reserved by the BIOS for this purpose and is known as the blocking/deblocking buffer. This buffer is located within the BIOS under CP/M 2.2 but under CP/M 3, the physical record buffers controlled by the BDOS (pointed to by the DTABCBs) are used. From now on I shall assume that CP/M 2.2 is in use and will describe the method of blocking/deblocking in respect of the CP/M 2.2 BIOS. Under CP/M 3, the BDOS will have calculated physical sector numbers (using PSH and PHM) and the BIOS reads them directly.

When deblocking, the BIOS takes the sector number provided by the BDOS and converts it into a physical sector number. i.e. the physical sector that contains the required CP/M record. This physical sector is then read into the blocking/deblocking buffer and the wanted 128 bytes are identified and passed to CP/M by copying them to the DMA address. At the time of reading the physical sector, certain facts about it are recorded. These are the drive, track and sector number for the physical sector. At the same time a flag is set to say that the buffer is in use.

The next time that a read is requested by CP/M, the physical sector number is calculated as before. This time, however, the BIOS will find that the buffer is in use. Bearing in mind that most disk reads are done sequentially, it is a fair bet that the CP/M record we want is in the buffer already. The BIOS checks the drive, track and sector numbers for the required sector against those stored for the buffered sector and if they match, the BIOS skips the disk read and simply transfers the required 128 bytes to the DMA address.

When reading sequentially using deblocking, the disk is accessed only every fourth request from CP/M. As disk drives are relatively slow when compared with the time required to extract data from RAM, you can now see why this process is faster than simple disk reads using 128 byte sectors. Generally, the larger the physical sector size, the greater the improvement in disk access time will be.

Writing to disk using blocking offers a similar (but not as great) increase in speed. You will see why writing is not as fast as reading in a moment.

Imagine that we wish to write a random record to disk and that we are updating one record among many adjacent records. Since we have to write to the disk in terms of complete physical sectors, we cannot simply write 128 bytes to the disk. If we put the CP/M record in the blocking/deblocking buffer and wrote that to disk, we would erase the other three records in the physical sector.

When writing CP/M records to disk, the BIOS computes the physical sector number in the same manner as for reading. It then checks to see if that physical sector is already in the buffer and reads it if not. This pre-read of physical sectors explains why the disk performance is not quite as good when writing as compared with reading. However, the performance achieved is still better than when writing 128 byte sectors as the disk is still only accessed twice for every four CP/M records written.

The pre-read may be skipped under certain circumstances. CP/M tells the BIOS that the data being written lies in an as yet unallocated block. If this is so, there is no data in the sector to be destroyed so the read is skipped. If the pre-read is required, before reading the sector into the buffer, there is another check that must be done by the BIOS as we shall see in a moment.

Having read the required physical sector, the buffer in use flag is set and the CP/M record may be put in it at the appropriate place. You will see that the other records in the sector have now been preserved. The BIOS doesn't write the buffer to disk at this point as another write to the same physical sector may be requested by CP/M, so it simply flags this record as containing unwritten data and returns control back to CP/M.

If another write is requested to the same sector, the pre-read of the disk will not be required as the wanted physical sector will already be in the buffer. However, if another write (or read) is requested that involves a different physical sector, the BIOS looks at the flag showing whether the buffer contains unwritten data and if it does, the BIOS now writes it to disk.

The only exception to the above explanation is when directory information is being written to disk. Since the BIOS maintains a separate buffer for physical directory sectors, the changed directory information MUST be written to disk immediately.

THE END !

A Review of two Modula 2s by Doug Taylor

If you liken programming languages to cars, then Cobol is an Austin 7, old fashioned and requiring a lot of maintenance, BASIC is a Fiat 127, noisy, prone to rust and always running out of performance when you need it. Pascal is the Volvo of the programming world, careful, safe and always willing to cruise up the hard shoulder at 90 MPH. Whereas Fortran 77 is a Range Rover, robust go anywhere, do anything, break or remake all the rules language.

So where does Modula 2 fit? Well this is a kit car, built with strength of a Volvo - it is after all a member of the Algol family, a child of Nicklaus Wirth, but has all the versatility of Fortran. Modula 2 has been used by Wirth to correct many of the mistakes of Pascal, the rather fussy and confusing syntax has been tidied up, (I could never remember when I needed a semicolon and when I didn't) and the most powerful feature of Fortran, the foundation in library based code, imported.

Modula 2, as it name suggests, encourages you to write modular code. These modules are combined into libraries, and programs created by importing code into the program from these libraries. The library will usually consist of Modula 2 code, it is a very recursive language, Modula 2 code is made from Modula 2 code which is made from Modula 2 code which is ...., but in most if not all applications, code from other relocatable code producing tools can be linked into the program, e.g. from an assembler or Fortran Compiler.

The two versions of Modula 2 I will review are Turbo Modula 2 from Borland International (only available for CP/M at the moment) and FTL Modula 2 from Workman and Associates (HiSoft in the U.K.) which is available for MSDOS and CP/M, the version reviewed being the MSDOS variant.

The Borland version of the language comes complete with a 544 page manual describing the language, the Wordstar like editor, the shell and the Standard library supplied. The standard library and reference directory to the modules is explained briefly with examples, covering 411 pages of the manual. If you have used Borland's other minor work Turbo Pascal, you will know the high standard of their manuals - all examples in the manual will work and you can learn the language from these alone.

In Modula 2 the definition has been left quite vague. The implementer has to provide certain modules such as system, texts, in/out, files etc., but the exact definition of what these modules contain has been left open. Never the less the spirit of the language is that there should be no code that could not be written by a user of the language.

Here Borland have infringed on this suggestion, in providing procedures *write* and *writeln* which can have a variable number of parameters, i.e. they are like the Pascal equivalents, but in the same way as Turbo Pascal has become the de facto standard for pascal, as it provided users with the extensions they wanted, Turbo Modula 2 may also become a de facto standard for the same reasons.

In the Borland implementation of the language the following simple types are available:

INTEGER	-32,768 to 32,767,
LONGINT	-2,147,483,648 to 2,147,483,647,
CARDINAL	0 to 65,535,
REAL	-6.80565E+38 to 6.80565e+38,
LONGREAL	-3.5953862697246D+308 to 3.5....D+308
CHAR	
BOOLEAN	

In Modula 2 as with Pascal it is possible to define any datatype that can be constructed from simple types. Also in most versions of Modula 2 is the SYSTEM module, or to be more correct pseudo-module, in which version dependent code is located and the dirty tricks definitions are included. In this version the types BYTE, WORD, ADDRESS and PROCESS are defined, the first three are obvious, I hope, but the last is a facility for allowing concurrent processes to occur and allowing transfer of information between processes. An analogy in another high level language is hard to think of, the use of system common, or shared EMA in Fortran is similar but much more limited. To achieve a similar effect in Fortran usually involves direct operating system calls.

In this version of Modula 2, you can supply modules written in Microsoft Macro 80 compatible code, (as .REL files), which can be linked into the main code once the appropriate definition module has been written. The Turbo environment is a particularly friendly environment to work in, you can set various options when compiling, for example you can turn the extensions to the standard off, you can make the compiler ignore case, in standard identifiers (such as END). Range and maths checks can be switched on or off, but the usual comments about wearing a lifebelt on land and taking it off when you go to sea apply in these cases and a listing of compilation can be obtained.

An interesting feature of this compiler is that it may compile to interpreted code or to native code the choice is left up to the user. Usually code is compiled to the interpreted form, for testing and then compiled to native code when fully debugged, but sometimes for compactness of code, it will be left as interpreted code. There are some bugs in the compiler in this area, I have had interpreted code work and native code then produced and turn its toes up.

The linker can produce either a .COM file, or can link together interpreted modules to reduce disk search time. The linker can be controlled in much the same way as linkers on super-minis and main frames, allowing direct formation of overlays and control of initialisation. Remember in Modula 2, each module may have a *main* as well as procedures.

The librarian is also an essential item of any Modula 2 system and the Turbo one is easy to use, quite like LU in its behaviour.

The shell is particularly friendly, allowing access to the editor, compiler linker and librarian, without leaving its environment. Particularly useful to hard disk users is the ease of moving from one user area to another and the ability to produce a directory list across the disk including all user areas. Access to file management utilities such as rename, copy, and delete are also provided.

The FTL Modula 2 (MSDOS version) comes with two manuals, one of 90 pages explaining the compiler in relation to the MSDOS operating system, and the second (of 105 pages) describing the language. It would not be possible to learn the language from these manuals alone, but they do not claim to teach the language and should not be expected to do so. The manual contains some examples of code but these tend to be notes describing features rather than full program code.

In addition to the types described for the Turbo Modula 2 an additional type LONGCARDINAL (32 bit unsigned integer) is available. A slight difference between the two versions is in the range of REALS, FTL reals are described as about 16 digit accuracy with an exponential range of -105 to +105, miserly in comparison with Turbo, but better by a long way than any Fortran I have used. As with Turbo any type can be created from the simple types and BYTE, WORD and ADDRESS are predefined.

A very valuable feature of this version is that all the sources of the modules have been supplied, a feature that enables you to overcome any shortcomings, real or imaginary in this package. In addition the editor supplied is one of the most friendly editors I have used. It is like the Turbo editor in being Wordstar like, but is considerably faster. The Turbo editor often has to perform a disk access to call an overlay, to perform a function such as block moves.

The FTL editor, besides being fast, enables up to three files to be edited at once with the screen split into three windows. This may seem a luxury, but in Modula 2 you often need to look up a definition module to find the exact form of the procedure call. (This is particularly true for Turbo which has very strong typing.) You can copy code between the windows, which if you are a cut and paste programmer like me, is very useful.

The code for the editor, written in Modula 2, can be obtained, allowing you to customise the editor. You can already customise to a large extent, as many keys have macros attached to them, e.g. ESC T is THEN, and you can define your own macros. A minor niggle is that backspace is NON destructive and Del. deletes backwards instead of forwards. Though I have changed the Del. key macro definition so that it is the same as Turbo's, I couldn't change the backspace definition.

The compiler can be run from the editor, either in Fast check mode which produces no output, or compiling to relocatable native code. In both cases, when an error is found the editor is reinvoked at the error location, with reasonable explanations of the error. It is possible to force further compilation ignoring the error. This can not be done with the Turbo compiler, but it also invokes the editor, and when the error is fixed it starts recompiling from the beginning of the module in error, instead of the beginning of the source, a vast improvement over Turbo Pascal. Other than these features it is an unremarkable, very fast compiler.

The linker can also be made to run from the editor, but I have failed to do so. This is probably due to my poor understanding of the MSDOS operating system, in particular the PATH command. The linker produces .COM files. Again it is unremarkable in its behaviour, just well behaved. Rather than remove the last comment I have decided to point out it was my misunderstanding of the path command SET PATH= works, SET PATH = (which is the correct form in RTE) does not work. The linker can be invoked as well from the editor and the object program run, all with out leaving the editor.

A surprising part of this package is an 8088 assembler, using the standard small memory model (64K data, 64k code). It is included so that you can compile the assembly languages library modules, or any assembly languages modules that you write yourself. This assembler is really a tiny assembler and I would have been happier with the approach adopted by Borland; it would be so much nicer to be able to link in Microsoft macro assembler and Microsoft Fortran 77 modules.

The librarian is again LU like, not quite as friendly as the Turbo Librarian or LU.

Also provided is a debugger, but it would be more exact to describe it as a tracer rather than a debugger. If the trace option is switched on in the compiler and linker, the modules when executed will printout their progress on the screen.

In completion the package contains a sort module and maths module which contain such useful items, such as matrix inversion (Gauss method, with Partial pivot) - this is restricted to 10 x 10 matrices, but if you want to solve larger matrices, perhaps you should be using a mini anyway.

In conclusion, both of these packages are good value for money, providing a cheap entry into the world of Modula 2. I hope this review, in addition to showing the strengths of these packages, will also tempt you to try writing programs in this new language.

A Review of the NE889 Battery "M" Drive by Derek Beaty.

The Newburn NE889 is a 512K, battery-backed static RAM board, a new board from Newburn Electronics. It is designed to be upwards compatible with the GM833 RAM-disk board. An important difference is that this board is battery-backed.

All data files are retained when power is removed & returned. So when you switch on your 80-BUS system, all files are still intact & ready to run. Has anyone else ever switched off their computer before saving the "M" drive, or is it only me that does that sort of thing?

The board is designed to accept all future memory ICs using both 28 pin and 32 pin formats. Whenever the 611024-LP devices are priced down, the board will be easily upgraded to 2MBytes.

Being battery-backed, the system tracks may be put on the board. Why would you wish to do this when you need a floppy disk to put it there in the first place? Well, for one thing, speed. A system will boot from "M" drive just about immediately. Also, once the system track is on the board, you can take the (physical) disk drives away.

The usual SIMON boot EPROM will not enable a cold-boot without a (physical) disk in drive "A". Instead, Newburn will update your SIMON boot EPROM to enable this. The boot-up operation checks for a valid system track. If found, it boots from it. If not found, it goes on to boot from (physical) disk, as per usual.

My particular application was a small control system, where cost & dust prevented the use of the usual disk drives. The system had to auto-boot on power-on, or reset. The complete system consisted of a GM813 CPU/RAM, NE889 RAM-disk, GM816 I/O board, and GM817 PSU, in a MultiNet workstation case. Some additional circuitry was included for input and output buffering. A row of Klippons were mounted on the rear of the workstation case.

The software was written in Turbo Pascal, tested on a disk-based system, and loaded into the NE889 board. The system track was "RAMgenned" onto the NE889 board. This is a program supplied with the boot EPROM to copy a CP/M onto the board. The write-Protect switch was set, the board then removed from the development system and inserted into the "target" workstation.
All worked as expected, including some bugs of my own! The system booted from cold and ran immediately. In fact, you could not see the effects of pressing reset on the system, being so fast.

I forgot that one of the files on the disk needed to be changed at frequent intervals. Write-protecting the board did not allow that. Because the write-protect is also on the edge of the board, I was able to wire a keyswitch to it, so the operator could make changes, as required.

A useful thing would be to mix static RAM and EPROM on the same board. The problem is that the disk directory needs to be write-allowed. Newburn say they are working on something. If more than one board is used in a system, this problem does not arise. One board can contain EPROM, the other static memory. The Gemini BIOS only supports one "M" drive. Fortunately, it is possible to obtain a modified CP/M to support 4 "M" drives. The boards have 2 dil switches to select one of 4 drives. In this CP/M, the "M" drives are A,B,C & D. Using EPROM allows large programs to be stored quite cheaply. The NE889 is available from Newburn without any memory ICs. This enables EPROMs up to 1 MByte per socket (they are 32-Pin) giving a possible 16MBytes of EPROM disk.

Using the standard Gemini BIOS, up to 16, NE889s can be fitted in the system as the same drive "M". 4 Dil-switches on the board provide address selection. *[Ed. - actually, the Gemini BIOSs currently only (!) support up to 2MB of RAM-disk.]*

On a Network Workstation, the NE889 provides further benefits. Using the modified Boot, the workstation boots up directly to a "Silicon system" giving 512K of battery-backed files. This allows personal files to be kept off the Fileserver. If the Network is down for any reason, maybe a quick game of Colossus Adventure ?.

If the MultiNet Logon program is kept on "M", the Network can be booted at any time.

In conclusion, I have found the NE889 board very useful, both in a standard Galaxy, and in a control system. I can't wait until the larger static memory ICs become available.

Doctor Dark's Diary - Episode 27

Pluto upgrade saga continues...

Having done a remarkably successful bit of cadging, I now have a mini palette board for my Pluto 1. The board is small, and plugs onto the two lots of pins at the edge of the Pluto away from the mother-board. Once on, it does not come off easily, so if you bought one in the *Scorpio* sale, be sure you don't want to do any work underneath it before you fit it.

Also supplied were two ROMs to replace the original single one. These contain extra routines, to carry out the extra functions of the palette board. There must be a lot of spare space in the second one, presumably, unless the new routines are a lot bigger than they might reasonably be expected to be.

The catch when cadging equipment is that one is unlikely to get any support from the manufacturer if it refuses to do anything. Thanks again, Io, for making the two boards work when I gave up the attempt; what nice people you must all be. Mind you, they didn't install the wire link that needs to be put in to decode the second ROM, but I eventually worked that one out from some circuit diagrams of part of the board that I had also cadged.

I made an interesting discovery when I changed the plug on the video lead, to suit the mini palette connector. In fact I made exactly the same stupid mistake that I made the first time I connected the old lead up. If you ever get an all green picture, very bright, and way out of synchronisation, from a Microvitec monitor, you have done the same thing. The picture of the pin-out, next to the socket on the back of the monitor, is not a picture of the pins as you look at them. It is a picture of what you would see if you were inside the monitor, and looking out.

This is so absurd that I can scarcely believe it caught me out twice. Anyway, once I reversed the connections to a mirror image of my first version, the picture became far more pleasant to look at. The height and width of the picture were slightly different from before, but only the height needed to be adjusted, which is fortunate, as the width adjuster is not easy to get at.

Now we come to the interesting bit. The monitor is a standard Microvitec one. I had always used it for digital RGB signals, and had always thought that was all it could cope with. Knowledgeable people on Micronet told me how to change it over to analogue signals, and this is so easy and useful a thing to do that I am going to pass the information on.

But first, the warning. Switch it off, and leave it for half an hour, before you unplug it. Only then should you take the case off a monitor. They really can kill you, and as I have said before, I need all the readers I can get!

Inside the box, near where the wires from the input lead join the printed circuit board, is a row of ten vertical pins, with three links plugged onto them. The pins are in three groups of three, with a spare pin on the right as you look at it from the rear of the monitor. Assuming the pin on the left is pin 1, as you look at the monitor from the rear, then when the links connect pins 2 and 3, 4 and 5, 8 and 9, the monitor is set for TTL levels. If you change the links to connect 1 and 2, 5 and 6, 7 and 8, then you have just converted a monitor to analogue. Put the case back on before going any further.

Having done all that, I found I had what appeared to be a normal Pluto 1. I still had to test the new colours, as when powered up, the board defaults to the original saturated colours. Clearly, a quick and dirty test program was required. Choosing a suitable language for such purposes, I wrote a program to change randomly the colours on screen. I then ran an old graphics program to get something to look at, and then ran this:

```
1000 PSTAT = 160                    'PLUTO STATUS PORT
1010 PDATA = 161                    'PLUTO DATA PORT
1020 IF INP(PSTAT)<128 THEN 1020    'WAIT UNTIL PLUTO IS READY
1030 OUT PDATA,197                  'PLUTO WRITE TO LOOK UP TABLE COMMAND
1040 IF INP(PSTAT)<128 THEN 1040    'WAIT UNTIL PLUTO IS READY
1050 OUT PDATA,INT(RND(1)*8)        'WHICH COLOUR
1060 OUT PDATA,INT(RND(1)*128)+128  'GREEN LEVEL
1070 OUT PDATA,INT(RND(1)*128)+128  'BLUE LEVEL
1080 OUT PDATA,INT(RND(1)*128)+128  'RED LEVEL
1090 GOTO 1020
```

Those of you who are horrified by the appearance of a BASIC program in an article by me have not fully understood my "horses for courses" policy.

The program chooses one of the logical display colours at random, and then sets it to a random colour from the possible 4096 colours available with the mini palette. The result is very pleasing, if you like that sort of thing. With a good picture to work on, it makes most disco lights look rather ordinary. I hate discos, so this may be a biased judgement.

The program is not in its original form, as when I set the colour levels to INT(RND(1)*15), as the Pluto manual suggested, only very dim colours appeared. In fact, the board uses the numbers 0, 15, 31, 47 etc., rather than 0, 1, 2, 3... as stated in the manual. I hope this information comes in time for whoever bought mini palettes in the *Scorpio* sale. If the board only produces black and a very dim selection of colours, try the above change of parameters.

Other commands available on the mini palette include initialising the colour look up table to a grey scale, reading and writing colour look up tables, changing which table is in use (there are two) and a blink routine with variable speeds.

I actually dithered too long before trying to buy the full palette board in the great sale, which is an annoying thing to have happen when one has made up one's mind to spend. There was only one of those boards in the sale, and some lucky devil bought it and saved a fortune. I hope it is working, honest!

I have not managed to do an enormous amount of work with the improved board, as once again the Open University is keeping me very busy, but various programs in BASIC to vary the colours in more organised ways have convinced me that the Pluto is well worth the time spent on the upgrade.

What to do with graphics boards...

Well, obviously, you write programs to produce displays of the Mandelbrot set, as these are very pretty, and will be recognised by anyone who has seen them in magazines, on the television, and on the screen of the new pocket Cray.

Not many people know what the Mandelbrot set is, so I thought I would try to tell you. Experts, skip to a later section. The idea is simple enough. You choose a point (x,y) in what is known as the complex plane. That is, the x co-ordinate is a real number, and the y co-ordinate is a so-called "imaginary" number, in fact a multiple of the square root of -1.

Then, you carry out a transformation of the co-ordinates, as described by Benoit Mandelbrot, over and over until one of two things happens. Either the co-ordinates increase rapidly, and the point moves off to infinity, or it stays inside a distance of two units from the point (0,0). Actually, the point can do two and a bit things, and the bit is where the interest is. Some points take quite a while before they zoom off, while others soon go.

If a colour is chosen on the basis of how long the point stays, a pretty display results. This usually has its central area coloured black, representing points that are definitely in the Mandelbrot set, with a range of colours heading out towards the circle of radius two units that has been shown to surround all the points in the set. The length of the line round all points in the set, but excluding all other points, is infinite. Honest!

You would be disappointed if I wrote all that, and then didn't give you a program to do plots of the pictures, so here is one. It is written in Hisoft Pascal, version HP-5, which uses my Belectra HSA-88B board to speed up the calculations. This Pascal only likes upper case, which explains why the program is shouting at you.

```
PROGRAM MANDELBROT;
CONST
  {Pluto port addresses.}
  STATUS = 160; DATA = 161;
  {Pluto routine numbers.}
  MOVETO = 151; SCCOL = 137; RFILL = 129; LINERS = 150;
VAR
  SX, SY, C : INTEGER;
  U, V, J, K, Q, H, SIZE, HALFSIZE, X, Y : REAL;
```

```
PROCEDURE USERINPUT;
{This bit is dreadful. Instead of reading in co-ordinates from the keyboard, or
allowing you to zoom in on an existing display, it just selects one of the sets
of co-ordinates in the case statement. I ran out of time...}
VAR
  S : INTEGER;
BEGIN
  REPEAT
    WRITELN('Which picture? (1-20)');
    READLN(S);
  UNTIL S IN [1..20];
  CASE S OF
    1 : BEGIN  J := -2.0;      K := -2.0;      Q := 4.0      END;
    2 : BEGIN  J :=  0.3;      K := -0.2;      Q := 0.4      END;
    3 : BEGIN  J := -1.7;      K := -0.2;      Q := 0.4      END;
    4 : BEGIN  J := -1.8;      K := -0.05;     Q := 0.1      END;
    5 : BEGIN  J :=  0.25;     K :=  0.75;     Q := 0.5      END;
    6 : BEGIN  J := -1.3;      K := -0.65;     Q := 0.2      END;
    7 : BEGIN  J := -0.35;     K :=  0.815;    Q := 0.1      END;
    8 : BEGIN  J := -0.34;     K :=  0.820;    Q := 0.05     END;
    9 : BEGIN  J := -0.33;     K :=  0.8225;   Q := 0.025    END;
   10 : BEGIN  J := -0.322;    K :=  0.8350;   Q := 0.0125   END;
   11 : BEGIN  J := -0.322;    K :=  0.844375; Q := 0.003125 END;
   12 : BEGIN                                               END;
   13 : BEGIN  {Fill these in for yourself!}                END;
   14 : BEGIN                                               END;
   15 : BEGIN                                               END;
   16 : BEGIN                                               END;
   17 : BEGIN                                               END;
   18 : BEGIN                                               END;
   19 : BEGIN                                               END;
   20 : BEGIN                                               END
  END
END;
FUNCTION CALCULATE(SX,SY:INTEGER):INTEGER;
VAR
  COUNT : INTEGER;
  X, Y, X2,Y2 : REAL;
BEGIN
  {Work out the actual co-ordinates represented by the screen co-ordinates SX and
  SY. Note that HALFSIZE adjusts it to the centre of the rectangle.}
  U := J+H*(SX+HALFSIZE);
  V := K+H*(SY+HALFSIZE);
  COUNT := -1;
  X := 0;
  Y := 0;
  REPEAT
    {The squares of X and Y are needed twice, but only calculated once.}
    X2 := X*X;
    Y2 := Y*Y;
    {The actual translation is carried out next.}
    X := X2-Y2+U;
    Y := 2*X*Y+V;
    COUNT := COUNT+1;
  UNTIL (COUNT = 255) OR (X2+Y2 >=4);
  {The following selects a colour to be returned by the function according to
  the number of times the above loop could be carried out.}
  IF COUNT = 1
  THEN CALCULATE := 0                                {Black.}
  ELSE
    IF COUNT = 2
    THEN CALCULATE := 2                              {Blue.}
    ELSE
      IF COUNT = 3
      THEN CALCULATE := 3                            {Cyan.}
      ELSE
        IF COUNT = 4
        THEN CALCULATE := 1                          {Green.}
        ELSE
          IF (COUNT > 4) AND (COUNT < 11)
          THEN CALCULATE :=5                         {Yellow.}
          ELSE
            IF (COUNT > 10) AND (COUNT < 21)
            THEN CALCULATE :=6                       {Magenta.}
            ELSE
```

```
            IF (COUNT > 20) AND (COUNT < 255)
            THEN CALCULATE := 4                    {Red.}
            ELSE CALCULATE := 0                    {Black.}
END;
PROCEDURE PLUTOWAIT;
{Not surprisingly, this just waits until the Pluto board is ready.}
BEGIN
  WHILE INP(STATUS) < CHR(128) DO {NOTHING}
END;
PROCEDURE SENDWORDS(W1,W2 : INTEGER);
{This sends two sixteen bit words to Pluto, a byte at a time. It's a bit of a
mess because HP-5 uses 32 bit integers, and the OUT command wants characters.}
BEGIN
  OUT(DATA,PEEK(ADDR(W1)+2,CHAR));
  OUT(DATA,PEEK(ADDR(W1)+3,CHAR));
  OUT(DATA,PEEK(ADDR(W2)+2,CHAR));
  OUT(DATA,PEEK(ADDR(W2)+3,CHAR))
END;
PROCEDURE RECTANGLE(SX,SY:INTEGER;SIZE:REAL;C:INTEGER);
{This draws a rectangle, at point (SX,SY) with sides SIZE long, using the
colour C. The X side of the rectangle is doubled because I am using pairs of
pixels, due to the small size of Pluto pixels...}
BEGIN
  PLUTOWAIT; OUT(DATA,CHR(SCCOL));
  PLUTOWAIT; OUT(DATA,CHR(ORD(C)));
  PLUTOWAIT; OUT(DATA,CHR(MOVETO));
  PLUTOWAIT; SENDWORDS(SX*2+64,SY+16);
  PLUTOWAIT; OUT(DATA,CHR(RFILL));
  PLUTOWAIT; SENDWORDS(TRUNC(SIZE)*2,TRUNC(SIZE))
END;
PROCEDURE DOTS(SX,SY,C:INTEGER);
{This routine plots two dots, rather than using the rectangle routine when the
block size has reduced to one. It uses the relative plotting command.}
BEGIN
  PLUTOWAIT; OUT(DATA,CHR(SCCOL));
  PLUTOWAIT; OUT(DATA,CHR(ORD(C)));
  PLUTOWAIT; OUT(DATA,CHR(MOVETO));
  PLUTOWAIT; SENDWORDS(SX*2+64,SY+16);
  PLUTOWAIT; OUT(DATA,CHR(LINERS));
  PLUTOWAIT; OUT(DATA,CHR(1)); OUT(DATA,CHR(0))
END;
{Main program starts here.}
BEGIN
  USERINPUT;
  SIZE := 128; {Change to a smaller power of two to speed matters up!}
  HALFSIZE := SIZE/2;
  {Q is the width in real terms represented by the 256 pixel width of the
  display area, hence H is the width represented by a single pixel.}
  H := Q/256;
  REPEAT
    SX := 0;
    SY := 0;
    REPEAT
      REPEAT
        {Next statement is a bit of debugging, but I left it in so I could
        observe how the program was getting along.}
        WRITE('SX = ',SX:3,'  SY = ',SY:3,'.',CHR(13));
        {Print a white rectangle as a cursor so that one can see that the
        program is working, as black areas are rather slow.}
        IF SIZE > 1
        THEN
          RECTANGLE(SX,SY,SIZE,7);
        C := CALCULATE(SX,SY);
        {Print the actual rectangle, in the colour returned by the calculate
        function, or a pair of pixels if size is down to 1.}
        IF SIZE > 1
        THEN
          RECTANGLE(SX,SY,SIZE,C)
        ELSE
          DOTS(SX,SY,C);
        SX := SX+TRUNC(SIZE); {Move to next horizontal position.}
      UNTIL SX >= 256;
      SX := 0;
      SY := SY+TRUNC(SIZE); {Move down to next row.}
    UNTIL SY >= 256;
```

```
    SIZE := SIZE/2;
    {Another debugging statement left in for interest's sake.}
    WRITELN('Size now = ',TRUNC(SIZE):3,'     ');
    HALFSIZE := SIZE/2
  UNTIL SIZE = 0.5
END.
```

A major advantage of this program is the way it successively refines the picture, starting with large blocks of colour, and reducing the size of the block. When you are exploring to find interesting areas, this is much better than waiting ages to see what appears on the screen.

The program is rather primitive in the way it fails completely to allow the user to choose an area while it is running. Instead, you need to insert co-ordinates in the user input routine, and recompile. As they say in the text books, improvement of this part of the program is left as an exercise for the reader...

Even using the Belectra board, the program takes a long time. And as the size of block being drawn is reduced, the time for a scan of the screen actually quadruples. As a consequence, I have only once let a picture be completed, as yet. Before I carry out a long run like that again, I intend to write a couple of programs to load and save pictures. Using a simple form of data compression, a picture will fit into 32K, with a quarter of this space being redundant. I could make the compression more efficient, but this would be at the expense of ease of writing the program.

I would have suggested that this program would make a useful benchmark of processor performance, but having seen a Meiko Computing Surface produce a screen of Mandelbrot in about one second, I would have found this computer's couple of hours too depressing.

Another Super Sale.

I am told that the *Scorpio* sale was a great success, with the non-working items being particularly popular. This suggests to me that there must still be a lot of people around who do enjoy making their own machine, and making it work. Well, I have some bits of equipment for sale too, some of them will need to go to enthusiasts. Or masochists, perhaps? I can be contacted on 0823-276768 if you want to buy any of this gear, and I might haggle a bit over some of the items.

- **Spectrum 48K**, issue 2, with **VTX5000** modem and some games, including "Doomdark's Revenge" which is really difficult. This one is £150 including postage and packing. The keyboard is ideal for the rubber fetishist, of course.
- **Iotec Iona**. An astounding machine in a really high quality case, good keyboard. This has a Z80, 64K of RAM, and a colour display. There are several internal expansion slots, and a printer controller board and disc drive controller board are part of the bargain. List price was around £900, but as the manuals are missing, you could collect it for £250. It is a bit heavy to post...
- An **S100 box**, with two single sided, single density 8" disk drives, PSU with colossal transformer, a Z80 processor board, no less than three memory boards, so it probably has a huge 48K of RAM, and an Input Output board. There is no display or keyboard, but anyone capable of using this would know how to fasten a terminal to it, via its 25 pin socket. Needs collecting as it is very heavy, and is a snip at £200.
- An **Ohio Challenger**, which was a very popular machine once upon a time. I think this has BASIC built in, and it has been expanded from 4K to 8K. Wow! Quite a nice machine for the enquiring junior programmer, and the makers of the Superboard copied it almost exactly. Just about postable, for £100.
- **Dolphin BD80P printer** fitted with Centronics interface. Seven needle head, so hardly NLQ, but probably fine for listings. Again, too heavy to post so it is going to have to be collected, unless you want to pay extra for one of the courier type services! I feel £50 is such a bargain I can't negotiate here!
- Brand new **8" drive by Control Data**, single sided, single density. £50?
- Dreadful wreck of a **Nascom 1**, half its chips missing, once the property of a famous technical writer. Comes with badly hammered Nascom 3A PSU remains, and a reasonably undamaged keyboard. This should be useful as spares if anyone has a Nascom 1 they want to keep going. I'll post it for ten measly quid.

Quite a list of bargains, I think you will agree. How any of you will be able to resist this amazing series of offers, I cannot imagine. You may suspect that I am trying to raise money for something, and if you do, you are right. More of this later, if time and the deadline permit...

Light Pen for the SVC.

I have wanted one of these for quite some time, and eventually saw an article describing one in the Maplin magazine. The article describes a design for which a kit is available from Maplin, naturally. Although the unit is not supposed to be for use with SVC's, I thought it just might work, and the price was fairly reasonable.

It's an easy kit to build, with only a single IC used to shape the pulse from the light sensitive transistor. Unfortunately, at present all I get from it is random numbers. I suspect the pen circuit is producing a negative pulse, and the documentation for the SVC doesn't say what kind of pulse it is expecting. Positive, probably.

The other possibility that occurred to me was that perhaps the pen was picking up too much stray light, or was not properly shielded from the electrical noise Marvin puts out. I would be interested to hear from anyone with experience of using a light pen with the SVC, and will naturally pass on anything that works properly!

Expanding Nascoms?

Bert Martin wrote to me, expanding on his suggestions for an amateur priced extension board for 80-BUS systems, among other things. (The other things included a suggestion that I should stop moaning about my boring job, and do something positive to get out of the rut. He has a point, but I enjoy a good moan, and can't promise not to do more of it, even if I escape the rut.)

He has obviously put quite a lot of thought into ways of making elderly Nascoms and 80-BUS machines in general able to use IBM type discs (well, can't the MFB series do this already? It must be possible.) *[Ed. - systems running Gemini BIOS 3.2 or later can read/write IBM PC-DOS disks with the IBM-COPY program, available separately.]* and made some interesting suggestions about how we could use their software by emulating the hardware. It wouldn't go fast enough. But never mind.

Marvin has now reached a stage of expansion where there are only three slots left on the mother board, and I am not sure I want to add more processing power to the brute. For a start, there are already two boards contending for a set of port addresses - the modem and the Belectra board. The bus is heavily used already, with the Pluto and SVC in situ.

I have considered various processors as possible candidates for an expansion board, ranging from exotic bit slice things from Texas, to the 32032 from National Semiconductor. The latter is too complex. The obvious choice of processor, if one wants real power, is the Transputer, or rather, lots of them. At the moment, they cost a lot. But nothing falls like the price of chips, and some time soon, they should reach a sensible level. I have been looking at OU courses which teach Occam, so I will be ready to program them. But rather than hang them onto Marvin, and make him even more complex, I have decided to halt extensions to the system. Not that I am going to get rid of Marvin; too many useful functions to list are now second nature on this machine. But I am getting a new machine. It will be used in conjunction with Marvin, once I work out an interface. And it will not be an IBM type machine. After all these years of criticising machines with plastic cases, and making snide remarks about WIMPs, I am going to have both.

The machine in question is expandable, although not as easily as an 80-BUS computer is, and it is multi-tasking, and is an ideal candidate for use with Transputers, as a "front end". I'm getting an Amiga A500. And before you say it isn't IBM compatible, and is therefore a side-track in evolution, I should point out that British Aerospace are using A1000s as workstations because they are so much better than IBMs. Somebody demonstrated one to me, and I was hooked. The mouse is good, and the graphics are (gulp) better than the Pluto. It has a poor BASIC, but Modula 2 is easily available, and worth having.

I hope that I will become good enough with the machine to write about it, but will probably send any stuff I produce about it to Amiga based magazines. I feel it would be out of place here, apart from details of how I connect the two machines, perhaps, if I manage to do it.

It is going to feel very strange, after all these years, having a machine the same as other people's. It will be a luxury, too, being able to buy programs instead of having to write them myself. But give up the Nascom? Never!

The MAP-80 VFC, MPI and 256K RAM Cards by Michael Newson

When faced with the choice of what 80-BUS disk controller or video card to buy, there are several choices on the market. The final choice must depend not only on the depth of pocket, but on the eventual planned requirements of the system. So, what are these for a disk-based system ?

If we want to run CP/M, then we need some means of 'booting' from the system disk. This can either take the form of a program loaded from tape running under some other monitor, or as is more usual, to load the system boot sector direct from disk using an EPROM routine running upon RESET. This boot sector in turn pages out the EPROM, loads the main system, and eventually we get the CP/M prompt.

The most common Boot EPROMs are Gemini's SIMON and RP/M, and the Map-80 VSOFT. The Gemini versions sits on the CPU card. RP/M is unsuitable for the Nascom-2. It can, however be used on its native card with no problem. There is a version of SIMON for the Nascom. Map-80's VSOFT is the controlling software for the Map-80 VFC. Whichever boot system is chosen reflects on the choice of supplier of CP/M as they include checks on the boot sector to determine that it is in fact a system disk being read. It can also affect your choice of RAM card(s), as some CBIOSs will support some types and not others.

MAP-80 VFC - Video / Floppy Controller

This card consists of a 25x80 video display, a parallel keyboard port, a floppy disk controller and the controlling software (VSOFT) EPROM. The card is available either ready built, or in kit form. The advantage of the kit approach is that you only need to pay for the options you require. (VSOFT is part of the basic card - you can choose any or all of the Video, Video Switch, Floppy or Keyboard options.) It will work with the Nascom-2, GM811, GM813, and Map-CPU cards.

How does it compare ? The standard card uses 16 ports, addressed E0 - EF but alternative ports C0 - CF are available on request. (This is necessary if using the GM809 FDC or Map-80 MPI.)

The video section is controlled by a 6845, and all character handling is done by the main system CPU running routines in VSOFT, unlike the Gemini IVC and SVC. It should therefore be slightly slower, as the IVC and SVC contain their own Z80s. Because the system CPU runs VSOFT, both VSOFT and the video RAM can be individually paged into the memory map under port control when required. (The overlaid system RAM is available at all other times.) The address at which VSOFT executes is also flexible - it can reside at a software controlled 4K boundary, and told where it is in an initialisation call. In addition, the auto-boot facility is link and software selectable. The card does not have a programmable character set, but does have a socket for a 2716/2732 second character generator. This socket is under software control - you can select character generator 1 or 2, and normal 00H-FFH, or normal 00H-7FH + inverse video of 00H-7FH for 80H-FFH with the selected set.

The final (optional) part of the video circuitry is a video switch which selects the output to be either the VFC screen, or pass through an external video signal - e.g. normal output from the Nascom-2 under software control. This means that although the VFC cannot generate the old Nascom format screen, (which would be at the wrong address anyway), the original Nascom screen can still be displayed when running NAS-SYS (as modified to MONITOR.COM to work with disks or loaded from disk and executed for standard NAS-SYS operation). This theoretically allows you to use your old software provided it doesn't attempt to change the non-existent EPROM or corrupt VFC work areas. The parallel keyboard input consists of a strobed latch which is read via VSOFT. The strobe polarity is link selectable.

The final section is the Floppy Disk Controller. This uses the WD2797, and is GM809 compatible. As with the GM809, however, the clock frequency of the FDC is link selectable, meaning that selection between 5.25" and 8" drives can only be done by link changing. If you want software selection, go for the Map-MPI card or Gemini GM829 or GM849 instead.

Control codes for the Map VFC are similar to the Gemini IVC, but since there is no onboard CPU, there is no facility to load or run user programs on the card. Almost all screen and character handling commands are identical, but differences creep in concerning character set selection, format and graphics. The VFC has a programmable (any) key string table, whereas the IVC can only reprogram its function keys. The VFC does not support a light pen. (If any of these are of concern, you should naturally obtain and compare the relevant manuals.)

MAP-80 MPI - Multi Purpose Interface

This card is primarily a disk interface card, capable of talking to 3", 5.25" and 8" floppy disk drives. A SASI bus interface also allows connection to winchester drives, via a hard disk controller card. As well as these, it provides communications via an on-board CTC and SIO with RS-232 and RS-485 (high speed multi-drop) interfaces. The board is addressed as 16 ports, starting at 00H, 20H, 40H ... etc. (link selectable: E0H is recommended with VSOFT).

The floppy disk controller section uses the WD2797 as standard, configured to make it compatible with the GM809, GM829 and GM849. Optionally, the WD2793 can be fitted instead and a few links changed to make the MPI operate as the Lucas-Nascom FDC. SASI, 5.25" and 8" drive cables may all be attached simultaneously. The drive size (clock) is software selectable, and with suitable software switching the maximum stepping rate of 6mS for 5.25" drives can be reduced to 3mS by selecting 8" mode while stepping.

The CTC may be clocked either from the system clock, or from two (user supplied) crystals. Output from the CTC can be link selected to clock the SIO for communications: 1 x RS-232 input, 1 x RS-232 output, and 1 x RS-485 bi-directional high-speed multi-drop (2 x data, 2 x clock lines).

MAP-80 256K RAM

While the Nascom-2 on-board RAM can be used in conjunction with another RAM card e.g. Gemini GM802, the system really takes off when more RAM is available for use as a virtual disk as it allows programs to run at RAM rather than disk-based speeds.

The 256K RAM card pages memory into the Z80 memory map, unlike the Gemini RAM-Disks, which are accessed on a port basis. When used as virtual disk there doesn't seem to be much to choose between them, BUT if you plan to run Map-80's implementation of CP/M PLUS, then you will need the banking facility offered by the 256K RAM boards. This is because parts of the OS. are held in different banks, and the TPA is actually in bank 1. Don't bother with Non-banked CP/M Plus on a 64K system, it doesn't have the full range of facilities - use CP/M 2.2 instead. Both 2.2 and Plus are available, configured, from Map-80. If you already have CP/M, they will customise yours to suit.)

The size of page paged depends upon the processor card used. With the Nascom-2 and GM811 the page size is 32K. With the GM813, any 4K block can be mapped into any 4K slot in the Z80 memory map (A19 may need to be connected to the bus - see "The Great A19 Debate" 80-BUS News Vol.2 Iss.1 p43). Selection of the mode and card addressing is performed via two link blocks. When these cards are supplying all the memory in a system, there is a facility to force Page 0 into place (link and software selectable) so that there is a common area of memory in place for program execution. The maximum paged RAM supported as standard is 1Mbyte, but there is no reason why Gemini Ram-disks (GM833 & GM873, which are port accessed) should not exist in a system at the same time - if you're rich enough. Even if you only have a GM802 64K RAM card, modification details are given to use it in conjunction with the Map 256K RAM. The minimum memory which can be fitted to the card is 64K, and a further 3 banks can be fitted as funds permit to make up the full 256K.

The GM862 256K Dynamic RAM card is limited to a single card with the GM811 CPU card (and probably Nascom-2 too), although up to 2Mbyte (8 cards) can be added to a GM813 based system. Its mapping arrangements are different to those employed by the Map 256K RAM; paging 56K, 60K or 64K pages with 8K, 4K or 0K of common memory when in page mode; or usable in 4K mappable blocks when used with the GM813.

All in all, a very powerful and flexible system can be assembled as described above, but never lose sight of what you aim to achieve with your system. It can only ever be as good as its software, and you must either ensure that software support is available for the products you wish to incorporate, or be prepared to do it yourself. (Map-80 are able to supply modifications for SIMON and RP/M to allow them to work with their various products.)

On that point, I close by stating that although I have no links with Map-80 (other than Adam!), they have always been most helpful with any questions or problems. The provision of circuits and sources is also most welcome as it is always better to have them and never need them, than the reverse. I do not mean to slight Gemini in any way for their products are first-rate, it's just that while I developed my system over the years, I made my choices with an eye to future flexibility based on what was available as well as cost.

WordStar V3.30 for the Gemini IVC/SVC by D.J. Birch

This article details the changes necessary to support the keyboard cursor keys for WordStar version 3.30. All other functions i.e. clear to end of line/screen etc. can be patched through the install program (called WINSTALL). See article in 80-BUS NEWS Vol. 2 issue 3 page 11 for Installation of WordStar V2.2 on IVC by Richard Beal.

New addresses are necessary for Ver. 3.30 as the patch area has been changed. Below are detailed the changes for support of the cursor keys and also a few other locations that users may find useful to patch. Patching can be done either with DDT/ZDT etc. or better via the patcher in WINSTALL (accessed via a "+"at the main install menu).

Address (label)	New Value	Meaning
04A3	43 68	Backspace delete char not just cursor left.
0545	7F	Disable Control-Delete (1F).
0655	1C 00 DA 63	Cursor left.
0659	1D 00 D0 63	Cursor right.
065D	1F 00 9A 64	Cursor down.
0661	1E 00 B4 64	Cursor up.

For dot matrix printers:

069A (BLDSTR) - number passes for bold strike (normally 3)
069B (DBLSTR) - number passes for double strike (normally 2)
03E3 (OVPCHR) - overprint character (normally "-")
0710 (ULCHR) - underline character used on printer (normally "_")

Useful addresses:

03DF (EOFCHR) - End of file character used on screen (normally ".")
03E1 (HARDCR) - Hard carriage return character used on screen (normally "<")
03E4 (PAGCHR) - Page break character displayed on screen (normally "P")
03E8 (PAGFIL) - Page break line character displayed on screen (normally "-")
07BA (PSTAB) - Proportional/micro-justification character spacing look-up table
03E7 (SOFHYC) - Soft hyphen character displayed on screen (normally inverse "-")
03E5 (SOFTCR) - Soft carriage return displayed on screen (normally " ")
02BA (SCRLSZ) - Scroll size of screen (normally 14h = 20 lines)
03DE (CONCHR) - Continued line on screen display character (normally "+")
03E0 (FDTCHR) - Character used to flag Mailmerge dot commands (normally "M")
03E2 (LFCHR) - Character used to flag line feed only in text (normally "J")

For people with an EV Real Time Clock and Gemini IVC card the following bytes in the Terminal Startup and Exit routines automatically deselect clock, unlock top line, clear screen and clear screen, select clock, line feed, lock top line of display respectively.

Terminal Startup - 1B 73 43 1B 4F 1A
Terminal Exit - 1A 1B 73 43 0A 1B 4D

I found this very useful as the clock display if selected under WordStar does get in the way.

Software Reviews by Robert Pearce

Following my letter to the Editor (see the letters' pages), I have been asked to scrawl a few words regarding a couple of software packages.

Review of Scorpio Systems SVC-03 Disk

I thought I should start with this one as it's already available, and also it was this disk that got me into this in the first place. In all honesty I have to say that the main reason I bought it was that I was considering writing a PCB design package of my own and had not got started yet.

With the last issue of *Scorpio News*, Scorpio offered three disks with software specifically for systems fitted with Gemini GM832 SVC video boards. There are in fact really only two disks, the third one (SVC-03) being the first two disks combined onto one disk - follow that?

SVC-01 contains:

SDRAW, a drawing package
TIME, for setting/displaying the GM888 board's RTC
PTIME, for displaying the GM816 or GM822 boards' RTCs
NCG, a new character set for the SVC
PCB, a simple PCB design package

SVC-02 contains:

MLINKS, the source code that adds SVC graphic commands to MBASIC

I shall start with SDRAW, the picture drawing package, which is the only one of the "5 plus 1" programs which does not have the source code supplied. Overall I was quite impressed by this program. All the usual facilities are provided in easy to use form with a very flash online help system. Cursor movement is designed for an anonymous keyboard (i.e. it can be used without cursor keys, using the "Wordstar diamond") but the standard Gemini cursor codes also work in most (but not all) cases. One of the cases where they do not work is in text mode. This uses the standard SVC character writing routine with the resulting restriction that text cannot be placed exactly where you want it, only on a 32 * 25 grid (the usual one). Another slight niggle is the way text mode always enters at top left rather than where the cursor is.

Facilities are provided for printer dump to an Epson compatible printer in two sizes. The big size, unfortunately, does not quite fit so the left and right edges (8 pixels each) are cut off. The print page also allows you to print out the current cursor co-ordinates, though why you should want to I don't quite know. Screens can also be saved to or read from disk files, which are compatible with those used by MLINKS.

The circle drawing routine understands Pythagoras, so the direction in which you define the radius is not important. There is, however, no way to remove an unwanted circle, which is perhaps even more annoying because such a facility is provided for the fill option. The fill option supports 12 patterns. Solid fill is rather quicker than the rest (presumably it uses the SVC internal routine) but all perform a proper flood fill. The help page shows four extra undefined fill patterns but these appear to be a mistake on the help file.

All in all this is quite a good product and the niggles are generally minor.

I could not test the two time programs as I don't have a real time clock fitted, but there can't be much wrong with them!

The new character set program is also pretty straight forward. The character set it gives is fatter than the default and is of a style which seems quite popular, particularly in high resolution character generators. I personally think it looks disgusting on an SVC but that is just personal taste. Anyway, the characters are easily re-defined if you want to.

The PCB program is rather limited, but apparently it is only a "taster" for a better version (see later). If you are designing a single sided PCB less than 3 inches square and don't mind copying it off the screen by hand (or writing your own dump routine), and also are of a patient nature then it is very good. I have found two bugs in it. One relates to track delete and is mentioned in my letter. The other bug is in track move, where if, having chosen and marked your track, you move the end over the end of another track then the second track is picked up and moved as well. This is not so easily fixed, but armed with the knowledge it can be avoided. I have used my re-written version of this program (which supports extra facilities) to design an interface board for my Galaxy and have found that it does speed up the process considerably. I had to take a dozen intermediate hard-copies (this was mostly because I was doing a 5" x 4" double sided board, but not entirely) so I feel that the lack of a hard-copy facility is probably the most serious drawback.

The other program supplied on SVC-03 is the source for the GM575 SVC-graphpac. Doctor Dark has covered this briefly in the last issue so I shall not cover it again here. What I shall say is that I found a bug in it within half an hour. I mentioned this in my letter in which I also mentioned how to correct it. For my troubles I received a copy of GM575 to check. The bug, for I still feel it to be such, is indeed present here, but all the demonstration programs cater for it by using the MBasic RESET instruction immediately before any use of GSI or PSI. I don't know whether the manual instructs this but the modification I gave removes the need for it.

As regards the claim that this "could be used as the basis for adding your own commands", I feel this is a little hopeful as the source is sparsely commented and written in a very confusing manner, but what do you expect ?

All in all a good package, well worth the £20 (this includes P&P and VAT).

A Better PCB Package

And one which does most of what mine does! Actually it does some things mine doesn't (yet).

The version I have is a "pre-release" (nth revision prototype) version, so I shall attempt to be "honest but not too destructive" as my instructions say. It has been written by the same person as the simple version in SVC-01 and -03, and he will be selling it directly for £20 + VAT.

I noticed immediately that the bugs I mentioned have been fixed. This version caters for boards up to 8.6" x 8.6", handles double sided boards and includes a dump routine for Epson compatibles (the width is restricted to 8" for this). Also added are 3 new types of pad and rotated versions of the two asymmetric types. The IC package facility is extended to include 64 pin packs, but for some obscure reason it does not support 20 pin ones.

Well those are the good points.....

I'm afraid I have to say there are several things I would want done differently. These relate mostly to the methods used to implement the improvements.

Firstly, the extended size capability. Because the SVC display is too small it is obviously necessary to have only a section of the board displayed at any one time, at least while work is being done. The system used divides the board into 9 segments roughly 3 inches square. At any one time there will be one of these on the screen and the others not visible. Unfortunately the boundaries are fixed and there is no overlap. The result is that there is a "noughts and crosses board" of almost un-usable bits which are impossible to see clearly. Anyone who has tried navigating across the boundary between non-overlapping maps without being able to see both at once will know how tricky it is. The solution would be to allow any section (or at least one of 16 or more overlapping sections) to be displayed.

To make this worse, the boundary zones cannot be crossed by tracks so you have to find the meeting point and define two tracks. This process is painful, and on the demo. PCB file it has obviously caused some confusion at one point.

The double sided facility is also bound to cause problems with the two colour (on / off) display. This has been done by displaying one side or the other but not both. This makes sense but without the facility to display both sides simultaneously (at least temporarily) it is still quite difficult to use.

These three problems are the most major ones and (sorry to boast but..) they are ones which I had already dealt with in my own version. My complaint regarding the printer dump is also one for which I have devised a solution but I have not yet written the code. When dumping a line to the printer it is necessary to draw three screens on the SVC to get the relevant width. Each screen, however, contains parts of about 30 lines. This program draws all three screens for each line thus plotting each one thirty times. Consequently it is very very very slow. If the data for all thirty lines were "cached" in memory it would be very much faster.

Another (and perhaps more major) criticism is that no effort is made to ignore totally blank areas. The result of this on a small-ish board, like the demo. file, is that the printer spends several minutes moving the print head from one side to the other and doing nothing with it, even to the extent that I began to wonder whether the machine had crashed. Surely if the largest Y co-ordinate in the file is 260 there is no need to start dumping from 800 down!

A rather obscure change is that pads now have a side code. This is necessary for double sided edge connector pads and nothing else. However, for some strange reason it refers to all pads with the result that holes do not apparently go through the board.

In order to fit the extra program size it now consists of several chained files for the various bits. I have had to do this too, but I think the way they are split is rather silly here and it results in inconvenient command structures and rather slow operation.

Well those are my moans. As I commented regarding the simple version, it helps to be patient with this one, but the basic concept is good, and at this low price you pay your money and make your choice.

The author of the PCB software replies

I would like to comment on the so called 'Review' of the SVC PCB program written by myself. I wouldn't have minded so much reading this 'review' if the person who had written it had done the review properly, instead he waded straight in to the complaints and failed to mention fully the features of the program and its ease of use in relation to its price (see the advert elsewhere in this issue). Instead he just compared it the the small pcb program issued on the *Scorpio Systems* Disks, which by no means compares anything at all to its big brother.

I do not accept his argument about the track routines not being able to move outside the currently displayed PCB segment since I did write a version which did allow this feature. Unfortunately this process, where all the tracks had to be checked for clipping against the edge boundary of the screen before they could be displayed, dramatically slowed down the display when it was being updated, and after the initial testing of the program it was decided to use the simpler track handling routines to keep the speed of the program up.

Other mistakes in his 'review' included the inability of the program to plot 20 pin ic layouts, the case being it has always been able to do so, the only area where this is mentioned is in the on-line help file where a syntax error has occurred. He also makes no mention about the numerous types of pads and symbols which are available (12 in total).

Also I cannot make any sense why he doesn't think that a side flag is required for the pcb symbols - surely if a person requires a through hole on a double sided PCB they would put the appropriate symbol on both sides at the same co-ordinates. This also leads me onto his idea of displaying both sides of the PCB at the same time, if this was done on a fairly packed pcb all this would lead to would be unrecognisable green patch on the screen, instead why not just use the 'flip side' key to turn the PCB over for a brief look at the other side.

The only one area with which I could agree with in the 'review' concerned that of the printer routine being very slow. This has been cured with each of the PCB segments now being displayed only once and stored in a temporary file which is dumped to the printer as a whole to produce a 1:1 scale pcb. The printer routine however still retains the facility to print out either individual segments quickly in order to review the PCB through its design stage, or to print either to the SVC or printer a complete compressed view of the entire PCB quickly.

The last point concern his idea that it was 'silly' to split the program up into so many sub-programs which to him seems to slow every thing down. If he has written any programs using a compiler (in this case Turbo Pascal) then it should be apparent that the user has a limited space in which to write his program, i.e. in Turbo this tends to be around 800 statement lines maximum including any allocated variable space, any one program segment, and in this case all the most commonly used commands are grouped together in the main program segment, with the rest of the satellite programs being centred around this central core.

The SVC PCB program has been tested by a few other Gemini enthusiasts (including a university technician and other people who might use the program in their work) and has gathered a fairly favourable response.

Regarding the bugs in the small version of the pcb program, I will pass a new rectified listing and .COM file to the Editor shortly.

Obtaining 00H to FFH from the Nascom Keyboard by Michael Newson

The full range of byte values from 00H to FFH is available direct from the Nascom keyboard - a fact which only becomes apparent through detailed study of the NAS-SYS listings, although the actual keys required are not specified. The table below gives a quick cross-reference to the required key combinations.

The MAP-80 CBIOS for CP/M also supports this full range, and it is probable that it also applies to Gemini's CBIOS. (Note also that if 'values' (e.g. <CNTL> L or <CNTL> Z) are typed as a filename in response to the CP/M prompt and it cannot be found as a filename, CP/M will echo the 'filename' unaltered to e.g. a printer if <CNTL>P is active.)

Characters not shown in the table below are available direct from the keyboard, using the shift key where relevant.

HEX	NAME	KEYS	HEX	NAME	KEYS	HEX	NAME	KEYS
00	NUL	cs @	0D	CR	c M	1A	SUB	c Z
01	SOH	c A	0E	SO	c N	1B	ESC	<ESC> = c [
02	STX	c B	0F	SI	c O	1C	FS	cs [
03	ETX	c C	10	DLE	c P	1D	GS	c]
04	EOT	c D	11	DC1(CUL)	(<-) = c Q	1E	RS	cs 0
05	ENQ	c E	12	DC2(CUR)	(->) = c R	1F	US	cs]
06	ACK	c F	13	DC3(CUU)	(↑) = c S	====typeable====		
07	BEL	c G	14	DC4(CUD)	(↓) = c T	60	`	cs (SP)
08	BS	c H	15	NAK	c U = s (<-)	7B	{	c ;
09	HT	c I	16	SYN	c V = s (->)	7C	|	cs ,
0A	LF	c J	17	ETB	(CH) = c W	7D	}	cs =
0B	VT	c K	18	CAN	c X	7E	~	cs .
0C	FF	c L	19	EM	c Y	7F	DEL	cs ?

In the KEYS column, 'c' represents the <CTRL> key, and 's' represents the <SHIFT> key.

Values in the range 80H to FFH may be obtained by pressing the <GRAPH> key as well. (It sets bit 7, and depending on the character generator, may display either reversed video or a graphic character.)

The Dave Hunt Pages - PCs & MS-DOS

The Prologue

Well this is the October article, and would you believe the date today is the 1st of June? Heck, this disciplined writing bit will be the death of me. Not, of course that I'll finish today, no, whatever I write today will end up on the *Scorpio* archive disk and stay there for the next two or three months, then the editor will be hounding me for it, and I still won't have finished.

As mentioned in the last article (I've just re-read it, despite the fact that you won't see it for the best part of a month yet) I note that this *might* be the last magazine. Pity! I have less time for writing these days, and that writing I do, I prefer to go were it will be most helpful (assuming that anything I write could be construed as helpful), that's why I rather like writing for low circulation 'private' magazines, this one in particular. I certainly don't do it for the machismo, I'll never get famous trotting out this rubbish, and it certainly isn't for the money, hint!!

It could be something about the fact that they'll take anything I care to write, "Never mind the quality, feel the width", without too much blue pencil work. I've never been asked to revise what I've written and only on a few occasions have I had anything rejected (never by this magazine). So the possible demise of this magazine means that I could be deprived of an ever copy-hungry publisher and you will be deprived of the few pearls of wisdom I might accidentally throw out every now and then between the 6000 odd words of my usual articles.

So don't let it die folks. At over four pounds a copy, it's not cheap, but so long as it publishes enough to keep you happy and for you to feel you're getting your money's worth, spread the word; don't rip it off or share copies, get people to buy their own; that way, there may be just enough subs to keep it alive.

Anyway, enough of that, come the copy deadline things may have changed so the above paragraph, and this, may yet have to be deleted.

New Toys ?

There has been a lot pontification in the computer press lately about what Big Blue has been up to. All their new machines and things. A lot of fuss about nothing if you ask me.

"IBM have scrapped the PC type computers"	-- *Mixed boo's and cheers*
"They've invented a new computer"	-- *Hooray*
"With an incompatible BUS"	-- *Boo*
"Which goes faster"	-- *Hooray*
"It hasn't got 5¼" 360K drives"	-- *Boo*
"It's got 3½" 1.44 Mbyte drives instead"	-- *Hooray*
"It doesn't have to have a winnie"	-- *Eh??*
"It can use an optical disk instead"	-- *What? you're kidding!!*
"They bunged a new DOS on it"	-- *Oh no!*
"But it's not available yet"	-- *Thank Gord for that*
"It got a new BUS controller to stop people cloning it"	-- *That figures*

and so on.

So the new machines are faster, bigger and cleverer, with 3½" drives. That's good. The new micro-channel BUS will only accept IBM approved add-on bits. That's bad. The new operating system OS/2 isn't even available yet. But what does that all mean? Particularly to someone like me who has just forked out a considerable sum (my money, not anyone else's) for an IBM AT clone. Well reading between the lines, and also following some of the remarks by software houses, other computer manufacturers, etc.; it's not all bad news.

Firstly OS/2 is upwards compatible from MS-DOS, so that most of the old stuff should work. OS/2 isn't IBM property, and Microsoft will be marketing it themselves with a particular eye on the AT and AT clone upgrade market. Many of the software houses aren't that happy, they see that concentrating effort on the new machines could split their markets and many have stated that they will continue to write for MS-DOS, with consideration that OS/2 is upward compatible. That means

that many of the cleverer aspects of OS/2 won't get implemented for some considerable time. Of course, the software houses will change their minds when there are enough of the new IBM machines around to make it worth their while.

Remember, one of the over-riding reasons for the 'open' architecture of the original IBM machines was to encourage writers to adopt MS-DOS as the then 'new' standard. A very successful ploy it was to. This time the new IBM machines use a 'closed' architecture, so the incentive for change does not exist to the degree it existed in 1982. It's a nice 'Catch 22' situation which many computer manufacturers' have found themselves in before. Software houses won't write the software for a new machine as there aren't enough of the new machines to make it worth it; on the other hand, customers won't buy the new machine as there is insufficient software about to make the machine viable. In this instance, the situation isn't as clear cut, as the old software standards are applicable, but the incentive is missing.

Next the micro-channel BUS architecture is a clever idea, but the user base of older machines will ensure that add-ons and bits will remain available, certainly during the projected life of my machine, and the benefits of speed brought about be the new BUS don't seem to be that great. I haven't seen any comparative timings yet, but the implication is that a 10 or 12 MHz clone is equally as fast as all but the fastest of IBM's new toys, and anyway, speed isn't everything.

What about the changes in disk format and the inclusion of optical disk. So with a bit of ingenuity existing machines will soon be able to handle the new 3½" format, fit a new box of drives, a new controller card and a DRIVER.SYS patched into DOS and away it goes (older 3½" formats will have to do the same, the Apricot XENi, the RML Nimbus, etc., were based on the older IBM 720K format and the drives aren't up to it). And as for optical disk, what's new in that? I've been playing with 5¼" WORMs for some months now and mine are 400 MByte per side not 100 Mbyte per side as proposed (but not yet available) from IBM. Interest will perk up when it comes to erasable read/write optical disks, but these are only just becoming available and it will be some time before they are a viable proposition for 'everyday' use.

Whilst passing optical disks, can I get on one of my pet hobby horses. People keep referring to '5¼"' optical disks. This is historical, it's also wrong! 5¼" floppy disks are 5.25" square, and if you take the magnetic disk out of a floppy disk, you'll find it close to 5.125" in diameter, as close to 5.25" as makes no difference. But unlike floppy disks, where a floppy disk is a floppy disk regardless of capacity, people seem to equate the diameter of an optical disk directly with its capacity.

To that end, then, people should be more precise about the actual size of the disks themselves. A '5¼"' optical disk IS NOT 5.25" in diameter, it's actually 125mm, that's as near as damn it 4.75". A 12" optical disk is 12" in diameter and the funny Kodak ones are actually 14" in diameter. (I haven't laid hands on one of the 8" optical disks yet to actually measure it.) When people refer to a '5¼"' optical disk, what they actually mean is that the drive will fit in the hole left by a 5¼" floppy disk drive (which is a shade under 6" anyway)! Oh well I suppose I can keep banging my head against a brick wall, it's so nice and peaceful when I stop!!

Something Completely Different ?

And now, for something completely different. (Yes, I had noticed that they've been re-running Monty Python on the box.) The upgrade from dBASE II to dBASE III and dBASE III Plus. (From now on I'm going to call them DB, where I mean any version of dBASE, and DBII, DBIII and DBIII+ to be specific.)

As you are no doubt painfully aware, I tend to concentrate my programming expertise into database management using the Ashton-Tate DB products. Now why those in particular? I don't think I've ever justified my thinking on those lines, so let's have a look at that first.

There are database managers and Database Managers, in fact an awful lot of them these days, and over the years I've had a look at a good few. I've had time enough and seen enough to postulate **'Dave's Law of Database Management'**, that is: *'Mickey Mouse database managers offer Mickey Mouse facilities, whilst big heavy OTT database managers require a PhD to run them.'*. Now I like Mickey Mouse, he's funny little cartoon character and in many ways quite bright, but I think we can be fairly sure of one fact, he doesn't know a lot about database management.

On the other hand, I'm not yer super brainy type and I didn't go to university during my misspent youth, as I don't think the OU has got round to handing out PhDs to honestly deserving but mentally lazy people like me yet, at least, not without me having to do some work for it, I'm not in the running from that source either. The only university I went to was the 'University of Life' with an assortment of day release courses and sandwich courses, which admittedly took longer to achieve a given goal (it was 5 or 6 years before I got my HNC), none the less, allowed me to earn my own keep and learn what happens in real world rather than what is supposed to happen in the not so real world which can exist in some universities. Anyway, the whole point is that some of the real big database managers are next to impossible for the average mortal to drive.

The Ashton-Tate product errs towards the PhD end of the scale, it's got a fair amount of reasonable facilities but is a bit of a pig to get used to. It has one feature which WAS unique (no longer so), it has its own programming language. The language was originally included to allow you to do the nastier and more complex bits in a sequence of little steps, so super mental capability wasn't necessary to achieve a given aim.

Of course with a language at your disposal, it meant that the clever guy could write programs which would ease the way for lesser mortals to achieve the same ends, and in that respect DB was/is very good. That doesn't mean that DB is perfect, it's got some glaring holes in its philosophy (a decent method of 'hashing' the database doesn't exist and is the most obvious omission). I'm still looking for a better database manager which will offer the same as DB but with more of what I need, but I haven't found one that competes overall yet, but I'm still looking.

So four or five years ago I taught myself DB, that was DBII version 2.0 at the time, and over a period of time DB grew, the ultimate version for CP/M being version 2.43. Ashton-Tate ported Version 2.43 to the IBM with virtually no changes and so it was usable with the same interpreted programs on either machine, the only obvious difference being that the filetype extension had been changed from .CMD to .PRG; annoying!

Then came DBIII, an original version written for 16 bit processors and MS-DOS. This had changed a lot, some changes for the better, some for the worse and some for no apparent reason at all. I don't intend to concentrate on that version, but rather on DBIII+, an even bigger version which has been around for about a year now. Most of what I will say applies to DBIII equally. DBIII+ is just bigger still (I didn't say better, it's noticeably slower!).

The differences between DBII and DBIII+, well not a lot really, some syntactical changes and the change of the odd command word here and there. Perhaps the most useful change was the way assignments were done, in DBII, the extremely messy:

```
STORE xx TO var
```

was supplemented by the more usual and logical:

```
var = xx
```

whilst still retaining the 'STORE' form for compatibility (and to get round an interesting anomaly) and for masochists who like to type more than is absolutely necessary. The anomaly? Oh yes, you can't assign a macro to a macro using the '=' form, so

```
&var1 = &var2
```

just won't work, but you can do it this way:

```
STORE &var2 TO &var1
```

don't ask me why this is, the manual doesn't mention it! Still on the subject of assignments, you can't make multiple assignments using the '=' but you can with the STORE. Another useful one is that it is now possible to assign a 'null' string, previously you couldn't. Let's do both:

```
var1 = var2 = .... var<n> = ""
```

won't work, you can either do it:

```
var1 = ""
var2 = ""
  .
  .
var<n> = ""
```

or you can do it simply and quicker:

```
STORE "" TO var1,var2,.....,var<n>
```

All round a big improvement for the better, and it makes programs shorter.

What else has changed? Well an oddity is the ERASE command, that word was perhaps not the best choice for clearing the screen, but more convenient than the current word, CLEAR. Let me explain, the word CLEAR in DBIII and DBIII + has several uses, CLEAR clears the screen, CLEAR MEMORY clears all the memory variables (but not the screen), CLEAR ALL clears everything in sight including closing the databases (but doesn't clear the screen) and CLEAR GETS chucks all pending inputs away (but still doesn't clear the screen). So why CLEAR to clear the screen? Incidentally, the word ERASE which no longer appears in the DB list of commands, used in the following fashion:

```
ERASE filename.typ
```

will, none the less, erase a file without checking if you meant it!

Another little change is the use of the underscore character instead of the colon as a separator in variable names so that a variable,

```
mstr:list
```

would be written as

```
mstr_list
```

Not that this made much difference, in fact, it appears more natural, for as far as I know, DBII was the only programming language which uses a colon as a separator, whereas many use an underscore.

Perhaps the biggest pain (for me at least) was the much improved method specifying logical data types. In DBII, four different characters can be used to specify true or false; they are 't' and 'y' for true and 'f' or 'n' for false. So the assignment:

```
STORE y TO var
```

would be interpreted by DBII as either a logical assignment or the assignment of a variable, y, depending upon whether y existed before the assignment, and what type of assignment it was previously. If in doubt, DBII would assume it to be logical, and would always interpret it so if it hit a conditional statement which used it, and the syntax of the conditional was such that it appear *as a logical assignment in the first place.*

Very messy, but once you realised what was happening you could live with it. DBIII draws a definite distinction between other data-types and the logical data-type but placing a dot before and after the assignment variable to denote its use as a logical. Sound thinking that, it's exactly the same construct they use to indicate logical operators in a command statement. Anyway, the logical assignment above becomes:

```
var = .y.
```

I lived with DBII method of doing things until I came to convert a load of stuff from DBII to DBIII using the supplied utility DCONVERT.EXE, which, amongst other things, whips through the source file converting all ERASEs to CLEARs, all colons in variable names to underscores and all occurrences of 'y', 'n', 't' and 'f' to logical assignments regardless of context, etc. The problem was, the source software did a lot of positioning of a device not unlike a graphics plotter (actually moving a piece of microfilm to predetermined positions looked up in the database), and I had used the variables 'x' and 'y' to indicate points within the plot. All my 'y' positions turned into logical assignments, and I spent hours looking through the software for machine introduced errors.

Perhaps I'm being unfair to the DCONVERT program, because it was my fault in using 'y' in the first place, an invitation for confusion. In places where DCONVERT had assumed a logical variable, it had marked it in the source, and any other doubtful bits as well, although it would sometimes miss colons when they were neither in drive assignments or text. None the less, I converted acres of files this way, and the majority worked straight away. The only pity was that DCONVERT didn't convert the STORE type assignments to '=' type assignments, I ended up doing that by hand afterwards.

There are many other differences but they are mainly insignificant in nature, the extra power of DBIII and DBIII+ comes from the new facilities, these are many and overcome a lot of the deficiencies of DBII. Take for instance the lack of fast subroutine calls (procedures) in DBII. To use a procedure, you wrote a separate little program to do your procedure bit, then told DBII to 'DO' it.

The problem was, that as the program was interpreted, and to keep the memory utilisation low, DBII read the program from disk as it went along, it had to drop the current part of the program, open the new procedure file, do it, close the file and then carry on with what it was doing before the procedure turned up. All very disk intensive and slow particularly if the procedure was inside a loop of some kind.

In fact from an operating speed point of view, DBII actively discouraged the use of procedures. DBIII and DBIII+ introduced a new idea. Again to keep the memory utilisation under control, the program is read from disk as before, but (a maximum of 32) procedures can now be grouped together in one file and opened before the program starts. As this file is opened before the program starts, and not closed afterwards (unless you have a number of 'procedures' files and therefore open another) the whole process becomes a lot less disk intensive.

Another area where DBII was deficient was the fact that only two databases could be opened at the same time. With DBIII and DBIII+ up to ten databases may be opened at once, but care as to the total number of open files (program files, databases and indexes) must be exercised. With up to 10 databases open, sorting out which one you're actually using at any one time can be fun, so a system of aliasing has been devised. You can either refer to a database by its name, or by its alias, which may be assigned when the file is opened. There are some rather clever relational linkages between databases which are allied to this ability of having a number of databases open at once (I haven't had cause to use them yet, so I can't tell you if they work).

Another major change is the increased number of variables (256) and the use of local variables within procedures, a bit C or Pascal-ish that, but it can catch the unwary. Unless a variable is declared PUBLIC at the start, or declared before it is passed to the procedure, or passed using the

```
DO <procedure> WITH <vars list>
```

command, followed by a matching PARAMETERS command in the procedure, strange things happen, and variables which would have been passed under DBII don't get passed under DBIII. On the whole this is change for the better but takes a little getting used to, particularly if you are converting programs and unaware of what's going on.

The maximum number of fields, the field lengths and numbers of records have all grown and are now large enough for even the most ambitious space gobbler.

As DBIII and DBIII+ are up-market products, all sorts of nice goodies have been bolted on, the SET COLOR commands are useful from a sales point of view, punters always like to see Technicolour screens even if they are constrained to black and white. Some nice box drawing functions have been added to DBIII+ and with a bit of cleverness 'windows' become possible. Calls to machine code routines, missing on DBII (MS-DOS version) and DBIII have been put back, albeit in a different form, and of course, DBIII can 'Run' other programs from inside itself (see last article).

DO WHILE loops have gained an EXIT command, so that a DO WHILE .t. can be exited with ease from inside a CASE statement (possible, but messy using DBII). On the subject of loops, a FOR .. NEXT loop has appeared.

A couple of new data-types have appeared (apart from formalising the logical data-type), a 'date' data-type, and the memo field in the database. The 'date' data-type automatically does the Julian date conversions from a date and back again, whilst the memo field saves database space where the occasional large comment may be required. The memo field links with a second database and

allows multiple 'pages' of up to 512 bytes at a time to be 'attached' to the main database without the horrendous space requirement which would be added to a database if the memo fields were included inside it.

ZAP is a nice one whilst testing and mucking about, ZAP simply empties a database! (Not to be written in to programs!!)

The debugging is much improved, the program may be 'SUSPENDed' and the debug mode entered. Watch out for the difference between CANCEL and SUSPEND, SUSPEND does just that and can be RESUMEd, CANCEL will certainly stop a program but it will also lose all the variables which weren't declared PUBLIC at the beginning. Having SUSPENDed, depending on what you turned on, it is then possible to single step through a program whilst reading the current command statement and observing its results. How I wish some other interpreters (notably BASIC) would do that.

That really sums up the changes, reading the manuals helps particularly the bits about converting from DBII, as DBIII gives a short form list of the syntax changes and the new commands.

The problem is that the short form list doesn't tell you what the new commands do, you have to look then up in the manual, well part 1 of the two manuals actually. The real problem with the manuals is that they've now got so thick (nearly 9" of A5 pages, excluding the binders) that I think Ashton Tate have made a conscious move towards ecology and protection of trees and all that, by economising on paper. The major sections of the books are separated as one would expect, but when it comes to a blow by blow description of the commands and functions, they've been bolted end to end rather than a separate page (or pages) for each. Worse, the command and function names aren't listed at the top of each page which would make the manual usable, but are printed in pale green at the start of the relevant paragraph with no indentation or anything. It makes it very frustrating to find anything.

I can't think of any good reason why it should be done this way, except perhaps that the pale green is the sort of colour that photo copiers don't like. But then, anyone who wants to copy eight or nine hundred double sided sheets to rip of the manual must be awarded full marks for masochism.

The saving grace is that the 'HELP' is fairly good, and may be called up from within the program by asking:

```
HELP XXXX
```

So all you need is an idea of what it is that you want help about (and the short form list provides that) and you can ask. It usually gives a brief description, the syntax and comments on any related quirks; it also lists allied commands for possible examination as well.

DBIII+ is usable and reasonably fast, not as fast as DBIII and again not as fast as DBII (the apparent speed increase between DBII and DBIII+ is more to do with the increased speed of the computer rather than any tightening up of the program). I guess this figures, as DBIII+ must have an horrendous command parser to unscramble some of the command lines it is now possible to write and that must take some time to do its stuff. I've been using one particular dBASE compiler for the fast year and very sweet it is.

DB Compilers

There are three DB compilers as far as I'm aware, Clipper, Foxbase and Quicksilver. Clipper's the one I've been using and the one I know most about.

Foxbase (I'm told) produces code which runs faster than Clipper and is 'more compatible' with DBIII+; but has two snags. During a recent trip to the States I had the advantage of speaking to someone who has been using both Clipper and Foxbase, and allowed me the opportunity to play around a bit. It's immediately apparent that Foxbase produces code of amazing size, I thought Clipper's knack of producing a 120K program just to say *"Hello"* was a bit much, but compared with Foxbase, Clipper is very economical with space! Typical code with Foxbase is around twice the size of code produced by Clipper, I've no idea why, both seem to pull in the runtime modules in total when compiling, it just happens that Foxbase's runtimes are bigger than Clipper's.

The second snag is licensing policy. With Clipper, once you've paid the loot for it, it's yours, and any programs produced for resale don't have any kick back to Nantucket (Clipper's publisher). Foxbase, on the other hand demand a percentage of each program sold using their

runtimes. Not a problem for an individual writing programs for himself, but when it comes to commercial use of either, Clipper is to be preferred as the end user doesn't have to pay anything for the licensing of the compiler's runtimes. Mind you, neither of these compilers is what you might call cheap, Clipper comes in at about £600, whilst Foxbase doesn't give you much change out of £400, so you aren't likely to buy either for fun.

My Stateside contact had recently confined Foxbase to the bin, as the disadvantages outweighed the advantages.

I know very little about Quicksilver. Paul is pressing me for copy (I said I wouldn't get round to finishing this article before the copy deadline) and I shan't lay hands on Quicksilver until next week. From what I read in the adverts, there are two things going for it. Firstly, not only are the database structures compatible with DBIII+, but the indexes are compatible as well (with Clipper and Foxbase, the databases are compatible, but the indexes aren't); secondly, it has I/O commands, the equivalent of BASIC's INP and OUT instructions. Not a major plus perhaps, except I have need of I/O commands in something I'm doing, and to be honest, I still find assembler patches in 8086 code a grade one pain (all this segment register lark). I haven't found out the bad bits yet, like the size and speed of the code, or compatibility or licensing arrangements, etc.

So that leaves us with Clipper. As I said, I've become intimately involved with Clipper over the last year. What can I say, it does the job - it compiles DBIII+ source with very little incompatibility. Ok, so it's not entirely compatible, I have to keep in mind that what I'm writing is going to end up compiled so that some of the little quirks are accounted for.

Some of the incompatibility of my version of Clipper appears to have come about by Nantucket being a little previous. It seems that my version was written around DBIII and then the designers thought, 'Wouldn't it be nice to include this, that, and the other.'. So they went ahead and added some nice bits, including windowing, box drawing and the ability to detect what particular key was used to exit from a GET statement. All worthwhile enhancements of DBIII.

Unfortunately, Ashton Tate also thought these would be a good idea and included them in DBIII+, but, you guessed it, they work slightly differently! The real killer of that bunch is the ability to return the key press on exit from a GET, in Clipper the function is called LASTKEY() whilst in DBIII+ it's called READKEY() and it returns a number. Worse, the same key returns a different number for either dBIII+ or Clipper, with more keys effective in DBIII+; this has resulted in me writing a translation table to sort it all out, messy and takes up space, but it does allow all sorts of things which would otherwise be impossible.

On the score of returning key presses, another incompatibility is the way INKEY() returns the function keys between DBIII+ and Clipper. Now any ordinary key is Ok, the value returned by INKEY() is the character value of the key pressed, but function keys work differently, within DOS, a minimum two byte code is returned for function keys, the first byte being the key number, indicating it's a function key, and the second and subsequent bytes being the function key string.

DBIII+ knows this and reads the second byte correctly returning the first character of what ever string has been placed in the function key buffer, but only through function keys f1 through f10 - it ignores the other possible permutations of Alt-<function key>, Shift-<function key> and Ctrl-<function key>. Clipper gets it wrong, it returns the key number only, but for all possible permutations. Good news (more keys to play with) and bad news (you can't preprogram them and have to sort them out later).

All this sounds like bad news from the programming point of view, but there are several ways round it, the neatest is that Nantucket anticipated this might happen so they have included the 'Clipper' variable. Now when a PUBLIC variable has been declared, it takes the datatype logical until used, not only that, it is always declared false, both by DBIII+ and Clipper. So Clipper looks at the PUBLIC declarations when compiling, and if it comes across a PUBLIC call 'Clipper', it automatically declares it true, so that compiler differences can be written round. Take the box drawing functions. In this example we declare the publics first, then write some of the main program and then we want to draw a box. As we will be using the box drawing procedure for drawing all types of boxes throughout the program, we must pass the x, y co-ordinates to the procedure, so I'll use the DO .. WITH <parameters>command:

```
* Declare the PUBLICS
PUBLIC clipper, frame

* Declare the constants
frame=CHR(218)+CHR(196)+CHR(191)+CHR(179)+CHR(217)+CHR(196)+;
      CHR(192)+CHR(179)

* Now some program
   .
   .
x1=20
x2=70
y1=10
y2=20
DO drw_box WITH x1,y1,x2,y2
   .
   .

* This procedure is in the 'Procedures' program
PROC drw_box
  PARAMETERS x1,y1,x2,y2
  IF clipper
    @ y1,x1,y2,x2 BOX frame
  ELSE
    @ y1,x1 TO y2,x2
  ENDIF
RETURN
```

This little routine will draw the predetermined box using the Clipper BOX command if compiled or the DBIII+ @ y1,x1 TO y2,x2 command if not. The 'frame' variable allows the characters to be used to construct the box to be predetermined with Clipper, in DBIII+ you're stuck with single line box drawing. As that little lot compiles, Clipper chucks out the @ y1,x1 TO y2,x2 line as an error, but as it never executes the code it doesn't matter.

Another little point from the above. the DO .. WITH and the matching PARAMETERS in the procedure are strictly speaking superfluous in this case as both DBIII+ and Clipper will pass parameters from a higher level routine to a lower, and as parameters are not passed back to the higher routine, it was not necessary to explicitly state them. I've only included them for completeness.

There are, of course dozens of little differences between DBIII+ and Clipper; almost all are insignificant but need to be kept in mind when programming. Clipper I find is almost ideal (I haven't got round to Quicksilver yet). I've ordered the up grade to bring my Clipper more up to date and, perhaps to remove the odd bug I've found (I've found two, and one is most frustrating) and to improve compatibility with DBIII+, but to be honest, I find that there is very little which can not be achieved with Clipper which can't be persuaded to work with a little imagination.

I suppose the majority of my programming effort is conducted in a combination of DBIII+ and Clipper these days and I find both comfortable and easy to work with.

For Sale

Gemini GM888, hardly used, complete in original packing, including documentation and CP/M-86. Also GM808 EPROM Programmer in neat box. 'Phone John Parrott on 0794-301411.

MAP VFC absolutely complete with ALL data, 1 year old, £175. GM812 IVC card, £50. GM802 64K + page mode, £50. 2 x Nascom 48K RAM B + page mode, £40. GM654 5-slot motherboard, all connectors, £22. Gemini 75W switch mode PSU, £32. Gemini 140W switch mode PSU, £42. Gemini CP/M 2.2 for Nascom 2 inc. Boot ROMs, etc., £80. Seagate ST506 5 MB Winchester, £55. Miniscribe 10 MB Winchester, £95. IBM Golfball + CP/M interface, £35. Olivetti TTY Punch/Reader, FREE (save it from the dump!). All items in good working order, any offers, Ian on Ipswich (0473) 831353.

A Quick Look at GEM by P.A. Greenhalgh

I have been "playing" with a selection of GEM software for the last few weeks, and had hoped to present a detailed review of it here. However, the information on the various new 80-BUS CPU boards arrived, and I felt that the space should be given over to them. Consequently what follows is just a brief over-view and taster, and I hope that I shall write a much more detailed report on GEM and various GEM applications in the near future.

GEM comes from Digital Research, and stands for Graphics Environment Manager. Basically, although I doubt that DR would officially agree with this, its intention is to bring MAC type facilities to PC compatibles, i.e. it provides a WIMPs environment (Windows, Icons, Mouse, Pull-down menus).

Installing GEM is a doddle. It supports a wide range of hardware, and you just answer the various options that it gives you. It can be run on floppy or Winchester based systems. Video can be CGA, EGA (in various options), or Hercules. A large assortment of printers are supported, as are plotters. Different mice are supported, and also graphics tablets.

With GEM installed you can go out and get a variety of packages. The ones I have are:

GEM Draw Plus GEM Graph GEM Paint GEM WordChart GEM Write

When you get the packages home, you just insert the master disk in a drive, run a program on that disk, and after a bit of whirring it's all done. So what, you say? Well, the clever bit really is the graphics screen and printer support. You see, when you add a new GEM application to your system you don't need to tell it anything about your hardware (compare that to putting new WP software on a Gemini IVC/SVC!). The software interface between the application package and your hardware is GEM itself. I suppose an analogy can be made with CP/M, as follows: CP/M handles all disk accesses for any application program, and the program is not interested in whether it is using floppy disk, hard disk, RAM-disk or whatever, and isn't interested in the exact capacity available; so with GEM applications, they don't care what the graphics resolution is, whether it's monochrome or colour, or what the printer or plotter resolution is - these aspects are handled by GEM.

The main portion of GEM is called GEM Desktop. GEM Desktop effectively becomes your Operating System prompt. From GEM Desktop you can basically do anything that you can from the MS-DOS prompt, the chief difference being that you are presented with files as pictures (icons) and you "point" at what you want to use by using the cursor, which is moved around the screen as you move the mouse.

GEM CAN be driven from the keyboard, but a mouse is a much better proposition. When I first received the software I didn't have a mouse, and so had no option but to drive everything from the keyboard. This can be a laborious process, as during the learning phase you inevitably tend to make mistakes, or want to try each and every option. In the process you keep finding yourself at totally the wrong corner of the screen, and it seems to take for-ever to move around with the cursor keys. Not recommended!

So, I got myself a mouse (a rather flash, cordless one, actually), and then very quickly became extremely impressed with the software.

Unfortunately I'm running out of space, and so really there is just one message I would like to get across now. That is that a few people have said to me that they aren't interested in GEM because they don't want to run any of the GEM applications. Well, you don't have to and it can still make life a lot easier for you. For example, you want to copy various files from one sub-directory to another. Simple, you can have both directories on the screen at the same time. You use the mouse to point to each of the files that you want to copy (they turn from white to black as you select them). When you point to the last one you keep your finger on the mouse button and move the cursor to point into the second directory. Release the mouse button and all the files are copied over! Want to erase some files? Again select the ones that you want, point to the trash can(!), and away they go.

GEM Desktop really can make a system much easier to use (even fun ?), and it may even make you confident enough to let your wife and kids onto the system, or is that what you want to avoid!

730K Disks in an AT by D.R. Hunt & J. Parrott

Dave Hunt Begins

My bit in the last issue about making AT High Density drives format 730K normal double density disks has caused quite a lot of fun and burrowing into the undocumented realms of IBMs, clones, MS-DOS and PC-DOS. Incidentally, the last article referred to 720K disks, no, the disks haven't grown an extra 10K in the space of three months, just that I got it wrong. I had done those experiments some time before I wrote it up and at the time I simply said two times 360 is 720 and there you are! The actual size is 731K, so from now on, they'll be referred to as 730K disks in sympathy with the MS-DOS habit of rounding down to the nearest 10K.

Firstly, the DRIVPARM statement only applies to MS-DOS 3.2 and later, Microsoft have provided a for various disk parameters and the syntax is similar to that given for DRIVPARM in the last issue. Examples of the use of DRIVER.SYS are given in John's letter below. Secondly, what follows only applies to AT type computers fitted with IBM AT type disk controllers and one or more High Density drives. The 730K disk trick will work with most other IBMs and clones which are NOT AT types.

Now we (the collective is John and myself, with phone calls to Paul and Brian) think we've got to the bottom of it and we're fairly sure who's to blame. It all comes down to the IBM spec. for the controller card for the IBM AT, and clone manufacturers being in the business of cloning Big Blue, clone away, warts and all. Put simply, the IBM controller card is physically incapable of formatting a disk with 80 tracks and in normal density.

For starters, I suppose I'd better explain a bit about High Density drives and disks. The drive itself looks and feels, and to all intents and purposes IS, a fairly normal 5¼" drive. The only real difference is pin 2 on the drive connector which for a normal drive is marked 'Reserved'. Actually, on a normal drive it isn't usually connected to anything, it just sits there. The HD drive uses this pin to switch it into HD mode. Put a logical 1 on the pin and two things happen; firstly, the drive increases rotational speed from 300 r.p.m. to 360 r.p.m., secondly, the electronics switch into a mode suitable for use at double the normal data rate, 500K bit/s instead of the usual 250K bit/s. In other words, it now behaves like an 8" drive, only smaller.

The problem is the data packing density on the disk, related to the disk speed. Obviously, the angular velocity of an 8" disk is a lot higher (the tracks are near the outer edge of the disk as is the case for 5¼" disks) than the smaller disk. When it comes to 5¼" disks, the normal disk can't cope, the packing density is too high to make it reliable. Two approaches are made to overcome this, the first is to increase the angular velocity by jacking up the speed from 300 r.p.m. to 360 r.p.m., the second is to use disks coated with a magnetic pudding of higher coercivity, we call these HD disks. Higher coercivity magnetic coatings means lower head currents which don't saturate the disks to the level that is used on normal disks, so it means that HD disks format unreliably (if at all) when used in place of 'normal' disks.

So back to the IBM controller card. Why IBM? Because that's the one I've got the circuit diagrams for. Now, apart from a real genuine Big Blue AT at work, the rest of us use an assortment of clones with an amazing variety of controller cards, and despite the different chippery on the controller cards, they share two things in common. They were intended for AT clone computers and they are all capable of running two floppy disks (not more) and two hard disks.

Now there are three possible permutations covered by the combinations of drives we could fit.

80 track HD drive should be able to do:
- a) 1.2M HD disks (80 tracks 15 sectors)
- b) 730K Normal density (80 tracks 9 sectors)
- c) 360K Normal density (40 tracks 9 sectors) double step

80 track 'Normal' drive should be able to do:
- b) 730K Normal density (80 tracks 9 sectors)
- c) 360K Normal density (40 tracks 9 sectors) double step

40 track 'Normal' drive should be able to do:
- c) 360K Normal density (40 tracks 9 sectors)

(There are other perms. as well, all should be capable of the older 8 sector format in both double and single sided, but as far as I'm concerned, these don't count.)

The IBM card (and I guess the others) has some simple and stupid dedicated hardware features. The major ghoulie is two input latches, one which says "80 track/40 track" which relates physically to the drive selected and the other which says "High/Normal density". This could give us four permutations which should cover the three cases we want and one we don't.

Unfortunately, when "80 track" is selected, it jams the other latch so that it's automatically high density so any drive assigned with 80 tracks is automatically assumed to be high density. Also, the head step hardware always sends two head-step pulses to the drive but in the case of a "High density" being selected a hardware 'divide 2' divider is inserted in the head-step line to ensure one step per track. If an "80 track" drive is selected and "Normal density" is asserted, then the divider is switched out to provide 40 tracks on the 80 track drive, and the data rate switches to 250K bits/s. Thus formatting a 360K disk. This explains my finding of a normal 80 track drive double stepping, despite being an 80 track drive and running normal density.

If "40 track" is selected, then it jams the "Density" latch to low and forces normal 360K operation.

That isn't the end of the story, I've since discovered a controller card which claims to solve the problem, although I haven't tried it (I'm not spending more money on something I'm prepared to live with, although I'd rather not - live with it, that is). The card is the 'UDC card' from UPIC of Dunstable.

The story is continued by John and what follows is an edited version of a long letter which I received in the middle of July. It appears to get round the problem with one particular controller (Everest) and throws some beginners light on to some of the other peculiar aspects of MS-DOS.

John Parrott Continues

There are many IBM XTs and ATs at the company where I used to work and the word was that 360k disks written on the high density drives were notoriously unreliable. Of course that made some sense to me, knowing that the difference in the head areas does cause a reduction in the signal to noise ratio when a 96tpi recording is played back on a 48tpi drive, because of the unrecorded areas (or even worse, unerased 96tpi tracks) "seen" by the 48tpi heads. But having said that, I have to say that with the Gemini one could get away with almost anything most of the time. Anyway, the computer services people were therefore equipping their ATs with 360K 48tpi drives so that disks could be shared around the company.

The time came for me to buy an AT (why, how and which are another story) and the addition of a 360KB drive was an essential item on my shopping list. I was also looking forward to playing with one of those high-density drives - should be a doddle to make them do any format - I thought. I bought a Japanese clone manufactured by Precise from CAS Computers. They fitted the extra drive while I waited and then demonstrated the machine.

I asked for some disks to be formatted and some programs run. All the programs ran and the Norton SI gave a system performance figure of 13.5 (although it typically gives 11.5 now that my system is fully configured). The formatting and disk copying were, however, proving very unreliable even when using the /4 option and a duff 360K drive was suspected, even though I had seen it removed from it's brand-new packaging. Before changing the drive, we tried formatting in the HD drive. This was also unreliable. Some "experts" were called and they said "ah ha, you're using quad density disks and they won't work at 48tpi". I explained that these were fresh unformatted disks and that my old fashioned CP/M system had no problem in formatting them in any format chosen. I tried to explain that the HD drive was 96tpi anyway etc., etc., but to no avail. They thought it was my disks, I thought there was evidence of a bug in MS-DOS relating to the HD disk and I wasn't sure about the 360k drive.

I finally persuaded them to try another 360k drive and that initially gave the same results as before. This pointed me towards there being an MS-DOS bug in respect of all floppy drives. I then quickly discovered that if I read a particular type of disk, I could successfully format and write disks of the same format. DISKCOPY remained a 50/50 affair (as it had been with the company XTs and ATs) but that didn't matter. I was satisfied that all the machine's hardware was working and the "experts" were happy that it had all been because of my disks. I left, clutching my new toy.

I had never touched MS-DOS until I picked up my AT, so when I got my machine home, I set about reading the MS-DOS 3.2 manual and discovered that it was, in my opinion, just a rip-off of CP/M 2.2 but with the addition of tree directories, directory hashing and more support for program co-residency. It's presentation and friendliness are poor and slapdash (e.g. you type DIR to have a list disappear off the top of the screen, or DIR/W to get an unsorted directory without total space occupied etc.). It is more reminiscent of CP/M 1.1 in it's friendliness. And this is version 3.2, God knows what the early versions were like! CP/M 2.2 patched with CCPZ leaves it standing and ZCPR is even better.

My first task then was to get the 360K drive configured for reliable operation and then the 730k format on the HD drive would be an easy next task. I constructed a CONFIG.SYS file with EDLIN (which though basic, is quite adequate and suitable for the creation of batch files). I worked out that for a system with 2 floppies and 1 winnie, the first memory device in the list would be allocated to logical device D:, the next to E:, etc. I used the statement:

```
DEVICE=DRIVER.SYS /D:1 /T:40 /F:0
```

rebooted, did FORMAT D: and the disk formatted perfectly. I must admit that I did mess around with the /S, /H and /C parameters before I arrived at the above, and I have to say that the results were peculiar. The only logic I could make of it was that MS-DOS didn't like being told something it didn't need to be told, even though it was true, or possibly there was an undocumented limit on the number of parameters that could be passed. Anyway why worry I thought, after all it works. So, full of confidence that I had mastered MS-DOS and shown the "experts" to be wrong, I set about an entry in CONFIG.SYS for the the HD drive and settled on:

```
DEVICE=DRIVER.SYS /D:0 /F:2
```

I had by this time come to some conclusions about the way MS-DOS handles disk formats: It does an "intelligent" analysis of the drives present at system initialisation, it sometimes does an "intelligent" analysis of the disk, it gets both analyses wrong from time to time and the two imperfections taken together produce low reliability and high inscrutability.

I therefore decided that I would by-pass any "auto density searching" by having distinct entries in my CONFIG.SYS file for each density I would be using and not depend on the MS-DOS logical devices A: and B:. My CONFIG.SYS file now looked something like this:-

```
BREAK=ON
BUFFERS=10
COUNTRY=044
DEVICE=ANSI.SYS
DEVICE=DRIVER.SYS /D:0 /S:15 /C /F:1          (D:HD 1.2 MB)
DEVICE=DRIVER.SYS /D:0 /S:9 /C /F:2           (E:730KB)
DEVICE=DRIVER.SYS /D:0 /T:40 /C /F:0          (F:360KB)
DEVICE=DRIVER.SYS /D:1 /T:40 /F:0             (G:360KB)
DEVICE=DRIVER.SYS /D:2 /C /F:2                (H:external 3½")
DEVICE=DRIVER.SYS /D:3 /T:77 /S:26 /N:1 /C /F:3 (I:8"SSSD)
DEVICE=DRIVER.SYS /D:3 /T:77 /S:26 /N:2 /C /F:3 (J:8"DSSD)
DEVICE=DRIVER.SYS /D:3 /T:77 /S:26 /N:2 /C /F:4 (K:8"DSDD)
DEVICE=DRIVER.SYS /D:129 /C /N /F:6           (L:tape drive)
DEVICE=VDISK.SYS 384 /E                       (M:ram disk)
LASTDRIVE=M                                   (M for Memory, neat)
```

I was getting pretty confident, including 3½" and 8" drives still to be connected, a tape drive which was a figment of my imagination and a ramdisk using the space between 640K and 1024K which MS-DOS seems unable to handle as ordinary memory. I was now able to format 360K, 730K and 1.2M discs without problems, or so I thought.

It was about this time that I read an article in Scorpio News by one DRH. Could it be that I knew something he did not? The answer is no or not much, for after our sequence of 'phone conversations it became evident that the 730K disks, although verifying as 731016 bytes were in fact only 360K with only even numbered tracks recorded plus track 79 recorded 40 times.

Also at this time I was trying to connect my 8"/3½" drive subsystem which I had been using with my Gemini. It didn't work. The scope showed that 2 device select lines were being driven for each drive i.e. 0 & 1 for D:0 and 2 & 3 for D:1, meaning that it was impossible to add a third or fourth drive. The only apparent and pretty appalling reason for this was the transposition of the wiring sequence of the D0 D1 D3 and Motor lines for D:0 and leaving them alone for D:1 meaning that all

drives can be shipped ready linked as D1 thus avoiding the assembly technician having to think about anything but LEGO mechanics. Talk about a sledge hammer to crack a walnut, the walnut presumably being the assumed size of the technician's brain.

I rang all the people who might know or be willing to find out what I had to change or where the limitation bodge or option was, i.e. in MS-DOS, the AT BIOS, or the floppy controller etc. The word from the IBM camp (dealers, cloners, IBM research staff etc.) was "the AT only supports 2 floppy drives". Great. Sounds like floppy drive interfacing became a state secret. Watch out ohms law, if IBM ever discovers you, you could become an unsupported feature.

By this time the lid was well and truly off my machine with bits spread out everywhere and I was able to observe the behaviour of the drives and sure enough as Dave said in his article the heads were double stepping and therefore spending half their time banging the end stop. In the ensuing experiments I achieved every permutation one could conjure, except for DDDS 730K. These included 730K at 9 high-density sectors per track (leaving 6 sectors worth unrecorded), 80 track single-sided 730K, and even double stepping on the 40 track drive. These did not result from spurious entries in the CONFIG.SYS file. The formats simply did not necessarily obey the manual. My revised theory became:-

1: After boot, at system initialisation the disk drives are addressed, and if present, analysed for number of tracks. This information is retained at least initially.

2: The parameters from CONFIG.SYS are read into the DPBs for user configured drives (D: to L: in my case).

3: When any disk is logged in (even to be formatted), it is checked for density of recording (if any present).

4: The drive is also checked by driving line 2 of the floppy bus into the high density state and if 360 R.P.M. is detected on the SECTOR line the drive is deemed HD otherwise it is double (i.e. normal these days) density.

5: MS-DOS decides which type of drive it must be and places the appropriate parameters in the DPB.

6: It follows from 3:, 4: and 5: that with an unrecorded disk in the HD drive it will never see that drive as a normal density drive.

7: It follows from 6: that one cannot format a blank disk at normal density on a high density drive unless one can convince the system that it is a normal density drive by some other means.

8: It follows from 3 that a 730K formatted disk can be written to on an HD drive and that a previously formatted 730k disk may be reformatted 730K.

9: In the absence of some user induced disk parameters, a normal density disk will be presumed to be 40 track.

The conclusion from the foregoing is that to get a 730K disk formatted from scratch one must:-

a) have the right disk parameters,

b) not have done any disk work which might overwrite these,

c) fool the system into thinking the HD drive is normal density.

One way of achieving c) is to intercept line 2 of the floppy bus, and this is effectively what I've done (I actually wired a switch in to one of the link options on the drive)

I now successfully format 730K by rebooting, throwing the switch and then formatting. I can then put the switch back to normal and continue to operate at 730.

I'm still working on the IBM/floppy bus problem and I would like to learn how to get deeper into the operating system to find a more satisfactory solution.

My current CONFIG.SYS is:

```
BREAK=ON
BUFFERS=10
COUNTRY=044
DEVICE=ANSI.SYS
DEVICE=DRIVER.SYS /D:0 /T:80 /S:15 /H:2 /F:1
DEVICE=DRIVER.SYS /D:0
DEVICE=DRIVER.SYS /D:0
DEVICE=DRIVER.SYS /D:0 /T:40 /F:0
DEVICE=DRIVER.SYS /D:1 /T:40 /F:0
DEVICE=DRIVER.SYS /D:2
DEVICE=DRIVER.SYS /D:3 /T:77 /S:26 /H:1 /F:3
DEVICE=DRIVER.SYS /D:3 /T:77 /S:26 /H:2 /F:3
DEVICE=DRIVER.SYS /D:129 /C /N /F:6
DEVICE=VDISK.SYS 384 /E
LASTDRIVE=M
```

Hope this didn't bore the pants off you, Regards

John Parrott

P.S. It is now clear that once disks are formatted, the system is quite happy using just logical drives A: and B:. It is probably unnecessary to have all those entries in the CONFIG.SYS file and I shall experiment with using a batch file to do formatting and this will contain temporary parameters and finish with a call to AUTOEXEC.BAT to restore normal system values.

P.P.S. I'm not sure I can get at these values from a .BAT file but I'm working on it.

80-BUS to/from Amstrad PC1512 by A.A. Bryan

This describes one method of transferring text and data files between a Nascom/Gemini hybrid and an Amstrad PC1512 computer.

Assuming that the following Nascom/Gemini system configuration and software is available:-

Nascom 2 with GM802 64k RAM and GM809 or GM829 disk system, with 2 x 5.25" Floppy Disk Drives (or equivalent Gemini or other system), CP/M Operating System with SYS Custom BIOS configured for IBM SSDD on Drive B:.

1) Format a disk on PC1512 using DOS-PLUS operating system and DISK Prog. Format Option: 160K CP/M Disk.

2) Insert CP/M 160 formatted disk (from 1) into Nascom/Gemini, Drive B:.

3) Transfer files from Nascom/Gemini, Drive A: to Drive B: using PIP, GEMPEN, etc.

4) Move CP/M 160 disk from Nascom/Gemini machine to PC1512, Drive B:.

5) Transfer the data to a DOS 360 formatted disk using PIP or COPY on DOS-PLUS or COPY on MS-DOS operating systems of the PC1512.

6) A reverse procedure to transfer text files from Amstrad PC1512 to Nascom/Gemini works all right if care is taken not to overload the available RAM and disk-file space on the Nascom/Gemini machine. Large text files over 48K in size should be split into smaller, more manageable parts before attempting a transfer.

Ed. - a far easier, but costlier, method is to use Gemini BIOS 3.2 or later, along with the optional (and in my opinion over-priced) IBM-COPY program. Or make use of Scorpio Systems' *disk transfer/copying service !....*

A Brief Look at WordStar 4 by P.A. Greenhalgh

This issue of Scorpio News is brought to you courtesy of WordStar 4. This is the latest version of this "classic" word processor, and it has a large number of enhancements over previous releases. Some of these are trivial, a few should have been in it from Version 1, and several are a boon. This is a very quick look at a number of its new features, and it will probably only make full sense to those of you familiar with earlier releases of WS.

For a start, there are three new dot commands included that make life so much easier. These are .lm, .rm and .pm - set left margin, set right margin and set paragraph margin (i.e. indentation of the first line of each paragraph). You can insert these as and when you like throughout a document, making reformatting (with ^QU, reformat from cursor to end) a totally painless affair.

There is also extensive support for a wide range of printers, including control of those with proportional character sets (e.g. lasers and many matrix printers). This is still somewhat of a "fraud", but it is a definite step forward for mankind. The "fraud" aspect of it is that WordStar still only justifies with a fixed number of characters per line, rather than being completely variable, as in typesetting where a line is of a specific physical length (e.g. x cms). However, WS4 contains width tables for the particular font in use, (which I haven't got quite right yet!) works out how much of a given line the text is going to occupy, and thus how much blank space there will be in total, and divides this blank space up into the gaps between words. You can see this by looking at this page; the last line of each paragraph has the correct spacing (unmodified by WS4) - look at the other lines in the paragraph. By the way, we now get 20% + more text per page!

The <ESC> key now has new functions. Press <ESC>! and "5:16 PM" appears, <ESC>@ and "8 September, 1987" appears - quite handy. However, the main functions of <ESC> are user-definable, and <ESC>y definitions, where y is any letter or number, may be set up to return whatever sequence of characters that you want, including control characters and sequences. This can be very useful.

Mail-merge is now a standard part of WS4, as is a spelling checker and thesaurus. The spelling checker is quite fascinating, as it works phonetically. For example, present it with sikiatryst, and the alternative it offers you is, quite correctly, psychiatrist. A single keystroke makes the substitution for you. It will, like most spelling checkers, allow you to build up your own personal dictionary, so that for example "Nascom" no longer throws up the "correction" of "Nazism" !!

The thesaurus is also fun (and useful!). Point to a word and hit a particular key combination. You are presented with the alternatives. Move the cursor to the one you fancy, press return, and the substitution is made for you. For example, I will now put the word "thesaurus" several times, and then use it to show you the options offered: dictionary, glossary, lexicon, list, synonym finder, synonym listing, vocabulary, word finder.

Help is now far more comprehensive, press ^J followed by the command that you are interested in, and you will be given a brief description of its use. This of course means that you need to know the commands to begin with (!), but you can see these, with their one or two word descriptions, if you have the top-of-screen menus turned on.

There is now an un-delete function. Each time you delete something, either a word, line, block, whatever, it is stored in a temporary buffer. ^U can be used to restore the buffer to the screen. A neat quirk is that the restore is done at the CURRENT cursor position, so if for example you have two words transposed, you can swap them by going to the first and typing ^T^F^U (delete word, move right one word, un-delete word). The buffer is of a finite, user-definable, size and if the delete you are about to do will not fit into the buffer, you are given a warning that you will be unable to un-delete it.

Maths functions are built-in. If you "block-mark" a horizontal column of numbers ^KM will add them up for you. Alternatively, ^QM takes you to a calculator, which has log, sin, square root functions etc. After you exit, <ESC>£ can be used to bring the last equation typed into your document, and <ESC>= will bring in the last result, e.g. log(int(sqr(64)*sin(30))^5) = 2.3856062736.

There are many, many, more features that I do not have the space to cover here. Operation of most functions is MUCH faster; a utility, WSCHANGE, can be used to set up just about anything and everything; and so on. WS4 is, as far as I am aware, only available for MS-DOS. Unfortunately I doubt that it will ever find its way to CP/M-80.

Gemini GM886 - Intel 80286 80-BUS CPU Board

Gemini have announced the imminent availability of an 80-BUS 80286 based CPU board - they are saying October. Price is to be £525 (+VAT). Below is the brief specification, extracted from the technical notes that Gemini have released so far.

This product has the following principle features:-

- 80286 CPU running at 8 MHz (Optional 12 MHz) Clock Frequency
- 1 Megabyte of Dynamic RAM (with No Wait State Operation via "interleaving" memory access)
- 64 KB of EPROM Memory
- 8250 Serial Port UART - compatible with IBM software
- CMOS Battery Supported Real Time Calendar Clock
- Operates in both Real (8086) and PVM memory modes
- Dual Parallel Port - with printer Handshake option
- Full Interrupt Control of peripherals, including 80-BUS peripherals
- Option of Advanced DMA Controller to improve transfer rates
- Full Memory and I/O interface to 80-BUS - including Word Transfers
- Port conversion - addressing ports 340H-342H will access 80-BUS ports A0H and A1H. For running IBM Pluto software.
- Full speed 80287 Numeric Co-Processor operation possible
- 80-BUS interface port - allows Dual Processor operation
- Reset Control facility
- Full RS-232C interface and Handshakes
- Will run MS-DOS Operating System and Applications *(* see Ed.'s notes below)*

Editor's Notes on new 80-BUS Products

In this issue there are articles describing three new 80-BUS CPU boards. As none of these products is actually in production as I write this, past experience tells me that there may well be changes to their specifications, even if only minor, before they are generally available.

I trust that the articles have whetted your appetite, but would strongly recommend that you obtain firm specifications from the manufacturers or their dealers before committing yourself to a purchase. It may well be that there are limitations as to what other 80-BUS boards may be used in the system with these new products (e.g. you may HAVE to have an SVC rather than an IVC, or a GM849A rather than a GM809, GM829 or GM849, etc.), so do check first.

As far as the GM886 80286 board is concerned, the level of software compatibility will have to be taken into consideration. There is little doubt that this will be a high performance product, and you will be able to run the MS-DOS Operating System. However, because of the very nature of the rest of the 80-BUS system around this new board, particularly the video side, this means that there must be some limitations as to what MS-DOS applications can be run. Do NOT expect to be able to run the infamous *Flight Simulator* program !

Please note that the above few paragraphs are included here, not to put you off purchasing the boards, but to make sure that you make yourself aware that what you may be purchasing will do what you want. Happy shopping!
