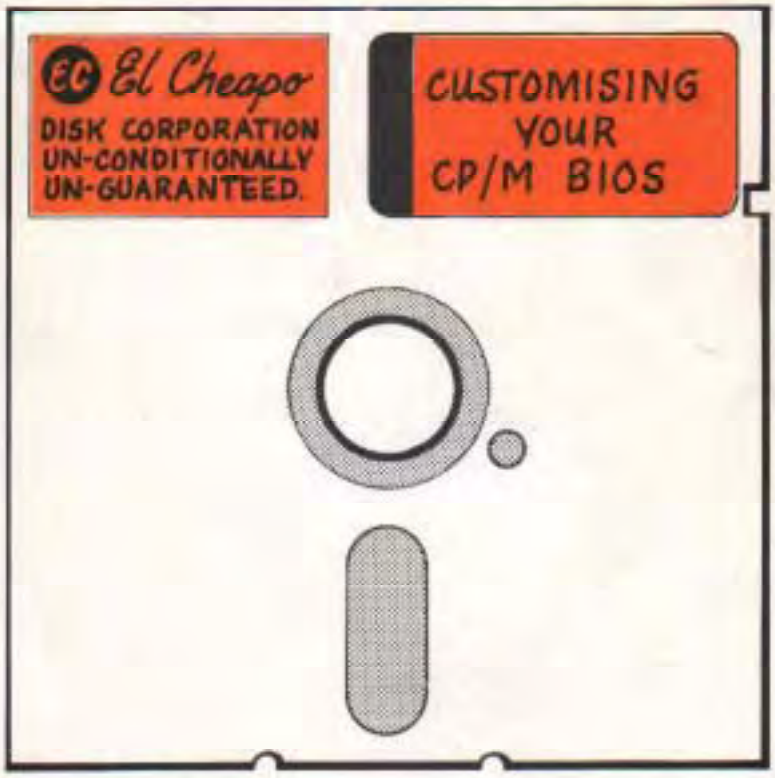
80-BUS NEWS

JANUARY–FEBRUARY 1983 VOL.2 ISSUE 1

- Z 80 INTERRUPTS
- BOOKS REVIEWS
- UCSD

J. S. WRIGHT G3VPW
5 WARMANS CLOSE
WANTAGE
OXON. OX12 9XS
ENGLAND



£1.50

The Magazine for
NASCOM & GEMINI USERS

January-February 1983 **80-BUS NEWS** Volume 2. Issue 1.

 Published by Interface Data Ltd. and printed by Excelsior Photoprinting Ltd., High Wycombe.

SUBSCRIPTIONS

Annual Rates	UK	£9	Rest of World Surface	£12
	Europe	£12	Rest of World Air Mail	£20

Subscriptions to 'Subscriptions' at the address below.

EDITORIAL

Editor : Paul Greenhalgh Associate Editor : David Hunt

Material for consideration to 'The Editor' at the address below.

ADVERTISING

Rates on application to 'The Advertising Manager' at the address below.

ADDRESS: 80-BUS News,
Interface Data Ltd.,
18 Woodside Road,
Amersham, Bucks, HP7 OBH.

EDITORIAL

Publishing Deadlines

All things being relative and equal, this issue should be the third one that you have received after fairly accurate two monthly intervals. Having achieved this amazing regularity, it now becomes possible to start announcing publishing-schedules. For a start, as you know, we do not have a great army of staff-writers, in fact we don't have any at all! Consequently we are pleased to receive articles for consideration at any time, but would prefer it to be well before the end of the second week of each odd month, if there is to be a chance of getting the article in the next issue.

Potential articles may be submitted in any form, but their relative chances of publication range from disks [any Gemini format (SD, DD or QD) and many others (Superbrain, Osborne, Xerox, IBM 8" etc - but what are you doing with one of those!)], down through tapes [Nascom 2/3 or Gemini RP/M formats, or N1 with CUTs], and then neat typing/handwriting on clean, uncrumpled paper. The lowest chance of getting something published is obtained by sending in an Nascom 1 format tape, as we can't read those! Crumpled loo paper is a better bet!

Advertisers should send in camera ready copy by the end of an odd month, and should, preferably, book space about one week before that. Rates are available upon application, but you can be assured, of course, that they are extremely reasonable! All private ads. are free of charge, providing that they are advertising unwanted hardware/software, and are not for 'commercial gain'. Clubs may also send in details of their meetings and activities, and these will be published free of charge when space permits.

Maintaining Order

I am currently trying to draw up, for publication, a chart showing all of the Z80 I/O ports occupied by the various 80-BUS/Nasbus products, the aim of which is to ensure that there are no clashes. Unfortunately, I am not getting on very well with it. I have spoken to a variety of manufacturers, all of whom have promised to send me details, but most of whom haven't. So, if you manufacture, or are about to manufacture, an 80-BUS/Nasbus compatible board that uses I/O ports, please send me:

1) Details of the standard port assignments.
2) Details of the optional assignments (if any).
3) Does the board support NASIO and DBDR correctly?

Hopefully I will be able to publish a complete I/O map in the next issue. If it is incomplete I will put in **VERY BIG LETTERS** the names of the unobliging companies.

Moving Technology

This Editorial is brought to you courtesy of Wordstar 3 and a Gemini Galaxy 3. Wordstar is very much the 'industry de facto standard' CP/M word processing software. It is, in many ways, nowhere near as easy to sit down in front of and type away as the 'PEN series (Naspen, Diskpen, Gempen), but I have been using those for four(?) years now and therefore have gained a certain understanding of them. Wordstar, on the other hand, has some pretty amazing features, the least of which must be the **highlighting** and the underlining facilities, not to mention the $_{\text{sub-}}$scripts and the $^{\text{super-}}$scripts, and a massive host of other features. I'll let you know how I get on. The Galaxy 3, by the way, is as Galaxy 2 but with a 5.4 MByte Winchester and 800Kbyte floppy, and on this one there is a Qume Sprint 5 attached. I can't work out how I ever managed to produce the old INMC mags using a Nascom1, cassettes, and an arthritic IBM typewriter. Those were the days.....

And finally, talking of moving technology(!), please note our **NEW ADDRESS** as given on the opposite page. By the time you get this we should have moved.

LETTERS TO THE EDITOR.

Right of Reply.

May I claim the accused's right of reply to Mr Perkins review of HS-1N (80-BUS Vol. 1, Issue 4)? But first, congratulations on a very thorough piece of detective work on the hardware and firmware!

We designed HS-1N for our own use in mid 1980, because we had Philips DCRs and Nascom 2s, and no prospect of 'official' disk drives. Following a demonstration of the system at the Scottish Amateur Computer Society in Dec '80, we were approached by Microspares and we negotiated a license for them to manufacture the product. We accept now that further work should have been done on the design at that stage, but we simply passed the prototype designs to Microspares and let them deal with the PCB, manufacture, manuals etc.

Design bugs:- We had already implemented NASIO on our N2s, and did not need DBDR, and hence did not implement these on-board in our prototype. They are implemented on current boards. The conflict between the SIO and Page Mode memory arose because we received assurance from Nascom in mid 1980 that no existing or planned product would use ports F8-FF. Current boards are supplied using ports 78-7F.

Firmware bugs:- The bugs in 'Initialise' are real - no excuse is available, except that if the system is properly used, and no tape is removed without 'eXiting', the bugs remain hidden! The criticism of the error handling routines (which use the HS-1N warm start after issuing the error message) applies equally to Nascom ROM BASIC when used by machine code calls, but the revised operating system (HS-1N+) avoids this by using a RAM vector, which can be patched to divert error routine calls back to a user's program.

I accept the criticisms of the absolute calls to Nas-Sys, and the use of address 0 to discard unwanted data. It takes several months of use to be reasonably sure that there are no major bugs in a system, and in this case pragmatism defeated the desire to rewrite the program and and purge its impurities!

ZEAP files were not directly supported simply because we do not use ZEAP, but the faster but apparently no longer obtainable Z2, which we have patched to allow direct access to HS-1N+. Relocation of files as they are loaded is supported by HS-1N+, and is an essential feature of HS-Kit, the toolkit for Xtal BASIC and HS-1N+ which Mr Perkins referred to in his review. This allows access to files by name from Xtal BASIC, but the additional code for this is just too much to be incorporated into a 2K operating system ROM. In our initial specification we deliberately chose access by number, not by name.

Yours sincerely,
Dr M.D. Hendry, HS Design Ltd.,
Linley, East Rd., Cupar, Fife. KY15 4HR.

Pascal facilities and PIO ports

Please allow me to comment on some items that cropped up in the Nov/Dec '82 issue.

Dr. Dark raised a couple of queries with regard to Pascal. He claimed that Pascal does not have the same simple string-handling facilities that BASIC

has, giving the example:

```
A$ = LEFT$(B$,4) + MID$(A$,3,3) + " Wowee!"
```

In fact, Pascal has these facilities in exactly the same way as BASIC: they may or may not be provided with the compiler/interpreter as "standard" or "intrinsic" functions. Most of the Pascal compilers around, of course, are really only "Tiny" Pascals (i.e. a subset) and many don't even allow the user to define his/her own data types (the solution suggested by Dr. Dark). The Nascom Pascal, approved by Lucas Logic, however, does indeed have the relevant functions and is probably one of the few Tiny Pascals which do.

Most of the Pascal implementations which reproduce, or nearly reproduce, the full Jensen and Wirth specification will also fling such functions in as standard intrinsics as a matter of course. In UCSD Pascal, for instance, Dr. Dark's line of BASIC could be re-written as:

```
astring := concat(copy(bstring,1,4),copy(astring,3,3)," Wowee!");
```

where astring and bstring (and the functions copy and concat) are STRING types (equivalent to PACKED ARRAY OF CHAR of dynamic length).

Elsewhere in the same article he attempts to tackle the problem of handling integers with a large number of digits. Again, if he had UCSD he would not have to do it the hard way. UCSD Pascal allows the use of a standard data type called LONGINTEGER which can be declared to an arbitrary precision up to 36 digits and can then be used with the usual arithmetic operations.

Big H, in the same issue, details his method of utilising other "Centronics" signals besides BUSY and /STROBE. However, his PIO Port assignments are not consistent with existing conventions for BUSY and /STROBE. Both Nascom and Gemini, I believe, now employ the convention that Port B is used for data and Port A for control with bit 0 assigned to BUSY and bit 1 to /STROBE. It would be helpful if other pioneers could maintain consistency as they progess further into the jungle!

On my own system I am experimenting with the assignment for Port A:

BIT	FUNCTION	I/O?
0	BUSY	I
1	/STROBE	O
2	PE (Paper Empty)	I
3	Not used	
4	Not used	
5	/INIT	O
6	/AUTO FEED XT	O
7	/ERROR	I

/ERROR, in particular, is assigned to bit 7 so that a simple RLA will enable the Carry flag to show its status.

Hope this is helpful.....

Mike York, London.

The Interrupt System of the Z80

by R. O'Farrell

The interrupt structure of the Z80 microprocessor chip is one of the least understood and at the same time potentially most powerful facilities provided by this i.c. My purpose in writing these notes is to survey the interrupt structures offered us, to encourage other users to try these out for themselves.

It should be remarked immediately that it while it is relatively easy to use one interrupt at a time, a complex set up is not easily obtained on a general purpose machine, although on a dedicated machine such as a process controller it is possible. It is very likely that a Nascom or Gemini system will be used as a development system, to develop software for a purpose built dedicated controller. This possibility underlies the following notes. Whether a dedicated controller will be used, or the system software of a Nascom/Gemini modified to support multiple interrupts is a matter for the individuals concerned, but it is essential that each interrupt be set up and carefully debugged along the lines we will examine.

By way of an aside, let me remark that a dedicated controller is no great deal. Consider a minimum Z80 system. This could comprise a similar configuration to that shown in Section 9 of the Mostek manual - a Z80, an EPROM, some RAM and a PIO. It would also need an oscillator to provide a system clock. A minimum system of this nature would not need any buffering, and could possibly be expanded with another PIO or two before that became necessary. Such a system could be wired up on a prototype board. What would it do? The PIO would allow it to sample or switch up to 16 lines without getting involved in elaborate multiplexing circuitry. A reasonable assumption might be that it would treat eight of these as inputs, and the other eight as outputs. An example of what can be achieved in this way was published in Micropower, Vol 2, No. 1, where it was shown how a Nascom could control a washing machine. Another example of what a dedicated system might do is to control a dot matrix printer, such as the IMP or EPSON.

As computer enthusiasts, we are concerned always with the finer points of Z80 usage. We have two problems - to get the hardware to work and to get the software written and debugged. These two problems are complementary and depend to a certain extent one on the other. In these notes, I will deal only with the software side of matters and will assume that the hardware is taken for granted and assumed correct - that all signals are clean, bounce free, and of the correct voltage levels. This is not always so!

The first interrupt structure on the Z80 is the Non Maskable Interrupt, known familiarly as the NMI. Those familiar with the pinouts of the CPU chip will know that pin 17 is called /NMI. This is an input, negative edge triggered. That is, the transition from high level (normal condition) to low level tells the CPU that an NMI is required. The CPU examines the state of this pin at the end of every instruction, and if the pin is active, then it proceeds to service an NMI. There are a small number of conditions which will prevent the NMI being recognised, and I'll mention these to get them out of the way. The first, and not very likely condition is if WAIT states are continually being requested the current instruction is prolonged and never reaches an end. In consequence the NMI test is never reached. Such a condition is hardly likely to arise in serious use of a Nascom or Gemini, as the dynamic memory would not be refreshed, and the program lost. It might happen in use of a Z80 based controller, where the machine was executing Wait states, waiting for a very slow peripheral to react - said peripheral having perhaps gone on the blink! The other failure to see an NMI is if /BUSRQ is active, i.e. if something like a DMA chip has control of the bus.

Having got these two exceptions out of the way, let us now look at the effect of an NMI. When the CPU recognises that the NMI line is active, it immediately saves the address of the current instruction, performing a call to location 0066H. Here, in theory, is the service subroutine for the NMI interrupts. This service routine should do whatever is required on an NMI and finish with a RETN (RET from NMI) instruction. This RETN instruction behaves like an ordinary RET instruction in popping the return address off the stack, but also acts as a signal to the CPU that the NMI routine is finished, and that the previous status of the normal interrupt structure can be restored. An NMI will automatically disable any other interrupts, saving the status until its own service routine is finished. So, if interrupts were enabled before an NMI, they are enabled afterwards, and contrariwise.

On the Nascom, the NMI is used for a specific purpose, well known to those of us who are into machine code. This is to provide a Single Step facility. (On Gemini systems the NMI is uncommitted, and Single Stepping is provided in software by Gemdebug.) The Nascom Single Step works as follows: on Execution of a command, a bit is set in Port 0. This bit is clocked through a succession of flipflops, ending up toggling the /NMI pin of the CPU. The CPU immediately jumps to the service routine, which looks to see if the instruction executed was an E or S. If E, the CPU carries on with the next instruction, but if S, it prints the register display.

As the Nascom has ROM at location 0066H (unless it is set up to run CP/M), you might assume that other use of the NMI is out of the question for Nascom users. Not so! At 0066H is a jump to 0C7DH, and this address is in RAM, so we can patch in here the address of our NMI handler. N2 owners can connect an NMI switch to their machine, which triggers a monostable to give a nice clean pulse, normally causing the register display to be printed out. N1 users can get this with a little modification published in INMC No.2. Its use is for those occasions when a program locks in a loop due to a programming flaw. Hitting NMISW can find where this loop is, and speed up debugging. N2 owners can also connect an NMI signal to Bus line 21. Zilog suggest that the NMI be used for catastrophic events, such as a power failure, where big capacitors on the PSU lines could maintain power for a short time after mains failure, long enough to save the data to disk perhaps. Using it to provide a single step facility will be regarded by most Nascom Users as quite satisfactory, but I deal with the NMI specifically because it is a free interrupt handling method, and when using the Z80 as the basis for an intelligent controller of some sort, it might provide an elegant simplification of the unit. To give you an idea of the potential power of the NMI, it is the method by which the IMP printer is driven. Within the IMP, an NMI is caused at regular intervals. The internal CPU (a Z80 - what else!) switches the address of the NMI routine around between a number of routines to look after checking for new data and printing each column of dots.

Now we move on to the 'normal' interrupts. There are three schemes. These are called Interrupt Mode 0, 1 and 2, and invoked by instructions IM n, where n is the appropriate number.

Interrupt Mode 0 is identical to the 8080 interrupt response mode. At the end of each instruction, the CPU looks at /INT (pin 16). If this line has gone low, something has requested an interrupt. Note that this line need not be low when examined. Its status is internally latched and reset when the CPU reads it. The CPU acknowledges this if interrupts have been enabled by making /M1 and /IORQ lines go low. The interrupting device is supposed to recognise this signal and to place the next instruction to be executed on the bus. Normally this instruction is a restart, which is a call to one of 8 specific locations, but it can be an actual three byte call if you wish, and if you can sort out how to do it! While acknowledging the interrupt, the CPU executes two wait states to allow time for the peripheral devices to sort out which one has priority. When it accepts an

interrupt, the Z80 CPU automatically disables interrupts, that is, it won't accept any more until it is told to. This has the advantage that you can control where your routines are interrupted. Say for example, that you had a disk unit, which was driven under interrupt control. Every time it needs data, it interrupts for it. Some disk units read data every 25uS, and if the data is not accepted in time by the CPU they report an error. In addition to this disk, suppose you have a RTC which ticks every 1/10 sec. Each tick is signalled by an interrupt, and the CPU writes the time to the corner of the screen.

So, you are reading sectors from your disk and your clock is ticking away. In the midst of the reading you get another interrupt for a tick, go and service that, and return to your reading service routine. Unfortunately, your tick service routine takes some 2 or 3 mS, and when you get back to reading, you've missed about 100 bytes so your disk reports a bad read. So you try again. Same story. The only way you can deal with this is to let the CPU disable interrupts on the first interrupt, and not to reenable them again until the read is finished. Fair enough - a read is so quick that it won't matter if the clock on the screen is not updated for a few 1/10s of secs. There is another way to deal with the problem - interrupt priority. This assigns to each device a certain priority, and only devices of higher priority than the device under service can interrupt. We'll come to this in a moment or two. The 8080 interrupt response (IM 0) is automatically invoked after a reset, with interrupts disabled. This mode allows (using the restarts) eight different interrupt handling routines. It is possible to make a number of devices share common handing routines, or for the appropriate routine to be selected for the device by the handler. This means that the handler has to identify which device has interrupted, which adds to the time overhead. In S100 systems, which were originally 8080 based, interrupt response was often obtained by using unintelligent devices. When the peripheral response was required, they did nothing. Using open collector bus drivers, the effect of the pull up resistors was to make the bus read as high, so the CPU thought that there was a byte OFFH on the bus. This meant that RST 38H was most frequently used. This provoked Zilog into providing Interrupt Mode 1.

IM 1 is similar to the NMI response. On interrupt, the CPU executes a call to location 0038H. This allows only one interrupt handling routine, but has the advantage of hardware simplicity. In a controller, there could be two interrupts, the more important one on the NMI line, and the lesser on IM1. The lesser interrupt could be disabled by the program at will, and by the NMI routine. The NMI routine would always take priority. In IM1, the peripheral device need not place any byte on the bus. Use of that mode will automatically get you to location 038H on interrupt.

The last interrupt mode of the Z80 is IM2. This is at once the most interesting and the most baffling interrupt method. What happens is this: in the I register, the CPU holds the high byte of an address of a table which points to all the interrupt handling routines. On interrupt acknowledgement, the peripheral places the low byte of an address on the bus. The CPU reads this byte and builds up the address of a line of the table. From this line of the table, it takes the address of the appropriate interrupt handling routine. The confusion arises because the address built up by the CPU and peripheral is not the address of the interrupt handling routine - it is the location where the address of the interrupt handling routine is stored. Further complication is introduced by the fact that we have to set up the 'I' register. This cannot know the address of the interrupt service table otherwise. We also have to tell the peripheral the appropriate line of the Interrupt Service Table. We must not forget also that we must construct this table, and all this before we Enable Interrupts. In addition, the peripheral, if a Zilog peripheral, is so complex that we have to spend a little time letting it know just what we want it to do.

Using this interrupt mode, which is in general known as 'Vectored Interrupt', we are more or less obliged to use the Z80 family of peripherals to allow us to avail of the vectoring. It may be possible to use other devices to the same effect, but this will depend on the device. Zilog make two families of peripheral devices. Designed as full members of the Z80 family are the following:

PIO giving two 8 bit parallel ports
CTC giving four counter/timer channels
SIO offering two serial input/output ports, with high speed synchronous facilities.
DART a reduced specification SIO, best thought of as an intelligent UART
DMA offering high speed transfer of data from port/memory to memory/port, with search facilities.

In addition to these devices, Zilog also offer another family, the 8500 peripheral device family. These devices are specialised microcomputers, which are set up to be input/output devices of astonishing complexity! I propose to deal only with the PIO and CTC at present, as these are the only two devices on which I can claim any experience. The SIO and DART I may be able to deal with in the future as Gemini have an SIO board in design. [Ed. - using SCCs from the 8500 family, I believe, not SIOs.] The DMA chip presents problems when using an N1 with Nascom bufferboard, as it becomes necessary to modify the buffer board to allow the DMA to reach into the 4k block of the standard Nascom.

Let us now park the CPU interrupt handling to one side, and consider first the PIO. This device offers a number of operating modes. It comprises two 8 bit parallel ports, each with two handshaking or control lines. As normally used, it appears as a block of four ports, with the data ports A & B, and then the corresponding control ports CTRLA & CTRLB. On the standard Nascom, this block occupies Ports 4,5,6,7. Port A is number 4 with its control port CTRLA number 6. The two data ports are Read/Write, the two control ports are Write only. We can set this device up in a number of operating modes. First of all, we can pick Mode 0, which configures the port as Output, say driving a parallel printer. The two handshake lines are used here, one from the PIO to indicate that the data is ready on its output pins, and the other from the peripheral device to say 'Thank you, I've got that'. Mode 1 sets the port up as Input, with the handshake lines working in a similar way. These two modes can be set up on ports A or B, but mode 2 differs. Mode 2 sets up port A to handle bidirectional data. As it now needs two hand shakes, one for in data and one for out data, we steal the handshake lines from Port B. So Port B can only be set to Mode 3 if port A is in Mode 2. Either or both ports can be set to mode 3 at any time - it is not necessary to have a port in Mode 2. Mode three is the Bit Mode. It allows the eight lines from the Port to be configured as any combination of Input and Output lines you require. As well, we may instruct the port to interrupt on simple logical combination (AND or OR) of some of these lines. We don't have to use them all for the interrupt, and can also define whether they should cause an interrupt when high or when low. There is one restriction - all interrupting lines must be at the same level.

On first introduction setting the PIO up is so complex, that an example is perhaps the best way to demonstrate. We will set the PIO up to its four modes by way of example.

On power up, the PIO enters a reset state. This means that the internal structures of the PIO are all set to zero, or neutral conditions, as appropriate. We unfortunately cannot get this effect by hitting System Reset on N1s, as the PIO hadn't enough pins to allow for this. Instead, the PIO designers provided that a reset would be signalled if the PIO received an /M1 signal without /RD or /IORQ. To obtain this signal, it is necessary for N1 owners to make a small

modification involving one gate, detailed in INMC no 2. This mod. has been implemented on Nascom 2, Gemini GM811 and 813, and both the Nascom and Gemini I/O boards. When in a reset state, the PIO remains that way until instructed otherwise.

We instruct it otherwise by sending it a Control Word, and in some cases the necessary further commands, all of which we write to the Control Port. First, we load the Interrupt vector, which is indicated by having the LSB zero, i.e., it must start on an even boundary. You will remember that this is the least byte of the address of the interrupt service table. The high byte is supplied by the I register of the CPU. Now, we select an operating mode. This is built up of two parts:

M1 MO x x 1 1 1 1

The 1111 indicates that this is the mode word. The M1 MO values differ according to the Mode, and the x x are don't care values. The mode is easy:

Output = 0	in binary 0 0	Modeword 0 = 0FH
Input = 1	in binary 0 1	Modeword 1 = 4FH
Bidir = 2	in binary 1 0	Modeword 2 = 8FH
Bit = 3	in binary 1 1	Modeword 3 = 0CFH

When Mode 0 is selected, the data byte written to the port is enabled onto the output lines, and the Ready handshake line goes high to let the peripheral know that the data is available. This signal remains until the peripheral handshakes. On the handshake, the PIO will cause an interrupt, if that has been enabled, to indicate that the next byte of output data is required.

Mode 1 is the Input mode. To start the handshake operation, the CPU has to perform a Read from the PIO. This empties the Input buffer, and sets the Ready line to let the peripheral know that the CPU is listening out for it. The peripheral loads the data onto the input register, and strobes the handshake line. This causes an interrupt, if enabled, and turns off the Ready signal. Note that if your peripheral is smart enough, the Ready line can be ignored, provided that you are careful not to load data too quickly.

Mode 2 is the bidirectional mode. This uses all four handshake lines, so that it is only possible on Port A. Port A handshake lines are used for Output control and Port B for input control.There is a difference from Mode 0. In Mode 0, the data is on the Port output lines, and the Ready line (A RDY) is high. In Mode 2, A RDY goes high, but doesn't put the data on the lines until the strobe line goes active.

Mode 3 is the mode we use most often. It uses no handshake signals, and hence can be of use to control a number of unrelated events, such as switching sections of a machine on and off, and reading status of input lines. When we have selected Mode 3, we must then send a control word to the Control Port to signal which lines are input and which output. This is easy to remember, as 1s are In and 0s are Out. During mode 3 operation, data can be read from or written to the port at any stage, with restrictions only when Port A is in mode 2 and Port B in mode 3. When reading a mode 3 port, the data returned is the values of the lines defined as inputs plus the values currently being output on the output lines.

To enable the interrupts, we must write an Interrupt Control Word to each port. This has the following format:

EI &/. H/L M 0 1 1 1

In this, the LSB 4 bits of 0111 indicate Interrupt Control Word. Bits 4 to 6 are only used in Mode 3, and Bit 7 enables or disables the Port to interrupt. If Bit 7 = 1, then Interrupts are enabled. If 0, then disabled. If an interrupt condition occurs while disabled, the PIO latches it, and interrupts as soon as Port interrupts are re-enabled. If we don't want this to happen, then we can set Bit 4 in the Interrupt Control Word to reset the pending interrupt condition. Bits 4 - 6 are used in Mode 3. If Bit 6 = 1, then selected lines are monitored for AND condition, i.e. they must all be active for interrupts. If bit 7 = 0, then any one of them can be active and cause an interrupt. Bit 5 =1 indicates that they should be monitored for High level, = 0 for low level. Bit 4 = 1 indicates that the next word sent to the port must define a mask. Only those lines whose mask bit is 0 will be monitored for generating an interrupt. To enable or disable a Port interrupt without modifying the rest of the Interrupt Control Word, we write the following word to the port.

EI x x x 0 0 1 1

Problems have occurred with the PIO's little peculiarities. Disabling interrupts by writing the above control word to the port is subject to them. For example, you write the disable control word to the port. As you do so, an interrupt occurs. The PIO generates an interrupt, and the CPU acknowledges it. By then, the PIO has deciphered the control word and deactivated the interrupt structure. In consequence, the PIO will not place the interrupt vector on the bus at the appropriate time. The CPU will read such a vector, however. As the bus lines are not in a well defined state, this value will be spurious, giving rise to eccentric behaviour of the interrupt structure. The cure for this is to disable CPU interrupts, send the disable control word, and reenable the CPU interrupts. Here is the sequence of instructions:

```
LD A,03H        ;Disable word
DI              ;Disable CPU
OUT (PIO),A     ;Disable PIO
EI              :Enable CPU
```

This causes the CPU to ignore any such spurious interrupts while the PIO is turning its own interrupt structure off.

So much for the actual instructions for setting up the interrupts in the PIO. Now we have to consider what the Interrupt handling routine is to do. Obviously, without knowing the precise form of your peripheral, this is something I can't deal with in particular, but there are a few points to draw to your attention. First of all, an interrupt automatically disables any further interrupts - not just from the same device, but over the whole Machine, except for the NMI. It is at your discretion when to reenable them. This action will allow higher priority devices to interrupt the service routine currently running. As a rough rule of thumb, timer and floppy disk interrupts are usually accorded higher priority than simple input/output devices. So, suppose your parallel printer is connected up to a port, and interrupts for more data. Usually a printer has a buffer within it, and a print rate of 60 - 120 chars per sec. So long as the printer has about 100 chars in its buffer, it won't need any further attention for about a second. This means that we could, for example, allow our CTC to interrupt the printer routine every half second or so with the time. We can only interrupt a routine when it is in a safe state, so we need to have saved any data resulting from the interrupt before permitting another. Reenabling interrupts is done by executing the EI instruction within the interrupt service routine. To look after the problem of signalling to the lower priority device that the higher priority device that has interrupted it is finished, and that the lower priority unit can now continue with its interrupt service, we have to terminate each interrupt service routine with EI (enable interrupts) if we have not already used it, and a RETI (RETurn from Interrupt). The EI is not acted upon

until one instruction later. This is to allow the RETI to take effect. In the event that it is possible for it to do so, the Interrupt service routine must clear the interrupt condition before it executes the RETI. Otherwise, a further interrupt will not show up. Say we interrupt on A0 or A1 high. If A0 goes high and causes an interrupt, we go to the interrupt service routine. While there, A1 goes high, but the PIO does not signal for a further interrupt. Why? Its instructions are to signal for interrupt if A0 is high OR A1 is high (or both). This means that the internal flag for an interrupt is set and remains set until the RETI. The service routine should clear A0 (if it doesn't clear itself automatically), so that on the RETI instruction, the condition that caused the interrupt has gone false. If after the RETI the interrupt condition is again present - having gone false - then a new interrupt is signaled. If the interrupt condition has not cleared, for whatever reason, the PIO knows that it has signaled for an interrupt for that condition. Remember that the PIO is smart, but not smart enough to distinguish between different causes of interrupts. Anything that causes a valid interrupt within the PIO is equal to any other valid interrupt cause.

The CTC is another interesting chip. It contains four channels, which can be configured as counters or timers. Each channel can be programmed to count down the system clock, divided by specified scaling factors, or to sample transitions of an external line, and interrupt on a specified count. This chip could, for example, allow you to use the standard Nascom UART so as to give interrupt driven cassette handling. In certain machines it is used to give a time of day clock, at the annoyance of having to enter the time as part of powerup procedure. I would not recommend it in such an application. Instead, I'd recommend most strongly a Real Time Clock, such as already written about in both INMC80 and Micropower. A use for the CTC that Zilog suggest is to use one of these units to prioritise four non Z80 system interrupts. The ready lines from these peripherals are connected to the four ports of the CTC, and it is configured to react to a count of one. Then when one of these lines goes active, the Interrupt service routine proceeds to service the appropriate peripheral, as if it were a Z80 type peripheral.

Those who read David Parkinson's interesting article on finding and optimising the most used sections of code in a program may like to know that it would be possible to program a CTC to cause regular interrupts to the program for profiling purposes, if you are SURE that the program does not disable interrupts at any stage. Alternately, an output from one channel could be made to cause an NMI at intervals in lieu of the little circuit he used.

All this sounds very complex - but it does work! Moreover, it works even if you don't understand why, so long as you do the correct things! What I have written above is all contained in the device manuals, and in the event of any difference between what I have written and they state, they should take priority. I know from my own experience that the interrupts do work, but that it takes a considerable amount of study and work on them to master them. I append a list of references which I have found to be of use. I hope they will prove useful to other Nascom and Gemini users.

References:

"May I Interrupt?", PCW Vol 3, No 6, June 1980, pages 60-63/111
Notes on PIO operation, INMC No. 2, pages 6-8
Understanding the PIO, INMC No. 6, pages 19-20
Using the PIO, INMC80 No. 1, pages 24-25
DIY Real Time Clock, 80-Bus News No. 1, Pages 17-24
Mostek (or Zilog) Parallel I/O Controller Manual
Mostek (or Zilog) Counter Timer Circuit Manual
Washing Machine controls Nascom, Micropower Vol2, No. 1, pages 2-6
The French Connection, Micropower Vol 2, No. 3, pages 2-5 (a RTC circuit!)
Parkinson's Pep-Up, PCW Vol 3, No. 10, October 1980, pages 82-83/123

The Dave Hunt Page(s)

This issue, the DH bit is going to be a hoch poch of bits and pieces, I did have a theme, but since thinking of this episodes' theme Paul has landed me with a load of letters and odd bits to chew through. If I actually get round to my final theme you will see that none of the previous bits fall comfortably into it so everything is going to get split in to small sections 'a la' Dr. Dark. Gone also is the turgid writing style of the last three issues. Not that anyone has complained, it's just that I can't write as fast when suffering from an acute case of 'Esoteric Verbal Grandiloqence'. So back comes the chatty nerve jarring style with those apalling puns that I sometimes throw in.

On the software front, that I have been to a lesser or greater extent involved in, are two interesting developments. Firstly, Richard, having recovered from his holiday on some exotic island, returned refreshed and brimming with new ideas. Those devotees of his versatile SYS overlay BIOS program should note that it has been subject to some Frankenstien type surgery and a new monster saw daylight sometime around the end of January. This is based on an idea that has plagued production CP/Ms because of the number of permutations of soft-/hard-ware configurations available. This has led to a number of different versions of CP/M for sale for Nascom, Gemini and Nascom/Gemini hybrids, and an even larger number of permutations not catered for. Richard hopes to have solved virtually all permutation problems between Nascom and Gemini CP/M machines in one fell swoop, throwing in hard disk and 8" compatibility into the bargain. All this in addition to handling a virtual disk option using either Nascom 48K, Gemini 64K or MAP 256K RAM cards. Those with 48 t.p.i. drives will also be able to enjoy full compatibility with SuperBrain and limited compatibility with Cromenco, RML, Osbourne and Xerox.

The second development is a major redesign of DISKPEN/GEMPEN, amongst other things making versions available for SuperBrain and the Mimi. The whole thing has been redesigned internally, although preserving the existing way in which it works and the main command keys. Peter has been busy on the major extension to allow 'PEN to handle overlay files internally, and to provide access to all the major internal routines through a jump table. This started out as the experimental addition of a HELP facility, where the HELP text would be overlayed onto the screen, but took on a whole different light as soon as it was realised that if space was allowed for help overlays, the space could also be allocated to other things. One of the problems with 'PEN has been the lack of a DESPOOL type facility where 'PEN would background print a file whilst another was being typed. Personally I can't stand the noise of the printer running whilst I'm typing, but other people seem to think this important. Some of the other features envisaged are to incorporate my little MULTIFORMAT multiple column printing utility which is ideal for price lists etc; overlays for the use of true proportional printing daisy wheel printers such as the Qume and Diablo; multiple index facilities; and a whole lot more. How many of the proposed overlays will see the light of day remains to be seen. The list grows longer daily. The new 'PEN manual (which I have yet to write) will contain all the competent machine code programmer will require to write his own overlays, so when the new 'PEN becomes available, sometime March - April, I rather hope that people will start to write and publish their own overlays. Remember that a major part of the philosophy of 'PEN has been that it is cheap, and with that is mind, upgrade 'PENs with a modest assortment of overlays will be available at a charge of £10.00 - £15.00 on production of the original disk. This upgrade will be available from Henry's and Amersham Computer Centre, and other Microvalue dealers should they wish to participate.

So on to the pile of letters and odds and sods which I have been given. Some have listings, and as our policy is to pay per page printed, and that listings represent a lot of empty space, we have taken to squashing listings from A4 to A5 getting two pages onto one. So as to allow us to fit things together sensibly when the time comes to print this lot out, all listings have been lumped together and referred to by the author.

The first letter on the pile is from M. L. Trim, of Garswood, Ashton in Makerfield, Merseyside.

Dear Editor,

This is a program written for use with the Gemini G805 disk system. It was an exercise to help me understand data storage on disk and has proved quite useful in storing my customers' addresses and phone numbers. However, other instructions could be inserted to suit the requirement, for example, a Christmas card list, general address list, etc.

Line 100 sets up the Nascom array size

Lines 110 - 150 is an INKEY$ routine

The program speaks for itself being a batch of small routines which can be located in line 840.

Your faithfully, M. L. Trim.

As Mr Trim's listing was supplied as hard copy, I had to type it in and in the process I couldn't resist the temptation to have a small go at it. I changed the input routines and the print routines into two subroutines. As this was written for a version of Basic I couldn't identify, and therefore couldn't get my hands on, I haven't tried it. So I hope I haven't mucked it about to the extent that it does not work.

Next off the pile comes a letter from R.A.C. Treen of Burgess Hill. Some time ago he wrote to me about networks for Nascom owners. This is something that could be interesting and Malcolm Alberry of Leighton Buzzard keeps muttering to me about an automatic disk based bulletin board using his Nascom and 8" drives. I don't know how far Malcolm has persued this, but such a scheme would be very practical for both disk based and non disk computers. The main trouble is modems for the 'phone lines. Two or three simple designs costing up to £30.00 have been published, but as far as I know no-one has looked into these. I wonder if anyone has played about with modems, and if so could they let us know. Mr. Treen is also interested in exchanging tapes, so if anyone is interested perhaps they could drop us a line and we will forward them to Mr. Treen. [Ed. - Perhaps Mr. Treen could communicate with Dr. Dark; see his pages for details.] Mr. Treen continues his letter with a thorough endorsement of the Level Nine Adventure program:

.... On a completely different subject, I purchased, from Level 9, a copy of the 32K Colossal Aventure recently. (I have a Nascom 2 under NAS-SYS 3 with 32K of user RAM.) My first impressions were that it was superb. The progam comes on a TDK D46 cassette (I have had ZERO TAPE ERRORS with this brand, YES, ZERO). The program is recorded at 1200 BAUD twice on side A and once at 300 BAUD on side B. The program loaded first time. With the program is an eight page A5 booklet. This contains the scenario for, and explicit and concise instructions on, playing the game. Included in the game is a stamped addressed envelope for you to request your free clue [a novel way of ensuring program registration - DH].

The game itself is the standard mainframe adventure with of course some modifications. The end game (which I have not yet reached) is supposed to contain 70 rooms instead of the usual 2. I have been playing the game for about 15 hours in all and my highest score is still only 165 out of a possible 1100. If you liked the mainframe adventure then you'll like this. If you haven't played Adventure, then I suggest you do. See the ads in the April - June 80BUS News, I consider it well worth the money.

P.S. Does anyone know how to open the clam .....

Yours sincerely, R.A.C. Treen

And, so on to Robert Wood of Cardiff who has sent two small and simple routines:

Dear Editor,

Readers might like to try the little routine listed below, which could be called 'Nascom Advertising Display'. It is a neat display whereby a message travels across the screen, descending line by line until it reaches the bottom, then there is a flurry of multiple displays. The message then goes to the top line and the next message starts. It is useful as an eyecatcher and can be left running if line 120 is included as this will repeat the whole series of messages.

Program One, as it is entitled has one or two points to be noted:
1) The quotes must be included in the DATA statements.
2) Leave a space between the the quotes and the first word of the statement. (Try without and see what happens.)
3) The variable on line 40 must change with the number of DATA statements. (e.g. with seven DATA statements, line 40 would be FOR I=1 TO 7)

Program Two highlights the neat INKEY$ routine in the Extension Basic reviewed in 80BUS News; it displays the decimal equivalent of the ASCII code of any key pressed, including graphics.

Your sincerely, Robert Wood.

The next one is easy, from David Hicks at Girton College, Cambridge. He wants to know what the 'well published' mod for converting the Nascom 2 screen to 16 TV lines is, as it doesn't seem to have been published in anything he has read. Well, there are two answers, the first is simple and crude but works, and that is to whip pin X of ICYY out of its socket. [Ed. - Really useful Dave!! I think X=1, YY=53.] The second solution was complicated and as I have forgotten where I read it, I'll gloss over it. [Ed. - chicken!]

D. W. Edgar of Greenock, Renfrewshire, has noted that people always seem to be moaning in various magazines about the tape loading of virtually all home computers. He recommends the Binatone Piper Mini Cassette recorder as providing 100% reliability with his Nascom. As the Binatone is only priced at £15.99, it should appeal to all Scotsmen. Personally, I have used a wide variety of cassette recorders over a period of years and have found it difficult to fault any of them. The cheapest was a Pye Mains/battery model that cost £12.95, The most expensive (and noticeably the least mechanically reliable) was a Tandy CTR80 at about £40.00. The Pye is still in daily use in the shop after two and a half years, the Tandy was consigned to the bin about a year ago after spending months back at Tandy's under warranty. The Tandy worked all right, it was simply that bits kept falling off it. In any event, almost without exception, all tape recorders we have tried have produced as near to 100% reliability as it is practical to achieve. The tape used, ah, now that's a different matter. In the situation that we are in, we have tried most makes of tape and have noted that 'el cheapo' tape is no good at all. We also noted that certain tape recorders were fussy about certain brands of tape. The one shining example which would go with almost any tape recorder was TDK D46 or D60, followed closely by Scotch computer tapes. Some of the special computer grade C10's we purchased were useless. So I endorse Mr Treen's recommendation and use TDK.

Bill Ratcliffe wants to know if there are any books on programming techniques for the Microsoft Basic as he says, "My program organisation is chaotically out of control and I need help". Well this one is a lot harder to answer, as sorting programs into a sensible order is a cross between discipline and sensible forward planning, both topics I am the last person to ask about. Most of my Basic programs end up as one hell of a mess, and the only person who understands them is myself (and then only for a short while after I've written them). What I do is to write the program and make it work. Then, if it's for publication I decide what it was I set out to achieve, and jot down the order of the main routines. Then an ASCII dump of the Basic program is put back into my DISKPEN and a 'scissors and paste job' performed, pulling all the common bits together and commenting them. The whole lot is then renumbered using the 'find and change command' starting from the top. A very messy process of program writing, and not to be encouraged. Perhaps we should follow Dr Dark's and Rory's exhortations and both learn Pascal.

I have had one entry in my 'Save a Byte' competition for my 'Simple Hangman' program. W. H. Turner of New Malden writes:

Dear Editor,

This is my entry in Dave Hunt's Kiddies Guide competition. I am an old byte-parer from way back so what an opportunity this is .... Wherever there is a

CALL followed by a RET, both can be replaced by a JP (saving one byte) or better still a JR (saving two bytes). This saved two bytes from Hangman, and further, because I removed a CALL , two bytes of stack space. Further, in CRLF1, where CALL SNDTXT and RET have been replaced by JP, this JP can be eliminated completely (saving another three bytes). The reduction of a CALL - RET sequence to a JP/JR also works for a relative CALL, but saves nothing with a RST.

If you like my random number generator, I can save even more bytes. This is based on SEED=((SEED*5)+21) AND 7FH; and produces an even distribution of numbers between 0 and 127 (i.e. after 128 CALLs it has produced each number once). The sequence cannot be described as truly random as the sequence is always the same so I use the R register to provide the initial seed. The call and return parameters are identical to those of the one used (i.e. called with A=range, and returning with A=0 to range-1).

This random number generator uses only 26 bytes plus one byte for workspace and four bytes to generate the initial value for seed from the R register, which must make it one of the shortest random number generators you have ever seen. The numbers 5 and 21 were chosen to get the best result over the chosen range and for a different maximum a different pair must be used (I used a GEC 4070 to do the calculations, but a Nascom could do it as easily).

Yours sincerely, W. H. Turner.

Well Mr. Turner you win, I should have my wrists slapped for not thinking of that one as it is one of those programming tricks that is lurking at the back of my mind, and, as you say, it saves space. As to your random routine, well that deserves the prize alone, so I'll be sending you the fiver.

Well, the DH bit gets written as and when I feel so inclined, as it is now shortly after Christmas, I am more inclined than usual (perhaps Dr. Dark sent me some of that West Country gut rot he keeps on about for Christmas). A couple of issues back I wrote a piece about data transmission by radio, which to my surprise brought rather more letters than I had expected, strangely all from radio amateurs. This has identified a new area of the 80BUS readership for me. On the basis that I had about half a dozen letters on that one topic, and that normally anything else only prompts two or three, I can only judge that radio amateurs form a substantial part of the readership, or, alternatively, that they are a vociferous minority. Since that article, the Home Office have seen fit to issue my licence and I have since discovered (by using my radio) that a lot of radio amateurs are also equally knowledgeable about computers, and that a fair number own Sinclair ZX81s (never mind, they'll learn), Nascoms and Geminis.

All this preamble is a way of getting round to my theme for the next issue or so, that of databases, Mbasic and disks. You see, I have written a radio log keeping program, which apart from serving its designed purpose, has taught me an awful lot about the way to tackle large programs in Basic and something about how to handle small databases. "All right", I hear you say, "What about databases, I'm not interested in keeping a log book so what use is it to me?". "Well", I reply, "The princples are the same whether you're running a log book or keeping a running inventory of the freezer (just in case large mice are persistently running off with next weeks Sunday joint)."

Now what is a 'DATABASE'. Well it's a file stored on tape or disk which contains the data you intend to work with, or to modify, or to delete, or what have you. The database has been so constructed as to allow easy access to the various individual parts of which it is composed. Ok? Not really, that sentence is written in the pseudo technical double talk you see in some manuals. Imagine your cheque book, it's a book isn't it, odd shape for reading, but a book none the less. Let's call that an empty database, it's empty because nothing has been written on the cheques, and it's the writing on the cheques we are interested in not the cheques themselves. Now the cheque book has a number of individual pages, each unique by merit of the serial number stamped on the bottom. Let's call this an empty record, again, empty because it's what's written on the cheque that matters. Now consider the cheque itself. It has a line for the payee, a couple of lines for the amount in writing, and a box for the amount in figures (a different form of the same data), it also has a line for your autograph and another for the

date. Each item, the amount, your autograph, etc, are assigned a specific place on the cheque. The bank gets cross if you put the wrong data in the wrong slots. So lets call the slots for your autograph, the payee, etc, the data fields because that is where the data is to go.

I hope that's tidied up some definitions for you. A database is a chunk (large or small) of data. The database is composed of individual records each numbered in some way so they may be identified. Each record has one or more fields, each assigned to a specific purpose in storing the data, quantity, date, item number, etc. The data base should be given a name, even if it's something uninspired like 'DATABASE'. The records would normally be numbered, and each field within a record would probably have a name, such as QTY for quantity, or ITEM for guess what. It helps if the field names are appropriate to the field use.

Now databases can come in many forms, and one of the most commonly sought after (but least commonly used) is the implimentation for the smaller machine, something like a 16K Nascom without disks for instance. Let's take a fairly typical database, a name, address and telephone list. Now I've found from bitter experience that this requires at least 6 fields per record, four fields of 25 characters and two fields of 10 to 12 characters. Well one thing is certain, a 16K machine isn't going to hold much data, as even at best using string arrays for the fields, the limitation is going to be the memory space available for those arrays. Let's be generous, lets's say the program to handle the database is no more than 4K long, then that leaves 12K available as string space. Now the way Microsoft Basic works is that it sets up a pointer to each string, which is, if I remember rightly, five bytes per string, so each field is going to have a five byte overhead for starters. So our name and address database is going to be about 150 characters per field. So we have room for about 85 names and addresses in the database before we run out of space. This is plenty big enough for the home user, but then the home user isn't going to wait a couple of minutes loading up the data from tape just to look up Fred's phone number. Far easier to look it up on the jotter by the phone. This is ignoring the difficulty of saving strings as data on a Nascom in any event. String saving has been hammered to death in past issues of INMC and INMC80 so I'm not going to waste time detailing them here.

Of course, there are ways of compressing the data, string compression where the data is shrunk arithmetically to reduce its size. Three characters into two is a typical compression ratio, or even 50% where lower case characters are converted to upper. Another way is 'dynamic' allocation of fields where empty fields are shuffled so that more space is available from the unallocated space. The Nascom version of the Microsoft Basic does this anyway when string space is getting tight. Have you ever noticed the Basic 'hang' for a few seconds, well that's when it's doing its garbage collection and re-allocating the string space. Whatever method is used for compressing data, this is done at the expense of program space, and what is gained on the swings is lost on the roundabouts, so the nett gain is usually trivial.

So what is the answer, well I'm afraid disks is (bad grammar, disks are) the answer, or of course oodles of RAM and some hairy software to get at it. That is, if anyone is intending to use a database seriously. With disk, access is slower than in RAM, but at least with sensible organisation at least the whole thing becomes feasible and the cost is not too astronomical when looked at in terms of price per bit saved. If on the other hand the intention is to simply provide a simple telephone numbers program, then a 16K machine is entirely adequate provided you can wait whilst the tape loads.

So I did get round to my topic after all, next issue I'll be rabbitting on about how databases could be organised, starting from humble beginnings with a simple program for the Nascom. For those too impatient to wait for the next issue, go and get yourself some disks and get hold of Mike Hessey's MANOR program. It's well worth a look at and it's simple to use.

Time and space preclude any more, so, that's all folks, until next time, that is.

Data base handler by M. L. Trim.

```
100 CLEAR 10000 : DIM A$(6,200) : LN=200
110 DATA 27085,14336,-13564,6399,18178,10927
120 DATA -8179,233
130 DATA 31711,1080,-53,536,-20665,3370,-5664,0
140 DATA 4100,3340 : FOR I=3340 TO 3354 STEP 2
150 READ J : DOKE I,J : NEXT
160 GOTO 720
170 REM
180 REM  1. Input list routine
190 PRINT"To abort I/P type 0 on request for customer."
200 FOR I=1 TO LN
210 GOSUB 1110
220 IF A$(0,I)="0" THEN 240
230 NEXT I
240 PRINT"Press R to return to menu"
250 Z=USR(0) : IF Z=82 THEN 720
260 GOTO 250
270 REM
280 REM  2. Display list
290 CLS : A=0
300 FOR I=1 TO LN
310 GOSUB 1220
320 A=A+1 : IF I=LN THEN 420
330 IF A=2 THEN 350
340 GOTO 410
350 T$="Press C to continue"
360 FOR T=1 TO LEN(T$) : POKE 3030+ASC(MID$(T$,T,1))
370 NEXT T : A=0
380 Z=USR(0) : IF Z=67 THEN CLS : GOTO 410
390 Z=USR(0) : IF Z=82 THEN I=LN : GOTO 720
400 GOTO 380
410 NEXT I
420 T$="Press R to return to menu"
430 FOR T=1 TO LEN(T$) : POKE 3030+ASC(MID$(T$,T,1))
440 NEXT T
450 Z=USR(0) : IF Z=82 THEN 720
460 GOTO 450
470 REM
480 REM  4. Save list
490 INPUT"Do you wish to save on disk (Y/N) ";C$
500 IF C$="Y" THEN 520
510 GOTO 720
520 SETCLS(1) : SETNEW(1),"DATA.TX",S : FOR I=1 TO LN
530 FOR J=0 TO 6 : SETOUT(1),A$(J,I) : NEXT J : NEXT I
540 SETCLS(1) : GOTO 720
550 REM
560 REM  5. Load list
570 SETCLS(1) : SETNEW(1),"DATA.TX" : FOR I=1 TO LN
580 FOR J=0 TO 6 : SETINP(1),A$(J,I) : NEXT J : NEXT I
590 GOTO 590
600 CLS
610 PRINT"                              Thank you and goodbye."
620 END
630 REM
640 REM  3. Change list
650 INPUT"Enter number of the item to change ";I
660 GOSUB 1110
670 PRINT"Press R to return to menu"
680 Z=USR(0) : IF Z=82 THEN 720
690 GOTO 680
700 REM
710 REM  Menu routine
720 CLS
730 PRINT TAB(15);"1. Input list"
740 PRINT TAB(15);"2. Display list"
750 PRINT TAB(15);"3. Change list"
760 PRINT TAB(15);"4. Save list"
770 PRINT TAB(15);"5. Load list"
780 PRINT TAB(15);"6. Print list"
790 PRINT TAB(15);"7. Find customer"
800 PRINT TAB(15);"8. End program"
810 PRINT
820 INPUT"Enter number of function required ";N
830 IF (N<1) + (N>8) THEN 720
840 ON N GOTO 190,290,650,490,570,870,940,600
850 REM
860 REM  6. Print list
870 CLS : INPUT"Number of customers to list ";LN
880 WIDTH 255 : LINES 2000 : SETPRON
890 FOR I=1 TO LN
900 GOSUB 1220 : NEXT I
910 WIDTH 48 : LINES 5 : SETPROFF : GOTO 450
920 REM
930 REM  7. Find customer
940 CLS
950 PRINT"    Type 1st 4 letters customer to be "
960 INPUT"    located";H$
970 FOR I=1 TO LN : IF H$=LEFT$(A$(0,I),4) THEN 1030
980 NEXT I
990 FOR I=1 TO LN : IF H$=A$(0,I) THEN 1030
1000 NEXT I
1010 PRINT"Customer not on file" : FOR T=1 TO 2000 : NEXT T
1020 GOTO 720
1030 GOSUB 1220
1040 INPUT"Do you wish to copy to printer ";K$
1050 IF K$="Y" THEN 1070
1060 I=LN : GOTO 720
1070 SETPRON : GOSUB 1220
1080 I=LN : SETPROFF : GOTO 720
1090 REM
1100 REM  Subroutine to get name and address input
1110 PRINT"Customer's name" : INPUT A$(0,I)
1120 IF A$(0,I)="0" THEN I=LN : RETURN
1130 INPUT"and address 1";A$(1,I)
1140 INPUT"and address 2";A$(2,I)
1150 INPUT"and address 3";A$(3,I)
1160 INPUT"and address 4";A$(4,I)
1170 INPUT"and address 5";A$(5,I)
1180 INPUT"and telephone ";A$(6,1)
1190 RETURN
1200 REM
1210 REM  Subroutine to print name and address
1220 PRINT I " ";A$(0,I)
1230 FOR J=1 TO 6 : PRINT "    ";A$(J,I) : NEXT J
1240 RETURN
```

```
Nascom Advertising Display by Robert Wood

 10 REM  *  PROGRAM ONE  *
 20 CLS
 30 RESTORE
 40 FOR T=1 TO 5
 50 READ A$
 60 FOR I=1 TO 16
 70 FOR J=1 TO 48
 80 SCREEN J,I
 90 PRINT A$
100 NEXT J,I
110 NEXT T
120 GOTO 20
130 DATA " NASCOM RULES OK"
140 DATA " A FINE EXPANDABLE COMPUTER"
150 DATA " EXCELLENT LOADING"
160 DATA " AMAZING EDITING!"
170 DATA " MANY LANGUAGES AVAILABLE"

 10 REM  *  PROGRAM TWO  *
 20 CLS
 30 INKEY C
 40 PRINT TAB(1);CHR$(C);TAB(3);"is";TAB(3)C
 50 GOTO 30
```

```
RANDOM  M-80   25 Nov 1982   20:51          PAGE    1

                        Title   RANDOM
                        .Comment "
                                 A generalised routine for producing
                                 random numbers in the range 0 to 127
                                 by W. H. Turner."
                        .Z80
0000'                   SEED:   DEFS    1               ; Workspace with initial seed
0001'  B7               RANDOM: OR      A               ; Done if zero
0002'  C8                       RET     Z
0003'  C5                       PUSH    BC
0004'  47                       LD      B,A             ; Save range
0005'  3A 0000'                 LD      A,(SEED)        ; A = seed
0008'  4F                       LD      C,A             ; Multiply ..
0009'  CB 27                    SLA     A               ;    A ..
000B'  CB 27                    SLA     A               ;      by ..
000D'  81                       ADD     A,C             ;        5
000E'  C6 15                    ADD     A,15H           ; and ADD 21
0010'  E6 7F                    AND     7FH             ; Mask for 7 bits
0012'  32 0000'                 LD      (SEED),A        ; Save back in SEED
0015'  90               RND2:   SUB     B               ; Result ..
0016'  30 FD                    JR      NC,RND2         ;   is in ..
0018'  80                       ADD     A,B             ;      seed modulo range
0019'  C1                       POP     BC
001A'  C9                       RET
                                END
Symbols:
0001'   RANDOM          0015'   RND2            0000'   SEED

No Fatal error(s)
```

CLASSIFIED ADS.

Teletype model 32 (75 baud, 5 unit). Complete and in very good condition. Offers. Ron. Basildon (0268) 22254.

GM803 EPROM board, built and tested, with ZEAP and BASIC. £80, or £40 without firmware. Tel. Nigel Thomas, Kings Lynn (0533) 673308.

Nascom 2 cased with 48K, Nas-Sys 3, Graphics, BASIC Tolkit, Imp printer, Hitachi 9" VDU, software including Pascal compiler, PROM programmer with software. £525. Lancing 765714.

Nascom series A RAM card, 16K fitted, also Nascom buffer card, offers. 051-608-1796.

Gemini GM803 EPROM/ROM board. As supplied by Gemini, boxed with edge connector and manual. Not used. £50.00. Tel. Malcolm Bay, Flitwick (0525) 714205.

UNDERSTANDING CP/M – CUSTOMIZING YOUR CBIOS

by C. Bowden

This article has been written mainly to try to assist those who would like to learn more about some of the operations of CP/M or its support utilities. As a comparative novice myself, I have just struggled through the operations that are described later and I feel that much of what I have learnt could be of use to others eager to get more out of their system. At the end of the article is a summary of the CP/M memory map, and showing the main addresses for CP/M as supplied by GEMINI for the Nascom, and also as altered to allow space for a 'SYS' CBIOS. Terms such as CBIOS, BDOS, CCP and so on are also briefly explained. There are also diagrams in the text, showing the internal details of MOVCPM and the System Track of the disk.

I had been using CP/M for about two years, first with version 1.4, and then upgrading to double density and Version 2.2. During this time the Micro has been used for Games, Word and Data Processing and some Assembler work but I had not devoted much time to trying to find out how the FDC Hardware and Software works, nor to exploring CP/M and its Utilities. I decided that it was time to start. To begin with I wrote a short program to read a few sectors off disk directly into memory in order to learn a bit about disk I/O. This was not too difficult as I simply 'pinched' what appeared to be the relevant bits of code from some of the various CBIOS source files that I had. After a bit of juggling I got it to work and armed with this new but rather vital knowledge I got down to solving the problem that I had set myself - to replace the original GEMINI CBIOS with a SYS CBIOS.

I had several reasons for wanting to change the CBIOS.

1) Primarily to learn how to do it, so as to increase my understanding of what was happening.
2) To avoid the extra wait required for SYS to load.
3) To clear the 'SYSxx' command from the CP/M Buffer so that some other program could be Auto-Run. (Although SYS could probably be made to put another name into the Command Buffer.)
4) The origional CBIOS supplied by GEMINI is very good, but 'SYS' does offer extra features in certain circumstances, so it can become the preferred CBIOS.
5) Perhaps rather trivial - a gain of 8K bytes of extra disk space as the SYSxx.COM file does not need to be on the disk.

I eventually succeeded in my task, but on the way I encountered a number of difficulties, the solution of which greatly increased my understanding of what was going on. Part of the solution involved Disassembling SIMON, SYSGEN and the COLD BOOT LOADER, and I had to alter 'SYS' a little bit as well. During the rest of this article I will be referring to certain programs that I used to carry out the job. To avoid having to repeat multiple names, I will define one suitable common name for the job:-

1)PEEK - A DISK UTILITY such as DDISK or REPAIR that allows direct access to the Disk, with facilities to 'patch' if required.
2)BUG - A UTILITY such as DDT, ZSID or GEMDEBUG.
3)OLDBIOS - the origional CBIOS, as supplied within CP/M by GEMINI.
4)SYS - The new CBIOS. In my case SYS Vn 11.0 at present.
5)MOVOLD - The MOVCPMN or V supplied by GEMINI. (N = version for NASCOM Screen, and V = version for the IVC Intelligent Video Card).
6)MOVMODV - A version of MOVOLD for the IVC screen modified to reserve extra room at the top of RAM for a larger BIOS. (See SYS.DOC as supplied with SYSxxxx.MAC). Use MOVCPMN as the starting point if NASCOM screen output is required.
7)MOVCPM - Any version of MOVCPM as appropriate.

It should be noted that if the origional CBIOS is replaced as described later, then the resulting CP/M cannot be directly changed in size to suit a different Memory size. This is not really a problem since most people will probably be using 64K RAM anyway and will not want to change the size. Once the Software is set up it does not take long to reconfigure CP/M and insert the new CBIOS in any case. The method described here is to put the new BIOS into a copy of CP/M configured for the RAM size required and which is non relocatable.

The other way to do the job is to change the BIOS in MOVCPM.COM, the program that is used to alter CP/M to suit changing memory size. To do this requires Software that most people do not have available, as described below.

----------*****----------

The MOVCPMV/N.COM Program

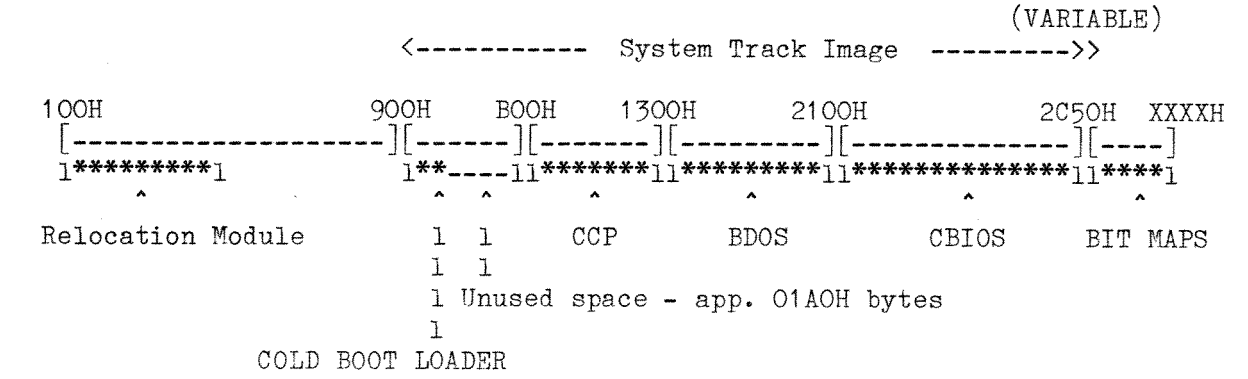


Figure 1. The MOVCPM Program and CP/M Image in RAM

The mapping of MOVCPM is shown in Fig. 1, which represents an 'image' of the program after it has been loaded into memory. It is important to note the following addresses. (Which could be different particularly on other systems.) Use your BUG program to look at your copy.

1)0100H - Start of MOVCPM Relocation Module
2)0900H - Start of CP/M System Track 'Image'. - 200H bytes that holds the COLD BOOT LOADER.(1 Sector = 512 bytes decimal.)
3)0B00H - Start of CCP. This is 800H bytes long (4 Sectors = 2K decimal.)
4)1300H - Start of BDOS. This is E00H bytes long. (7 Sectors=3.5K decimal.)
5)2100H - Start of BIOS. This length varies as described later, but cannot exceed 8 Sectors = 4K decimal on this Disk System.
6)2C50H - BIT MAPS. The start address is approximate and it will change depanding on the version of CP/M and length of the BIOS.
7)XXXXH - End of BIT MAPS. Again a variable address.

(According to CP/M UG UK the first 6 bytes of the BDOS are the Serial number of CP/M and should be retained intact for programs to run.)

The MOVCPM program contains a section to Relocate CP/M, and also a copy of CP/M 'organized' for low memory. The program also contains a module called a BIT MAP. This contains data which tells the relocation part which BYTES to alter when CP/M is 'moved' to suit a different memory size. If the BIOS is changed then the relevant part of the BIT MAP must also be changed. An assembler that can create a .PRL (Page Relocatable) File is needed to do the job, so this method is not available unless such an assembler is to hand.

To configure CP/M for a 64K memory system the command MOVMODV 64 * (CR) is entered. This would generate a 64K system for the IVC card with extra reserved BIOS space to accomodate 'SYS'. The CBIOS in CP/M would still be OLDBIOS. At this stage a copy of this 64K CP/M can be placed on to the System track by using SYSGEN, or it can be saved to disk as a named file - SAVE 43 CPM64.COM (CR). If you do this and then use a BUG program on the saved copy, it will look identical to MOVMODV.COM except that address high bytes will be changed. (It is interesting to try a CPM64 48 * command. It will try to work as if it were a MOVCPM but will probably give a SYNC ERROR.)

It is now possible to put a copy of this saved file onto a disk using the command SYSGEN CPM64.COM (CR). (Another very poorly documented feature of SYSGEN is its ability to read files off disk.) SYSGEN will read in the named file and will discard the first 800H bytes (100H-8FFH) which includes the relocation part of MOVMODV. Byte 900H will be put into the first byte of the first sector of the TRACK 0, SIDE 0 of the disk and so on up thro the CCP, BDOS and BIOS as shown in the Track Map. SYSGEN also permits one to READ the SYSTEM off one disk and WRITE it to another which is usually the most convenient way of moving the system from disk to disk.

The file CPM64.COM will be used later as the basis for the new CP/M with SYS CBIOS. It might be thought that at this stage it would only be necessary to replace the CBIOS in the file with the new one, and the job would be done. Unfortunately there is much more to do. First SYS must have all of its internal addresses correctly matched to suit a 64K System. As assembled SYS is a self relocating program with a base address of 3A00H and it is not amenable to assembly for a higher address due to the way in which it is configured. The solution to this problem is easy however. If CPM64.COM is put onto the System track and 'BOOTED' up, and then the required SYS is 'run up' in the usual way, there will automatically be a copy of the required SYS in RAM at OEE00H, with correct addresses and workspaces initialized. This relocated SYS can be used as the new BIOS, but first it is necessary to look more closely at some other problems.

----------******----------

SIMON is the ROM based program that loads the COLD BOOT LOADER that in turn loads CP/M. In my case SIMON is loaded to RAM at OF000H from a paged EPROM card. It can be overwritten when it's job is done, allowing use of full 64K RAM. (See Note at end.) On RESET, SIMON is loaded and one of the first jobs it does is to RESET the IVC card, after which it:-

1)Checks to see if there is a disk in Drive 'A'. If none is found, then SIMON outputs a 'NO DISK' message.
2)If a disk is found, it checks to see that the disk can be read correctly, and if not prints a 'READ ERROR' message.
3)If all is well, it reads in the first sector of Track 0 to high RAM and then copies some of it to RAM at 0000. This part is the COLD BOOT LOADER. SIMON then checks the validity of the disk by comparing the first two bytes read with an internal reference. If an error is detected the message 'WRONG DISK' is printed on the screen.

In the event of an error SIMON enters a MONITOR mode and commands can be executed to Tabulate, Alter Memory and so on. This is extremely useful even on 'good' disk loads because, after loading the system and any other program of interest, the disk can be removed and RESET pressed. The contents of memory can then be examined without the disturbing effect of the usual BUG program. (The area OF000H - OF400H and around OFC00H will be overwritten by SIMON)..

If no errors occur, SIMON transfers program control to 0002H, in the COLD BOOT. The first instruction is a LD HL, ADDR where ADDR is the start address of the CCP - D800H in the case under discussion. The BOOT now reads in more sectors from the disk, loading the data direct to D800H and so on, until the system is all loaded. Near the end of the BOOT is a JP ADDR1 instruction, where ADDR1 is the cold start address of the CBIOS. (OEE00H in this case).

(N.B. ADDR and ADDR1 are both relocation addresses, altered by MOVCPM, and they will vary with system size.)

----------*********----------

The System Track on the Disk.

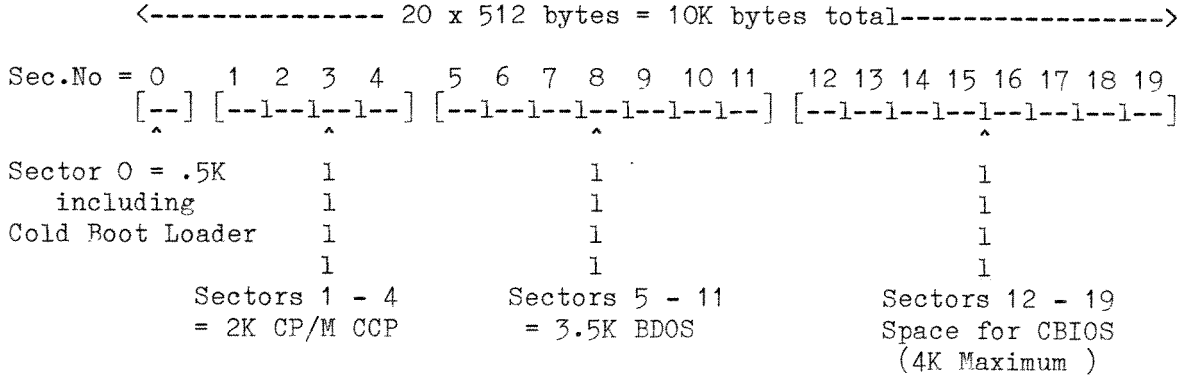


Figure 2. System Track Layout

Fig.2 shows diagramatically how CP/M relates to the Disk. The GEMINI DS/DD system uses Tracks of 10 sectors on each side of the disk. The tracks of side 0 and side 1 combine to make up one 20 sector track. Each sector holds 512 bytes (.5K) of data so the total capacity of one track is 10K bytes. There are 35 tracks, numbered from 0 to 34, so disk capacity is 350K. Track 0 is reserved for the CP/M system, and eight sectors (4K) of track 1 are reserved for the Directory. This leaves 336K for user programs. Fig. 2 shows that sector 0 of the system track holds the COLD BOOT LOADER. The CCP is in sectors 1-4, and the BDOS in sectors 5-11. This leaves a maximum of 8 sectors for the CBIOS.(4K)

A look at the RAM MAP in Fig. 3 shows that CBIOS space is available from OEE00H to OFFFFH, equal to 4.5K. When I assemble SYS the final size is around 4.5K including workspaces and buffers (and also includes nearly 400 bytes for code expansion). There is enough of room in RAM for SYS CBIOS but there is a shortfall of .5K on the disk.

The clue to the solution of this problem lies in the paragraph above. There is no real need to hold Buffer space on the Disk as it is only used when the system is running in RAM. There are 512 bytes for a sector byffer and 128 bytes for a directory buffer located near the start of the Disk Routines in the SYS Source code. If the System track is examined with a PEEK program, these Buffers appear as large blocks of 00 characters within the BIOS sectors. I moved these two Buffers by deleting their definition statements at the old position in the 'SYS' source code, and re-entering them after the CBIOS Workspace at the end of the Source Code. To make the addresses come right it was also necessary to delete three assembler directives - ".dephase" at the end of Read Section, just before the workspace, and the "org pbios+11a0h" and ".phase bios16+11a0h" directives just at the start of the workspace. The net result is that the operative SYS code and workspace only ammount to about 3.5K bytes. (The workspace should be held on

disk as it contains a certain ammount of initialized data). So there are now app. 400H bytes (1K) of RAM free above SYS. The two Buffers take 280H bytes so another 180H (384 decimal) bytes of code could be added without pushing the Buffers off the top end of RAM. I understand that a new SYS with support for the excellent MAP80 256K RAM card and a number of other extra new features might be available soon. I use one of the MAP RAM Cards as a Virtual Disk and it certainly speeds things up a bit but I miss the extra features of SYS in the support CBIOS.

The disk with CPM64.COM on its System track can now be put to use. Boot it up and then 'RUN UP' the modified SYS which will self locate over the OLDBIOS. (from OEE00H in this case). Then use a BUG program to load a saved copy of CPM64.COM into low memory. (e.g. ZSID CPM64.COM (CR).) Remembering the addresses noted earlier, copy the active 'SYS' from CP/M in high memory over the old CBIOS starting at 2100H. Since 4K is the maximum ammount that can be added, move 4K down to make it easy. (e.g. MEE00,FE00,2100 (CR).) will copy it down. Use the 'D' command to check at 2100H to see that data has changed and is the same as the data at EE00H. This new CP/M can now be saved. For tidyness though, I first filled from the end of the workspace to the highest possible end of the CBIOS with 00's using the 'F' command. This makes it easier to see the end of the code section when using a PEEK program.

The next step is to enter GO (CR) followed by SAVE 48 CPMSYS64.COM. Note the changes. The name has been changed to denote the fixed size and the embedded 'SYS' function. The ammount to be saved has increased from 43 to 48 Pages. The CBIOS started at 2100H and was 4K long (max) so the top address possible is 3100H. It is thus necessary to save memory from 100H to 3100H which is 48 - 256 byte pages. As previously noted SYSGEN will put memory from 900H upwards onto disk, but the area 100H - 8FFH must be preserved to be compatible with SYSGEN.

In theory if we put this file on the System Track of a disk and boot, the job should be done, since all addresses should be correct, and there is enough room on the disk. So a SYSGEN CPMSYS64.COM (CR) command is issued and the new system placed on the disk. Then RESET is pressed to try it. DISASTER !! - IT DOES NOT WORK.

At this stage, I tried a PEEK on the System Track. The BIOS all seemed to be there, with code and workspace area ending within a few bytes of the top of sector 19. That last sector, number 20 though - it should have been all 00's as the top of the BIOS was filled with 00's before saving it.But here I found E5's which is the pattern written to disk during formating so for some reason the whole of the CPMSYS64.COM file is not being written to disk. This last sector is not needed at the moment as all of the CBIOS was present on previous sectors, but I could want it with a new larger 'SYS', so why was it not written. To solve this one I disassembled SYSGEN. Luckily this is a short Program of about 1K, with a lot of ASCII text in it, so the task was not too bad. It did not take very long to find out that it was the sector counting routine that was responsible for the problem. It was Programmed to only put 19 Sectors to disk. The count only needed to be 'upped' by one to get all 20 sectors actioned. Locations 01BAH/01BBH in my copy were 3E 14H. (LD A,14H). I used my BUG program to set this to 15H, and saved the altered program as SYSGEN1.COM. The revised program now Reads and Writes the whole system track.

The SYSGEN mod. did not solve the problem though. I needed to see what was getting into RAM. I put the disk back into drive A and RESET. After a few seconds, I removed the disk, RESET again and used SIMON to explore memory. The bottom of RAM contained the COLD BOOT, unaltered, so CP/M had obviously not got going. If it had, the jump vectors and IOBYTE at the low end of the memory would have been visible. So I checked high RAM. It soon became apparent that the CBIOS was missing from FA00H up. This meant that for some reason two whole sectors had not been loaded from the system track to the BIOS RAM. Since the vital sector

relating to RAM FAOOH to FCOOH was on the disk, the only possible culprit was the COLD BOOT. This little program is only about 50H bytes long and was soon taken to pieces and once again a sector counter was found to be responsible. The sector count byte at address 0009 had been 11H (17 decimal) and SIMON had already loaded the first sector for a total of 18. To make the BOOT load the full 20 I set this byte to 13H using a PEEK program on the actual disk. This time, when booted in drive A, the system worked, and that is nearly the end of the story. The byte at 909H in the CPMSYS64.COM file was changed to 13H so that all copies of the BOOT LOADER on the System track would now load the full BIOS.

(A COLD BOOT modified in this way should NOT be used to load the origional GEMINI CP/M to a RAM address above D800H. Since the origional CBIOS is just over 2.5K long, the GEMINI CP/M would normally start higher in RAM, unless spcifically configured for say 63K. If loading were tried to the higher address using the modified BOOT, then the HL register pair, that hold the destination address of the bytes read in from the disk, would be incremented past OFFFFH, and would overflow back to OOOOH. The incoming data would load to low RAM, overwriting the BOOT, and soon cause a CRASH.)

To complete the job only one or two tasks remained. Richard Beal had included a little routine in the Load/Relocate module of SYS which he calls "cinit", which is there to remove any 'stray' input characters particularly when using Virtual Disk. I have successfully run CPMSYS64 without this routine, as SYS 11 does not support my MAP RAM card, and I do not use my G802 or any RAM B's as V.Disk. It might be necessary to include it with a later version of SYS that does support my MAP RAM though. The other task concerns 'Sign On' messages. The origional CBIOS messages have been wiped out by the new BIOS, and since the Load/Relocate Module of SYS is not now used, the comprehensive list of features that SYS displays on execution has also gone. It is nice to have an impressive list of system Software and Hardware features displayed on Cold Boot, to 'awe' the Plastic Box Brigade, and some form of Copyright mark should also be shown.

There are several places where it is possible to restore "signon" message routines, but "cinit" should be run by the COLD BOOT, or by SIMON after CP/M is in RAM. Since SIMON is overwritten by CP/M in larger size systems though, the latter course is not possible directly. There is currently enough room to put "signon" routines and messages into the BIOS area, but this would take up space needed for future expansions of SYS, so the BIOS is not the best choice.

There are app. 400 bytes free in the first disk sector that might be used for initialization routines, since the BOOT is only about 50H bytes long. There is another 'snag' though. SIMON loads the first disk sector into high memory, and then does a 'copy' to RAM at OOOOH before making a jump to location 0002H, to start the COLD BOOT. The problem is that only 80H bytes are copied from OFC2EH to OOOOH so any text messages would be largely left behind in high memory to be overwritten by CP/M as it is loaded. This means that to use the free space in the first sector, it is necessary to modify SIMON so that the whole of the sector is copied to low memory. The boot could then be altered so that it executes "signon" and any other initialization routines before jumping to start CP/M.

----------*********----------

If however, SIMON is going to be altered, then the EPROM programming effort might as well be made worth while. For convenience my present SIMON and the short program that copies SIMON to RAM and pages out the EPROM card are all in a 2716 ROM on the card. There is nearly 1K of space free in the 2716. I decided to use up this free space by adding to and by modifying SIMON. The 'Mods' so far are :-

1)I have included "signon" messages and also a 'Putvid' type of screen printing routine. These are copied to RAM at 09000H and 'Jumped to' by the COLD BOOT LOADER after CPMSYS64 has been loaded. If I need routines like "cinit" or UART initialization, they can be placed in this area of memory as well. After printing the 'signon' messages the routine jumps to BIOS to start CP/M. Any CP/M that contains OLDBIOS, and does not have a modified COLD BOOT will skip these messages but will display its' own messages.
2)I have modified the 'TAB' routine so that 16 locations are displayed on a line, in HEX and ASCII. The HEX is displayed in columns of four bytes. This mod. is really more useful with the IVC card display as there is a lot of 'wrap around' on the NASCOM 48 wide screen, but it can be used.
3)I have also added a few extra bytes so that I can now 'Dump' the 'TAB' output to my IMP Printer.
4)The 'SIMON' signon messages have been increased to show that a modified SIMON is in use. A list of Command keys is also displayed as a reminder of the commands available in SIMON.

It may be necessary in the future to add a 'cinit' routine, and it would also be possible to include any special initialization routines for UARTS etc. By reducing the very extensive 'signon' messages, or using an even larger ROM (!) there is scope for almost any addition, such as direct Memory dump to cassette/disk, and so on.

N.B. If the initialization routines in middle memory make any calls to CP/M to print messages, or to clear characters as "cinit" does, then since CP/M has at that stage NOT been started, it will be necessary to initialize the ' BDOS jump ' and IOBYTE in low memory BEFORE making any calls to BDOS otherwise the system will 'die'.

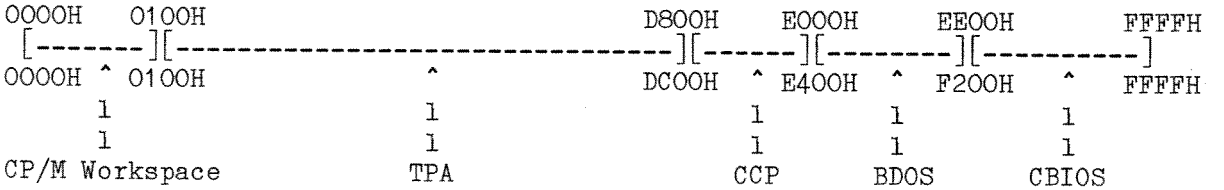
I hope that this article has been of some help to others who, like me want to learn more about some of the less well documented features and operations of CP/M systems. (I must apologise however, to the Authors of the various Programs for the liberties that I, an amateur 'hacker', have taken with their Software, and hope that this will be accepted in the cause of 'Learning'.)

NOTE :- My method of Booting NASSYS and CP/M from EPROM CARD is described in 8OBUS NEWS No 1. (N.B. There is an error in the diagram on page 45 of that article. The wiper of S1A goes to pin 2 of LKS1, not to pin 3.)

----------**********----------

Appendix 1. The Mapping of CP/M in Memory.

Above the line, addresses refer to CP/M from MOVMODV



Below the line, addresses refer to CP/M from MOVOLD

Figure 3. CP/M Memory Addresses for two different CBIOS's, assuming 64K RAM.

The manuals and guides to CP/M should be consulted for more details.

CP/M uses RAM below 100H as a Workspace and Buffer area. Above the workspace, starting at 100H is the TRANSIENT PROGRAM AREA (TPA). This is where all programs other than CP/M reside and operate. The TPA extends up through memory, including the CCP area of CP/M. In the systems drawn, the TPA will run from 100H to E000H/E400H, which is around 56K or 57K bytes.

The CONSOL COMMAND PROCESSOR (CCP) is the area of CP/M that interacts with the user. It supports commands like DIR, REN, TYPE, ERA, etc. and allows selection of other drives, loading of files etc; When programs are loaded and run, the CCP is not needed, so the 2K of space that it occupies can be used by programs.

Above the CCP is the BASIC DISK OPERATING SYSTEM (BDOS). The user accesses the BDOS through the jump at 0005H, to use the 36 or so Functions that are available such as Keyboard input, Screen output. Disk read, and so on. The BDOS is 3.5K long and cannot be overwritten.

The final unit, at the very top of memory is the BASIC INPUT OUTPUT SYSTEM (BIOS). Often called CBIOS, the C standing for CUSTOMIZED. i.e. altered to suit.). This part of CP/M is not supplied by DIGITAL RESEARCH, but by the suppliers of the Computer. The job of the BIOS is to marry together the standard CCP/BDOS to the thousand and one different Hardwares that have to run CP/M. The BIOS will therefore vary from computer to computer. The quality of the BIOS varies tremendously from system to system, and CP/M can get a bad name because the BIOS is poor. The various BIOS's available to run on the NASCOM/GEMINI implementation contain features that are not always common. In some cases they are almost unique. RICHARD BEAL, the author of NASSYS seems to have made a speciality of writing a very powerful CBIOS for NASCOM/GEMINI systems that he calls 'SYS', and this article describes how to replace the origional CBIOS with 'SYS' CBIOS.

----------*********----------

David Parkinson adds..
==========

1) If you are overwriting the standard BIOS by one of your own you may find that insufficient space is available for it. The obvious solution to this problem is to generate a slightly smaller CP/M system to give yourself more room (eg MOVCPM 63 * in a 64k system). However this results in the loss of 1k of memory from the TPA, which seems quite excessive if you only want an extra 100 bytes. By changing one byte in the MOVCPM.COM file you can move the memory image of CP/M down (or up!) in increments of 256 bytes. The byte is at 23Dh and holds the number of 256-byte pages that MOVCPM.COM adds on to the system it generates to allow for workspace area for the BIOS. Increasing the current value of this byte by one will give you an additional 256 bytes of memory, and pro rata.

Remember that the extra memory does not appear out of fresh air, CP/M just moves down a bit to allow a little more room for the BIOS.

2) As Mr. Bowden points out, there is little point in saving a large workspace/buffer area on the system track of the disk, and so all workspace should be placed at the end of the BIOS so that all the code, (which HAS to be saved on the disk), comes first. However an additional saving can be made in the BIOS memory requirements by realising that certain sections of the code are used once only on start-up. A prime example of this is the sign-on message, which only appears when the system starts up from cold, and then is never seen again. All the Gemini BIOS's have the sign-on message stored in the area of memory that the BIOS uses for workspace, as it will be printed before the BIOS actually uses any of that area as workspace.

With the current BIOSs for the Gemini Galaxy and Multiboard systems the idea has been extended by including all the initialisation code that is only executed once. This covers items such as PIO initialisation, UART initialisation, printing the sign-on message, setting up the parameters of the "Memory" drive, setting up the Winchester disk controller, ...and so on. This results in a saving of 300+ bytes in the BIOS memory requirements.

[Do this by:

```
                ... Bios code...
                RET                     ; End of BIOS code
;
WSPACE:
DRIVE:  DEFS    1                       ; Current drive
        ....
        DEFS    512                     ; Sector buffer
        etc
; end-of-workspace
        ORG     WSPACE                  ; Reset assembler origin
COLD:   LD      SP,STACK
        ..init code...
SIGNON: DEFB    '...
        END
```

]

3) SIMON and the boot sector. As pointed out by Mr Bowden the SIMON for the Nascom loads the boot sector in its entirity to high memory and then copies the first 128 bytes down to 0. The SIMON for Galaxy/Multiboard reads in the boot sector directly to 0, but does not load any data after the first 128 bytes, (the remaining bytes are discarded without being stored). In each case this was done deliberately, to ensure that the TPA was left unaltered by the actions of SIMON.

This means that when your next door neighbour turns on his arc welder and dims the lights of the neighbourhood (crashing your editor) just after you've spent four hours typing in a program, you can press reset with some degree of hope. Once CP/M has been reloaded you can then type "SAVE 250 JUNK" (or some such large number). This will save the entire contents of memory to a disk file "JUNK". GEMDEBUG can then be used to reload it, and hopefully you can find your source somewhere in what was the memory buffer of the editor. Move it down to 100h, add a few 1Ahs on the end, exit and then do another "SAVE" and you should have your source file.

In other words I arranged it so that if you are ever forced to press "Reset" you can get a disk copy of the TPA (as it was when you pressed reset), and hopefully recover something from it, either a data file, or information of what might have lead to the crash. So there was no 'malice aforethought' in ignoring the extra 384 bytes!

(Note GEMDEBUG will load as much of JUNK in as it can. ZSID on the other hand will give an 'out-of-memory' message and return to CP/M. In the later case you can try using the "S" and "Q" commands of PIP to split the file up into smaller segments).

The MAP 80 RAM board.

It was getting to be about time someone brought out a new 80-BUS board that would astound the plastic box brigade, and suddenly there are two around, both containing massive amounts of memory. The MAP 80 board is first on the agenda, simply to make you wonder what the other one is until you get to the appropriate page, although I have put a really subtle clue in the heading. You have seen the advertisements, so you know that the MAP 80 board can hold up to 256K of memory. You may even have seen a hardware review somewhere, that went on about the sheer amazing blueness of the board. So I am going to write about how I went about putting mine together, and the software that is supplied for it. Mind you it is very, very blue...

First of all, being the adventurous type, I ordered the bare printed circuit board, which costs £42-50. The day after I sent the order, I was telephoned by MAP 80, who wanted to make sure that I was not trying to use the board with a Nascom 1. Apparently, this is a combination that does not work! So, I reassured them that I was now using a Nascom 2, and the board arrived very soon after. The board is of a high standard, nice solid glass fibre, with the blue solder resist, and the usual silk screening to show where things should be put. Things are pretty closely packed, as you would expect, since there are fifty four DIL sockets to fit on the board. With the board were the parts list and construction instructions. I ordered all the sockets, resistors, capacitors and most of the TTL chips from Maplin, who managed their usual rapid response. A couple of the chips needed are not mentioned in the Maplin catalogue, but can be obtained from other firms, such as Watford Electronics, or Technomatic. When I unpacked the sockets, I was surprised to find that they were made from a rather nice shade of pale blue plastic, instead of the usual black. When these had been soldered to the blue pcb, the result looked like something from another dimension, after years of looking at green boards with black sockets! Cost of components (excluding PCB and RAMs) was under £20, just! The next step, which I am still saving up for, is to buy the thirty two RAM chips the board can carry. As these are about £4-50 each, the 256K board can be built for just over £200, which is a lot cheaper than people with S100 based computers seem to pay for their boards. The chips have now begun the gradual slide down to a more sensible price, so it is nice to know that the board will work with just a single row to start with, as long as you don't connect up the chip select lines for rows that are not there.

There may well be murmurs of "What on earth can you use it for?" in the ranks. At first, I used to find a 1K Nascom 1 big enough. Now my programs are much bigger, but there is as yet little probability that I will overflow the 64K mark! No, the magic phrase that made me so keen to get the board was "virtual disk". [Ed. - see R. Beal's article elsewhere on VDs (oops!).] And you don't have to mess around modifying CP/M, because MAP 80 will do it for you. They have done this for me, and the disk has been returned with two new versions of MOVCPM on it, one for the usual display and one for use if you have an IVC card on your system. The source code for the revised BIOS is supplied, which is a very sensible approach, enabling even further customisation of the software without any need to guess what their bit is doing. [Ed. - I wonder how Gemini feel about other manufacturers modifying their CP/M BIOSs and sending out source listings! Naughty!] No doubt, sooner or later, I will have time to have a look how it works! The CP/M works in the usual way when there is 64K or less of memory connected, but when there is more the sign on message will announce that the virtual disk is available, and give its size, according to the documentation. Must order some RAM chips... As a bonus, my "old" 64K, GM802, card is not made redundant by all this. The instructions supplied with the MAP 256 tell you how to fix it so that the software will use it as well. Presumably, the effect of the modifications is, in part, to change the GM802's paging control port to £FE, from

the usual £FF.This would explain why the new BIOS does not interfere with extension memory boards on pages other than page zero, in the Gemini/RAM B paging system. Anyway, when I get the chips together and get it all going, I will have a nice tidy 64K system with an equally tidy 256K virtual disk. When that is done, I intend to produce a couple of file updating programs in Pascal, so that I can produce some sort of "bench-mark", and give you some indication of what sort of difference these changes make to the speed of operation of the system.

"Pluto review", he said casually.

I don't know where the money for all this stuff is going to come from, but I have bought it anyway! Unlike MAP 80, IO Research took quite a while to send my Pluto out, but then they do say that the board is selling like hot cakes. It is even being sold to the makers of non-80-Bus computers, for use as a graphics unit on their bizarre (i.e. non 80-Bus) computers. I did try asking for an OEM discount, but it did no good at all. I bet you thought all the hardware manufacturers would be sending out free boards to likely reviewers, in order to get favourable reviews printed. Well, it may be like that in glossy magazine land, but there ain't no free lunch out here in amateur land, yet! (Of course it is a hint, Mr Manufacturer, of course it is.)

Anyway. The board is of a high standard of construction, and all those things I would write if I thought you wanted to be told. Come on now chaps! Anything that costs this much is going to be of a high standard, or you would have heard rumours to the contrary. What you get is eight inches square, green, and packed with components. There are three rows of RAM chips, making 192K, an 8088 processor, TTL chips galore, and something with 40 pins whose function is unknown to me. The latter is probably some sort of video controller chip. [Ed. - a 6845 CRTC chip, as used on the Gemini IVC and Nascom AVC.] Then there are two connection plugs at the outer edge, one with twenty pins, for connection to a colour monitor, the other with fifty pins (I think) which is for connection to the Pluto Palette, when it appears. Also supplied is a manual. This is somewhat terse in its explanations of quite complicated matters. The pages are not all in the sequence you expect, there being appendices in the middle of the text. There are two example programs, one in BASIC, one in assembler, but they don't do a lot. The most useful thing in the whole slim volume is a table of all the control codes the board accepts, listing all the parameters they need, and what values they return. Sounds like Pascal, doesn't it? A nice touch in the manual is the way the routines are described, with headings that look like Pascal procedure and function headings. This has made me start thinking about producing some sort of software package to interface Hisoft Pascal to the Pluto, along the lines of a little known utility called Vortex, that I cobbled together a while ago. This time I won't have to write a line drawing routine, that's for sure! The manual also tells you how to connect the board up to a video monitor, as long as you know which connection is which on the monitor! No, it isn't always obvious, that would make life too simple. Just to make life more entertaining, I managed to wire mine up wrong, but failed to destroy my nice colour monitor. See later section, for a very relieved review...

As far as your computer is concerned, the Pluto is just two ports (similar to the Gemini IVC), normally £A0 and £A1, although these addresses can be changed if you have to. Port £A0 is the status port - if bit 7 is set when the port is read, then Pluto is ready to receive a command. Writing anything at all to this port resets Pluto. Port £A1 is the data port, through which you send all the commands and data needed to produce the pretty pictures, and read data sent back by Pluto in response to your requests. It might at first seem necessary to check the status before you send each byte, but this is not so. Once the command byte has been accepted, and the status port says Pluto is ready, data can be sent as fast as a 4MHz Z80 is able to do it. This includes the use of the amazing Z80 block output instructions.

And what sort of instructions are there? Well, lots. For a start, there are two screens of 640 by 288 dots, and any dot can be individually set to any of the colours the board produces. So there is a routine to plot a point, given its coordinates; there's another one to plot a point at a given x-y displacement from the current position. This latter routine has two forms, one for distances that can be expressed in a single byte, and one for longer distances. Then there's the line drawing routines, also in three forms. There are routines to move to a given point, or by a given displacement. It is possible to read the colour of a pixel, again using absolute or relative coordinates. After all that, you could be forgiven for thinking I have finished, but not a bit of it! Rectangles can be filled, or even copied to another location. Part of one screen can even be copied onto the other screen.

And each of the operations described above is influenced by two system variables called WPROT and STYLE. WPROT is used to write protect one or more colour planes, while STYLE determines whether the new information replaces the old, or is combined with it in several other possible ways. For instance, the colour being printed can be exclusive OR'ed with the existing colour of a pixel. This sort of thing can be used to produce red/green flashing text on a green/red flashing background, if that is what turns you on. Text? But of course! The standard ASCII character set is catered for, and an 80 column display can be produced. It must be possible to amend CP/M to use this board as its display device, somewhat like an amazing IVC, but I may not even try. [Ed. - I know of someone who did this some time ago, and commented on its amazing slowness due to all the characters being held as dots, and the consequent amount of memory shifting that has to be done when the screen is scrolled. The AVC suffers similarly (but even worse I believe). He also commented that backspace was not supported, and so 'cursor left, space, cursor left' has to be issued by the CP/M BIOS. It's an obvious case of horses for courses - text cards for text, and colour graphics cards for ....] You see, there is a marvellous moment when the ZX81 owner first notices that your computer has two screens. More to the point, really, is the fact that I have a lot of software that uses the old Nascom screen, or rather, its relocated equivalent. I don't really think I want to edit these articles in colour, unless the magazine is going to change over to using a colour plotter instead of the daisy wheel, and start looking like OZ...

As part of the Taunton Computer Club's latest theme program writing game, I also wrote a thing that draws a picture of a winter scene. In this program, I made use of the Pluto facility for defining new characters, of any kind and colour whatsoever; these are referred to as symbols. Once a symbol has been defined, it can be printed at whatever location is current, just by sending a single byte out through the data port. This function is impressively fast, as is the other very useful one I left out above. Referred to as "polyfill", this fills complex shapes in with colour. I saw someone using their BBC computer the other day, and their polyfill routine (in BASIC, naturally) took ages, as well as not always getting into all the corners. Having this sort of facility readily available will make the rapid development of programs much more easy. I should be able to do a really nice space invaders in no time...

It is a tremendous shame that such a superb lump of hardware is being let down by the way IO run their business. They take a long time to send things out, which can be forgiven when a firm is building up its business, and may be running with less staff than it will have when things level off abit. But they could easily send out a card to say they have had your order. They don't even bother to answer all the letters they are sent, and this is unforgivable, when they are being asked technical questions about the product they have sold you. The only way to get answers to this sort of question seems to be to ring them up. Their phone is more or less permanently engaged, of course, presumably by people who want to know why their letters are ignored! I wanted to write about the additional ROM, which contains even more amazing facilities like the marvellous "wallpaper" routine, but after three weeks I am still waiting. Still, it will be something to write about next time.

Change of address.

For the benefit of any of my fans who are still reading this (Sid and Doris Bonkers only - the other two have gone out to rob a bank, so that they can buy Pluto boards), here is my new address:-

Chris Blackmore,
27 Laburnum Street,
TAUNTON,
Somerset,
TA1 1LB.

And now it is software review time!

In line with my current policy of megalomania, I propose to review 96K of programs in one go. The programs are all by Level 9 Computing, of High Wycombe. As you may have noticed, 96K is three times 32K, and I have been playing three adventure games, each of nominally 32K size. These run on Nascoms with Nas-Sys and at least 32K of memory, but they also work on my system, under the control of my fairly famous MONITOR.COM program. This is because sensible software authors always use the standard Nas-Sys way of calling the necessary monitor routines, rather than accessing the screen memory directly. In fairness, I should point out that the Syrtis adventure also works perfectly on systems with MONITOR.COM instead of Nas-Sys, which I forgot to mention in my review of their program. Sorry!

The programs are called "Colossal Adventure", "Adventure Quest" and "Dungeon Adventure". They are supplied on very good tape, which loads with no bother, apart from the volume being so high that it endangered the VU meter needles on my tape machine! Documentation is provided with each game in the form of a small book. Also included is a stamped addressed envelope which you can use to request a free hint. This is a fine idea, although they warn that if you ask for too much the replies may be somewhat cryptic.

The first of the three games is a fairly standard implementation of the original cave adventure, in which you have to fetch all the treasure out of Colossal Cave, and bring it back to the hut in the woods. This is almost the same in its topography as the Syrtis version, with the exception that a few objects are in different places, and there is a picnic area in the woods that can be surprisingly dangerous, if you behave anti-socially! The program does not give as full a description of some places as the Syrtis one, and does not ask you if you want hints. The authors say that this gives them room for an extra 70 locations in the cave, which they refer to as the "end-game", and it could well be so, if only I could find them! I still don't know how to procede beyond the infuriating "Plover Room" without falling down a pit. When I eventually resorted to the practice of copying the program into the screen memory, to see what words it would let me use, I found that about 2K at the end consisted of the "a-code" source from which the program was compiled. There are more spelling errors in this program than in the Syrtis one, although they may well have been corrected in later versions.

"Adventure Quest" starts out in the same forest as the previous program, and you can even find the grate that used to lead to Colossal Cave, but you can no longer get down there. The idea this time is to find a nasty character called Agaliarept (etymology obscure, as they say in dictionaries!) and do him in. Of course, it just isn't that simple; you can't even begin to think about attacking him without doing a lot of exploring, and finding the necessary weapons. I am still getting a lot of fun out of this program, and seem to have found about a third of the locations there are supposed to be. I have had considerable problems with a model lung-fish that lurks in one location, and think I have found an original way to cheat the giant sand worms that trundle around in the desert. So far, I am still getting rotten scores, and am nowhere near beating Agawhatsit. I have even resorted to drawing maps, which purists amongst the adventuring fraternity would no doubt frown on, unless the program said that they were carrying a pencil and paper!

In the third game, it is assumed that you have defeated the bad guy of "Adventure Quest", and have decided to go and liberate all his treasure. Your plans go badly wrong when all your weapons and equipment are stolen, and the game starts with you waking on a mud bank by a river. So far, this seems a very difficult game to crack, as almost everything seems able to kill you, while you have no weapons to do likewise to them. On the other hand, this could well be just the result of my lack of experience with this game, and it is true that the satisfaction of cracking these things is directly proportional to the difficulty experienced on the way. Perhaps when I find a way to defeat the "hideous yellow bird" I will do better? The program accepts the word "Buzby" without question, which makes me wonder what they have against him...

This sort of thing is of course very much a matter of personal taste: I read all the hardware oriented science fiction I have time for, and never bother with "sword and sorcery" books at all. These programs are a taste that I seem to have acquired in spite of myself, much to my own surprise. It is fortunate that you can save the status quo on tape for later reloading, or I would get even less sleep. Have a go with one, and if you hate it, never mind. If you find it starts to get to you, then my recommendation is to get the Syrtis adventure and the second and third of the two Level 9 adventures. That will keep you off the streets for ages. As I have failed to mention, they are very reasonably priced, especially when you consider just how long they will keep you busy. They are definitely not the thing for people who like to hammer the space bar until all the funny shaped blobs have been destroyed...

The program library is dead! Long live...

Well, I have ordered a copy of the source code of the famous Lollypop Lady Trainer, in the hope that it will help me to write a fun program for the Pluto. Lots of red, when the cars hit the dear little kiddies...

The reason why the library is no more, we were told, was that all that paper took up a lot of space. That makes sense. The computer revolution is supposed to do away with all this tree murder, after all. So I put on my thinking cap, and I think I have thought of something to put in the place of the old paper library. See what you think, bearing in mind that the proposals are in rather embryonic form, and that they are addressed mainly to disk users. There is no reason at all why tape users should not do a similar thing, of course.

"The Circle of Iron", or some such corny name, would consist of a group of users posting a disk around in a circle, adding their programs to it, and copying any of the ones already on it that they wanted. Let us be clear from the start, these will be your programs, not borrowed ones! When a disk arrives at your abode, what do you do? Well, first you wipe out anything you put on it the last time you saw it. Then you look at what is new, and help yourself to any you like. Put your new files on the disk, and put it back in the post. Much easier than all that bother with paper, I am sure you will agree.

So now we get a bit technical. There are a variety of disk systems, to understate things rather. There will need to be a separate circle for each of them. This will mean one for each of the following:-

(a) Gemini single density - Henelec FDC + 48TPI drives (e.g. GM805)
(b) Gemini double density - GM809/GM829 + 48TPI drives (e.g. GM815)
(c) Gemini quad density - GM809/GM829 + 96TPI drives (e.g. GM825/Galaxy/Quantum)
(d) Nascom disk systems (yes, they do read this!)
(e) Whatever I have forgotten....

Now you will have to stir yourselves. If you want to join in the fun, I want you to write to me, saying what sort of system you have. I am NOT going to organise the whole thing, but am willing to do the donkey work of setting up the Gemini double density loop. So if you use some other system, and are willing to start up their loop, let me know. Now, as everyone knows, apathy will do its evil work, and none of you will write. Prove me wrong, please!

Another thought has just struck! (Two in one day!) Anyone who knows how to transfer programs between Nascom disk systems and Gemini disk systems using tape will have spotted this. A program that works on one CP/M system ought to have been written so that it will work on another CP/M system. So perhaps there will be a figure eight amongst the circles. Anyway, disk users, please write to me: tape users, if you want a similar system, let me know. The first tape user to write gets to organise the fun on behalf of the others. If this idea can be made to work, we'll be light years ahead of the plastic box brigade (no, this is not a reference to the Nascom 3, that is a nice box!) as usual.

Back to hardware! The Microvitec 14" monitor.

You need something like this if you are going to use a Pluto. The cost of a monitor capable of displaying the full 640 dots across the screen is phenomenal, and if you are not rich, you will just have to compromise a bit. Microvitec do make a high resolution monitor that can cope, but I could not get the necessary finance, so I bought their 14 inch RGB nearly as high resolution monitor. This is the one you see on the BBC's Computer Program, in a nice steel case. The display quality is excellent, but is not quite enough for 80 columns of text in colour. Please note that the text CAN be read easily enough, it is just a fraction this side of perfect, not all furry the way it would be on the average television, after going through a modulator and a tuner. The monitor is built in Bradford, but the tube was made in a country where it is said that they can't write software like ours. (Come to that, thay make some rather Mickey Mouse computers, as well.) Microvitec have recently won one of the awards to industry, according to my Sunday paper, and deserve it. This unit is excellent value. It is available from many dealers, and the price varies, so you should shop around. The catch, and there has to be one, is that if you buy cheap, the dealer expects to be able to get away with giving almost no after sales service. I got mine from a firm called Microage, who are nothing to do with Microvitec, I hasten to add, and when I wrote and asked them a question, they sent my letter back with the words "Sorry can't help" scrawled on the bottom of the page. Not good enough, Microage.

Anyway, when you write to ask Microvitec how to make adjustments, so that as few as possible of your expensive pixels are off the edge of the screen, not only do they send the required information, they also send a letter reminding you just how dangerous it is to open the box. I would echo their warning - it is extremely dangerous to open the set up and adjust the height and width of the picture. The voltages used by colour tubes can do some very spectacular things to you, from a distance, and the tube can hold a charge for a long time after it is switched off. If you are only fairly sure you know what you are doing, leave it alone! The height is easy to get at, but the width adjustment is under the tube.

Sneaky advertisement.

You can buy a historic computer for a very reasonable price, as a reader of this magazine. I would like to sell the original Marvin to an enthusiast (you'll need to be!) for a low type price. Nascom 1, buffer board, 32K RAM A modified to run properly, and MNUG (Merseyside Nascom User's Group) EPROM board with Nascom BASIC, Bits & P.C.s toolkit and extension keyboard, complete with 3 amp PSU, and various bits of documentation for only £150. Nascom I.O. board with one PIO and a CTC for a mere £50. Winchester Technology sound board (likely to become exceedingly rare!) with amazing Doorbell chip, only £60, because it won't go at 4MHz without wait states.

finished := TRUE

Book Reviews

by R. O'Farrell

With the permission of the Editor, I would like to apologise to the enraged readers of the 80-Bus News for the absence of any book reviews in 80-Bus No. 3. This was due in part to other demands on my time (my surgeon won't allow me bring my Computer into hospital with me - I think it makes him feel insecure) and also to the fact that I found very few books worth buying and reading. It is not generally known, but most, if not all, of the reviews published in 80-Bus News are based on items purchased by the reviewer using real money. The idea that we sit around all day unwrapping parcels of items sent to us free, gratis, and for nothing for review is a gross misconception in general. If some kind manufacturer wishes to send me a dual floppy disk set up for review and extended test, I will be happy to facilitate him, as my car is currently running around on my dual floppy disk funds, they having been requisitioned to rebuild its automatic transmission. Did you know that early models of the Volvo 343 suffer from a design fault in the bobweights, resulting in catastrophic failure of the transmision every 10 - 20 thousand miles? I found this out the hard way! I'm glad to say that, having spared no expense, the transmission is now rebuilt, using the new improved parts, and should be good for many hundreds of thousands of miles. Anyone like to buy a slightly used Volvo 343?...

Enough of the bulletin from the battlefront. Bring on the books! As I said, of recent months there has been little published that interested me. This may be due in part to the fact that Dublin is very nearly a computer desert. One book I found was:

Microcomputer Technology by Prof. Julian R Ullman,

published by Pitman at about £5

This book is an introductory survey to microcomputers and their use. Its specific chip is the Z80, and it deals with the programming of the Z80, the types of data structures one meets in computing, the use of high level languages, and a fair whack of logic design. The most interesting section is the last chapter, where he uses Pascal to show how a Z80 assembler might be written.

Microcomputer-based Design by J.B.Peatman,

published by McGraw Hill (student edition about £7.50)

This is in many ways a similar book to the foregoing. Published in 1977, its emphasis is a little oriented towards the 8080 and other older machines. Nevertheless, it does have appendices on the Z80 and 6809. It also gives a fairly good overview of the problems of interfacing from chip to chip, and suggested circuits for voltage level shifting.

An Introduction to Database Systems by C.J.Date,

published Addison Wesley

Is a substantial work reviewing the differing techniques used in data base programming. I've read it once, and have put it aside for a rest before reading it again, to give its contents time to be assimilated. I'm interested in the problems of data base management as I intend to write a simple Database Management System for a particular application.

The Art Of Computer Programming

Vol 1: Fundamental Algorithms
Vol 2: Seminumerical Algorithms
Vol 3: Searching and Sorting

by D. Knuth, published Addison Wesley

These books form the Bible of computer studies. There is a rule - "If it's in Knuth then its right". They are not light reading. Usually about 20 minutes is all you can take, and the rest of the day is spent assimilating what you have

read. They are also not cheap, costing about £20 each, although there is a paperback of Vol 1, and possibly also of the others. I'm in the process of reading them, and mention them here to draw your attention to them.

If you are into Statistics, you might wish to look out for:

Basic Statistical Computing by Cooke, Craven and Clarke,
published by Arnold (circa £6.50)
This is a book to show how microcomputers can be used in the analysis of statistical data. It gives fully documented listings of many statistical procedures in BASIC (the shame of it!) and claims that they have been proved on four common micros. If you are into statistical analysis, this might well save you a lot of work.

Without wishing to cast aspertions on Knuth's master work, I have kept the good wine until last.

The Mythical Man Month by F.P.Brooks,
publ. Addison Wesley, costs circa £7.
This is a series of 15 short humourous essays on various aspects of writing large complex computer programs. It is based on Brooks experience as director of the team which wrote the operating system for the IBM 360. I first heard of this book some five years ago, but only recently came across a copy. It has been reprinted earlier this year (by popular demand, I think). Brooks deals lightly and humourously with his subjects, but gets his point across - possibly the better for the light touch. If you enjoyed reading Browns 'Interactive Compliers and Interpreters' and 'Pascal from Basic', then I'm sure you will like this, more particularly if you are involved with communication and management. It must rate alongside Kernighan and Plauger's 'Software Tools' as one of the seminal books.

So much for the rave review. As I've said before, many of these books are textbooks and not ther lightest reading in the world. Don't rely only on my reaction to them. Look for them in your local technical bookshop and browse through them before you purchase.

I realise that the foregoing reviews offer very little for the beginner. I've been asked to suggest a good starting book on Assembly Language. Looking through my bookshelves, I keep coming back to "The Z80 Microcomputer Handbook" by Barden, published by Sams (distr. Prentice Hall), cost about £7. This was one of the first books on programming the Z80. It taught me much of what I know on the subject. It is readable, and accurate, which cannot be said of all Z80 books! This is the book I recommend whenever I am asked for something on assembly language for Z80 users.

To show that I am not utterly involved in computers to the exclusion of any trace of the humanities (as in Arts and Humanities), I have also been reading (inter alia):

The Sources for the Early History of Ireland - Ecclesiastical. by Kenney, published by Columbia University Press 1929, and recently reprinted (c. £20)
This is very nearly to the early history of Ireland as Knuth is to computing science. It surveys the extant literary sources, giving useful synopses and references to previous publications on the subject, so that you may follow historical lines of enquiry quite easily.

RP/M --- MAP 256K RAM --- SYS --- VIRTUAL DISK

by RICHARD BEAL

This issue of 80-BUS NEWS is the first for which I have written an article, so I would like to start by congratulating the editors on keeping up the best traditions of INMC NEWS and INMC80 NEWS, by publishing my contributions! In fact not a word has appeared from me for a year, but I haven't been completely out of action during that time, as you will see.

There are four related articles in this issue, and it might help if I explain how they are tied together. First is the article on RP/M. This is the ROM operating system for Gemini computers, which has been updated. The article explains all the changes that have taken place, and how RP/M has been altered to keep up to date. The latest changes, bringing RP/M to version 2.1, have been triggered off by the introduction of the MAP 256K RAM card.

The MAP RAM is the subject of the second article, which reviews this exciting new 80-BUS card and explains how to program it.

The third article announces the latest version of SYS (15.0), which has grown considerably over the last year, and now supports almost every 80-BUS configuration you can imagine. How about a Nascom 2 fitted with a Gemini Winchester hard disk, or, perhaps Micropolis double sided floppy disks (or Pertec double sided), a standard eight inch floppy, a Gemini video card, and a megabyte of MAP RAM! With SYS it's easy, provided a Gemini GM829 FDC controller is in use.

The fourth article, about Virtual Disks, brings together the hardware of the MAP RAM with the software of SYS. Are Virtual Disks useful or just a desperate attempt to find a use for too much RAM?

The fifth article (that surprised you didn't it) is a review of the E.V. Beeper. It has nothing to do with the others, except that I wrote it. Happy New Year.

LATEST NEWS OF RP/M

by RICHARD BEAL

I noticed a strange thing when I was looking through old issues of the 80-BUS NEWS and INMC 80. Nobody has ever written anything that I can see about RP/M. There must be lots of people with RP/M in their systems, because most Gemini GM811 and GM813 computers have been supplied with it, except for those packaged with disks such as the Gemini and Quantum computers, which have special boot programs (called SIMON). One reason for this may be that many people have disk systems nowadays, so they have CP/M and are not interested in RP/M. However perhaps they give it a passing thought as it boots up their disk for them!

In case you don't know what RP/M is, let me remind you. RP/M means ROM Program for Microcomputers. It is a 4K ROM which provides simple monitor program facilities rather like those in NAS-SYS. However the main feature of its design is that it provides a programming interface almost identical to that provided by CP/M. This means that all software developed under RP/M will run under CP/M, allowing easy migration to disks later. Also, some CP/M software, such as the Microsoft BASIC interpreter, will run under RP/M. It simply has to be loaded from tape (or from EPROM).

It is also possible to develop and test software under CP/M which is intended for use on a ROM based system using RP/M. This could be most useful where a dedicated control system is being built. For example a Gemini GM811 board could be used by itself, without video card or disks, to control equipment attached to its PIO. The sockets on the GM811 would contain RP/M, a special EPROM designed to control the system, and RAM. Also a terminal could be attached to the

serial port. This could make a very economical solution to many engineering problems. It would be interesting to know what applications RP/M is being used for - why not write an article about yours?

Now to bring you up to date on RP/M. The first release was Version 0.1, which was issued with all GM811 boards until recently. This version was quite successful as a first attempt, since it had no reported bugs! However there were a number of areas for improvement to the design, and when the GM813 was produced and the Micropolis eighty track drives appeared, a new version was required. Version 2.0 was issued in June 1982, and like the original still has no reported bugs. The description of Version 2.0 below, which describes the changes made, may convince you to try to persuade a Gemini dealer to sell you one.

The introduction of add-on memory mapped RAM cards introduced a small problem since if you put together a system comprising a Gemini GM811 CPU card, RP/M Version 2.0 and the MAP 256K RAM card then unfortunately it doesn't work at all. Any other combination is all right, so if the CPU is a GM813, or if the system is a GM811 with RP/M V0.1 (which is more likely anyway), then there is no problem. The reason is to do with the memory addressing ports chosen by MAP, which unfortunately conflict with the way RP/M finds out whether it is plugged into an 811 or an 813.

The good news is that a very simple modification to RP/M cures this problem completely and has no unpleasant side effects. Gemini will (once the old stocks are used up) be issuing RP/M Version 2.1 as standard for 811 and 813, but for those of you with V2.0 and an EPROM programmer able to cope with the 2732, here is a complete description of the changes between 2.0 and 2.1.

Address	V2.0	V2.1
F068	30	31
F108	20	ED
F109	04	41
F10A	ED	20
F10B	41	02

The first change simply updates the version number, and the rest reverses two instructions. Please note that some people investigating this problem have suggested other changes. These will cause bad things to happen if the RP/M is used in an 813, and the changes described here are the only official ones. By the way, please do not ask Gemini to replace your V2.0 with V2.1, as they will not be amused! I suspect that since most GM811s have V0.1, less than 10 people in the world will ever put together a system which requires this change, and these people should ask MAP for help. Credit goes to Gemini for issuing V2.1 at all, since MAP are in competition with them. This reflects their awareness of the benefits that result from cooperation over the 80-BUS and keeping the whole array of products from different suppliers compatible.

As promised, here is the description of RP/M V2.0, which all V0.1 owners should study carefully.

RP/M VERSION 2.0 (OR 2.1) – A SUMMARY OF CHANGES TO VERSION 0.1

NEW FEATURES

The disk boot routine operates with both Pertec DD/DS drives and Micropolis DD/<SS or DS> double tracking drives.

Operates both with original GM811 and new GM813 CPU cards.

Automatic disk boot on power-on or reset if a disk card is in the system.

Operation is possible without a video card, using the serial interface.

Support of parallel printers is included.

THE KEYBOARD

If the CPU card is a GM813 then bit 1 of location 0003 (IOBYTE) is automatically set to 1 on power-on or reset. This disables the operation of the keyboard attached directly to the CPU card, since this is not provided for on the GM813.

Screen edit mode is entered by Control-% (Ed. - %='at') instead of DEL to be consistent with the various disk BIOSes.

Screen dump operates only within screen edit mode to be consistent with SYS and to allow the control code previously used to be available for program use.

RESETTING RP/M

On reset the system is tested to see if a disk card is present. If it is then an attempt is made to boot from the disk. If this fails an appropriate message is output as from the B command.

COMMANDS

The D command may be cancelled during output by typing a Space.

SCREEN EDITING

Use Control-% [% = 'at'] (Null) to enter screen edit mode. The cursor changes to a blinking block while in this mode. Entering Control-C exits screen edit returning a Control-C and a carriage return, instead of performing a warm boot.

SCREEN DUMP

The contents of the screen may be printed by pressing Control-B while in screen edit mode. When the screen has been printed screen edit mode is terminated automatically. The output is directed via the I/O jump table, allowing users who modify this table to obtain screen dumps.

PRINTER SUPPORT

Both serial and parallel printers are supported. To use a parallel printer use the S command to change location 0003 (IOBYTE). Bit 7 is tested to determine which type of printer to use. If it is 1 then the parallel printer is selected, so change the value from 01H to 81H or 03H to 83H. The parallel printer interface uses Centronics conventions. Port B is the output data port, and port A is used to control the data transfer. Bit 0 of port A is connected to the BUSY line from the printer, and bit 1 is the strobe which indicates to the printer that data is available.

OPERATION WITHOUT VIDEO CARD

RP/M is designed to be used with the Gemini GM812 intelligent video card. However for some applications it is useful to be able to operate the system with minimal hardware. RP/M will now operate without a video card if bit 0 of location 0003 (IOBYTE) is set to 0. Since this needs to be set on power-on, this bit is automatically flipped if a link has been installed on the CPU card. This is the Ring Indicator link attached to the Modem Status Port. If this link is made then the serial port is used as the console device. Note that serial handshaking is used, so ensure that this is provided. If there is a video card in the system it will be reset but it will not be used and its keyboard will not operate. If there is a disk card then an attempt will be made to boot a disk, as normal. All messages are output to the serial printer, and input may be from a serial keyboard or from a keyboard on the CPU card (GM811 only).

It is even possible to operate the system with no video card and no serial terminal, but with a keyboard on the CPU card (GM811 only) and a parallel printer for output, by turning on the computer and immediately using the S command without being able to see the serial output, to turn on the parallel printer.

There are several limitations on the facilities provided if there is no video card. No screen edit mode or screen dump is available. Also, no cassette input or output will operate, and use of the R or W commands results in an error message.

FIXED LOCATIONS IN RP/M

For certain applications it may be convenient to modify certain default values used by RP/M. These may easily be changed when using RP/M by using the appropriate command, but if access to an EPROM programmer is available then some initial values may be changed. These are stored at fixed locations in RP/M.

F009 contains the two byte value used as the UART divisor. The normal value is 417 decimal, 01A1 hex, which is stored as A1 01. This gives a speed of 300 bps (30 characters per second).

F00B contains the initial value of the IOBYTE. This is 01H, giving a serial printer. Change to 81H for a parallel printer.

F00C contains the number of lines per page. This is set to 66 decimal, 42 hex.

OTHER FEATURES

If a form feed character (0CH) is output to the video card by a program, this is translated to a carriage return and line feed.

While the console input routine is waiting for an input the cursor is displayed as a blinking underline. Otherwise the cursor is displayed but does not blink. This overcomes problems relating to programs that scan the console for an input instead of calling the console input routine.

An attempt to boot a disk, whether successful or not, does not change the contents of the program area. This allows a disk to be booted and then a Save command to be issued, providing another means of moving data between an RP/M and a CP/M system.

If an attempt is made to boot a disk without a disk card in the system, then an error message is output. Previously this could cause the system to hang up.

The screen editing logic has been improved to return the correct console status when an edit buffer is pending.

CONCLUSION

Please notify us of any problems with RP/M, as well as any suggestions for its improvement. We hope that this new and more advanced version will be as free of errors as the original version.

REVIEW OF THE MAP 256K RAM CARD

by RICHARD BEAL & D. R. Hunt

The DH bit:

I had almost completed my review of the MAP RAM by the copy date of the last issue, but lack of time did not allow me to finish, so when Richard produced his review of the card with particular emphasis on the software side of matters, I scrapped my review in favour of the following as his is far more thorough. However, Richard has not covered the obvious points concerning the hardware so I will detail them here.

The card is a NASBUS/8OBUS 8" x 8" card similar in appearance to all the existing RAM cards, and particularly the Gemini GM802, in that it contains a block of RAM surrounded by the necessary decoding and driving logic. The board is

supplied built in either 64K or 256K versions. The RAM block is socketed to allow easy upgrading from 64K through 128K, 192K to 256K. The quality of the pcb is of the standard expected of NASBUS/80BUS cards, being double sided through hole plated and coated in a solder resist lacquer. A slight departure from the norm here, as the solder resist is blue in instead of the usual green. Dare I say it, adding a touch of colour to the system. The pcb edge connector is gold flashed as is to be expected. Overall quality of construction is good (the early sample I had was hand soldered, I do not know if production boards will be flow soldered).

My only complaint from the hardware point of view is the manual supplied. This excels in the trend started by the early Nascom manuals and continued by the current DRI CP/M manuals in its total incomprehensibilty. (I hasten to add that there is not a lot wrong with the current Nascom manuals.) It is the most difficult document to understand. I wasted two evenings trying to access more than 64K, not because it didn't work, but because I couldn't understand the words. No examples of use were given, and the description of the mapping system was inadequate and referred to IC numbers, which in the absence of a circuit diagram, was less than helpful.

DH's conclusion:

The MAP RAM is well built and well engineered, and arguably worth the money if you can find something to do with it (read on), let down badly by totally inadequate documentation. Now software has been written for it, it is of much greater use.

... continued by Richard Beal:

The MAP 256K card, which I shall refer to as MAP RAM, is the first memory card for the 80-BUS which offers more than 64K. The history of the RAM cards is worth remembering.

First came the Nascom RAM A, which had 32K RAM and 4K ROM, which seemed marvellous at the time. This card was plagued with problems, some not perhaps its own fault, and it took a long time before a definitive set of modifications to make it work perfectly was produced (see INMC News Issue 7). To be fair, the original specification never included operation at 4MHz, which everyone wanted.

This was followed by the Nascom 48K RAM card, which worked perfectly, and had the apparently useless new feature of a page select system allowing any of four 64K pages to be selected. But this gave the first hints of what was to follow.

Gemini then produced a 64K RAM card, which was useful to disk system users, who needed the full RAM memory. It too supported the four page system. Then came the Gemini GM813 CPU and 64K RAM card which puzzled everyone by having memory mapping. This uses a high speed RAM between the CPU address lines and the memory chips, which alters the addresses selected so that many different physical blocks of 4K are mapped onto the 16 4K logical memory areas. This translation process is set up by output of values to port FF.

When MAP decided to produce a 256K card they therefore had a challenge, which was to produce a card which would operate with all the existing systems. They have put a great deal of thought into their product, and have achieved this aim excellently. MAP RAM works with the Nascom 1, Nascom 2 and Gemini GM811 computers using a new 32K paging system, which is very easy to control. It also works with the Gemini GM813 computer, exactly as Gemini intended, by adding more physical 4K pages which are addressed by the memory mapping system, as a logical extension of the GM813 design.

The next important question is, "Does it work?". The answer is a definite YES. There is no difficulty in getting the card going. If the card is plugged in to a Nascom or GM811 system, it immediately acts as a normal 64K RAM card. The appropriate software can then be used to activate the other memory pages. If the card is to be used with the GM813, then the two header plugs on the MAP board have to be changed, and one link altered. It would be better if this link had been made easier to change. MAP offer to help if you are not confident of making any of these changes yourself. You can have up to four MAP RAM cards in a system,

giving up to 960K of virtual disk. In this case each card must have a different header plug, and the manual supplied shows clearly how to wire these up.

It is also possible to buy the card with only 64K RAM and upgrade it later by simply adding the extra memory chips. MAP are quite happy that you do this and will supply the extra chips as you want them. This has a great advantage as a low cost approach, as the price of the 256 kbit chips will no doubt continue to fall. This makes the MAP RAM an interesting, although more expensive, alternative to the Gemini 64K RAM card.

I do have one major complaint about MAP RAM. The documentation is very hard to understand, and there is not a circuit diagram. There are no examples of how to program the MAP RAM, and not even a clear indication of the programming instructions you would use to address it. When so much good work has been done on the design of the card, it is a shame that a few hours more were not spent on the manual. Also it would be helpful to have a circuit diagram, as many people like to be able to work on their own equipment should anything go wrong. Instead of just criticising, I had better try to help! So here is some of the crucial information which I eventually deduced - with some advice from MAP, who are very helpful!

How to program the MAP RAM if you have a Nascom or GM811 computer

The control port is FE. If you output 00 to port FE you have a normal 64K card, and this is the condition when you turn the system on.

If you want to swop the bottom 32K of RAM with other 32K pages, you output a different value to port FE. You output C2 for the first extra 32K page, C3 for the next, and so on, all the way up to DF. To switch back to normal, output 00 to port FE.

For example to page in the third extra 32K page, execute the instructions

```
LD A,0C4H       ; C2 is the first extra page, so C4 is the third
OUT (0FEH),A
```

Then to return to normal

```
XOR A
OUT (0FEH),A
```

How to program the MAP RAM if you have a GM813 computer

The control port on the Gemini GM813 is FE. It must be addressed by a Z80 instruction using the C register to address the port. The top half of the B register must contain the logical 4K page (0-F) to which the physical memory is to be mapped, and the data value output must contain the number of the physical 4K block.

For example to swop the 4K of memory starting at 2000H with the fourth extra 4K block, use the instructions

```
LD BC,20FEH     ; 20 because 2000H, FE is the port
LD A,13H        ; 10H is the first extra page, so 13H is the fourth
OUT (C),A
```

To then return this area to normal, use the instructions

```
LD BC,20FEH     ; 20 because 2000H, FE is the port
LD A,02H        ; The normal blocks are 00-0F, so 02 for 2000H
OUT (C),A
```

More Advice

When paging in and out memory, there are two vital rules to remember (which I forgot several times).

1. If the program code is in the area which is paged out then the system will crash.

2. If the stack pointer is in the paged area and you use the stack the paged memory will be corrupted. If you depend on the stack contents (such as return address) still being there, then the program will crash.

MAP Software

Most MAP RAM users will probably never want to write their own software for controlling the memory paging, so perhaps the information about programming the card is not very important. MAP supply a modified version of the Gemini BIOS which supports the MAP RAM as a virtual disk of up to 512K, using a GM811 or a GM813. An alternative is to use a new version of SYS (Version 15.0 or later) which now provides support for MAP RAM for Nascom and Gemini computers, with a virtual disk of up to 960K bytes. (See article on SYS elsewhere in this issue). Most users of additional memory at the moment will want to use it as a virtual disk, which can be very useful. (See article on virtual disks elsewhere in this issue). However CP/M Version 3, otherwise known as CP/M Plus, is on the way (slowly - horribly complicated), and this requires 96K of memory as a minimum, if advantage is to be taken of most of its features. The MAP RAM will then be in even greater demand, assuming that CP/M Plus can ever be got to work on it!

[Ed.'s notes - from what we have seen so far of CP/M Plus (Gemini have a pre-release version running) the MAP RAM board is NOT the best way to provide the additional memory that is required, and it looks as though yet another method of RAM switching is yet to be born! The main attraction of CP/M Plus is its increased speed, gained by keeping various directory and data buffers in additional RAM. The way Digital Research have arranged this, it would seem that the type of memory switching provided by the MAP board is unsuitable. By the way, there are two versions of CP/M Plus, one running in 64K or less and one running in 96K or more, but I wouldn't hold my breath waiting for either as DRI will not yet state the anticipated release date of fully debugged versions. Watch this mag. for further details.]

The Great A19 Debate

The MAP manual points out that the GM813 does not bring address line A19 to the 80-BUS. While this is quite true, the GM813 was merely following the 80-BUS standard, which does not allow for A19 on the BUS. The obvious place for it is line 49, which was instead allocated as an additional ground by Nascom some time ago. This extra ground line does not appear to be needed, and everyone now seems to agree that the 80-BUS specification should have had line 49 as A19. But what should be done about it? Without A19 the maximum memory size is limited to 512K bytes, and people are already installing systems with a full megabyte. The MAP approach has been to unilaterally redefine the 80-BUS and they have used line 49 for A19. They suggest that anyone with a GM813 who wants to have more than 512K should modify it so that it no longer grounds line 49, but connects it to A19 instead. But if you do this you should be careful to check all the other cards in the system, and the motherboard itself, as any of these may have implemented the 80-BUS specification and grounded this line. It would be nice if Gemini agreed that this was a good idea, but I suspect that instead they may feel that it would be better to define some other line as A19, in order to keep to the universally agreed specification and avoid telling people to mutilate the GM813 [Ed. - and all other Gemini boards and some boards of other manufacturers, including Nascom, which as per spec. have this line grounded.] and their motherboards. On your system it is probably wisest to follow the recommendations which come with the large RAM card you buy, and make a note of what your 80-BUS does. See INMC 80 Issue 4 for the 80-BUS specification.

Conclusion

I have no hesitation in recommending the MAP RAM card, despite the poor documentation. It is well engineered and works perfectly. With the growing popularity of virtual disks, it is likely to be a success. I look forward to seeing the next MAP product which is rumoured to be a memory mapped (paged out) 80 by 25 video card complete with an optional floppy disk controller which is software compatible with the Gemini GM809 disk controller!!! Gemini are rumoured to be producing a new memory card, but there is no information yet on how much memory it will have. My guess is that 256K is the minimum they will consider, and 512K could be possible! Anyway, it looks as if Gemini are getting some tough competition, which will no doubt be good for us all.

The MAP RAM is available from MAP 80 SYSTEMS LTD., as advertised in this issue.

SYS – LATEST DEVELOPMENTS

by RICHARD BEAL

In the December 1981 issue of INMC 80 (Issue 5 and last) I wrote an article about SYS. At that time I said that I had just finished a new version which supported the Gemini double density disk card. That seemed quite a step forward at the time, but a lot has happened to SYS in the last year, and a new version, V15.0 has just been released, so I thought it would be a good time to bring you up to date on the latest developments. First I will just give a few hints of what it now includes:-

NASCOM --- GEMINI GM811 --- GEMINI GM813 --- GALAXY --- QUANTUM
MICROPOLIS --- WINCHESTER --- SHUGART --- 'SHUGART COMPATIBLE' --- 8 INCH
PERTEC --- CROMEMCO --- RML --- XEROX --- SUPERBRAIN --- RAIR
VIRTUAL DISK --- VIRTUAL BOOT --- MAP 256K --- MEMORY MAPPING
MEGABYTE OF RAM SYSTEMS --- CONFIGURATION MESSAGES
'FIXED' SCREEN EDITING --- SCREEN PAGING CONTROL --- SCREEN DUMP

First a brief description of what SYS is, for those few of you who are not yet using it!

SYS is a CP/M program which replaces the BIOS of the CP/M system with an expanded BIOS which has many additional features. The BIOS is the part of CP/M which deals with input and output.

SYS can be executed automatically on Reset or Cold Boot and it automatically relocates the new BIOS to match whatever size CP/M system is in use. This means that it can easily become an integral part of the computer system.

SYS is always stored on disk with the standard name of SYS.COM, but there are a vast number of possible configurations which support different hardware requirements. These are selected by conditional assembly.

SYS may be used on Nascom or Gemini computers. It requires a Gemini GM809 or GM829 double density disk card. At present it supports Version 2.2 of the CP/M operating system.

SYS contains two versions of the disk software. The first of these supports only Pertec 35 track 48 t.p.i drives or Shugart compatible 48 t.p.i drives (if the time constants are changed to suit). However it has the advantage that it is able to provide support for a whole range of disk formats which are described below, allowing data to be exchanged with many other types of computer. The second incorporates the Gemini standard disk software which includes support for Winchester hard disks and standard eight inch floppy disks, aa well as further options for Pertec, Micropolis, and other 'Shugart compatible' 96 t.p.i five inch disk drives. See the suppliers note on Winchester version availability at the end of the article.

The main features of the SYS expanded BIOS are:-

(a) Full screen editing, which allows the cursor to be moved back up to a line already on the screen, so that the line can be edited and reentered to CP/M. This feature, also found in all of Gemini's BIOSs for Nascoms and Geminis, is most unusual if not unique for CP/M systems.
(b) Screen dump to the CP/M list device, so that an image of the screen can easily be printed. (Also now in Gemini's own BIOSs.)
(c) Automatic screen paging, so that information does not roll off the top of the display before it has been read.
(d) Support of the Nascom screen display or the Gemini Video Card. An additional keyboard is supported on the Video Card.
(e) Support of the Nascom keyboard or an ASCII encoded keyboard. On the Nascom keyboard, the action of the Shift key may be reversed by pressing Control/Enter. The @ (@ = 'at') key when not shifted may be used as a Tab key or as an alternative Control key. A number of command strings may be automatically generated by pressing the GRAPH key and a letter, and this simplifies the entry of several commonly used commands.
(f) Full support for the CP/M IOBYTE option.
(g) Support for both serial (Teletype compatible) and parallel (Centronics type) printers. A variety of options are available including handshaking and automatic handling of form feeds so that printers without a page throw mechanism will operate correctly.
(h) Ability to automatically identify single density disks in drives B, C and D. The primary format supported is the SD Systems format, extended to also use the second side of the disk. This format was used by the Gemini/Henelec GM805 single density disk system. Special versions can be generated which instead allow alternative single density formats to be used. These are the Cromemco SS/SD, RML SS/SD and Xerox 820 SS/SD formats. In each case these formats require 40 track drives which are not supported. You must take care to access only the first 35 tracks of the disk if you use these formats.
(i) Ability to read and write other double density disk formats. Special versions can be generated which allow an alternative double density format for disks in drive B. At present, Superbrain QD (DS/DD) format, Nascom SS/DD 80 track (only with a Shugart compatible interface 96 t.p.i drive) and Rair DS/DD formats are supported.
(j) Optional read after write checking for all changes to disks. This gives greater security while decreasing performance.
(k) Standard disk software supporting Winchester hard disks (see note on availabiility), standard eight inch floppy disks, Pertec, 'Shugart compatible', and Micropolis five inch disks.
(l) Support of a virtual disk which appears exactly like a real disk but is in fact additional memory. This can use the standard 64K page mode allowing either additional RAM cards, each of up to 64K. This option is now also included in Gemini's own BIOSs. An alternative option allows the use of MAP memory cards with up to one megabyte of memory. These operate either in 32K pages with the Nascom 2 or Gemini GM811 CPU cards, or in full memory mapped mode with the Gemini GM813 CPU card.
(m) Warm boot from the virtual disk, which makes this process very fast.
(n) Extensive messages are displayed when SYS is executed, describing the configuration for which it has been generated and the main features included.
(o) Most features are set by easily understood assembly options, allowing either a small simple BIOS or a large advanced BIOS to be generated to the exact requirements of the user.

'Fixed' Screen Edit

A new feature of SYS which requires some explanation is the 'fixed' screen edit mode. At the moment if you enter screen edit mode, edit a line and press Return, the system returns to normal. But back in the days of NAS-SYS you were

effectively in screen edit mode all the time. This was very useful when, for example, editing BASIC programs, as you did not have to remember to enter screen edit mode before editing each line. Now this feature is available with SYS and CP/M, using the new 'fixed' screen edit mode.

To enter 'fixed' screen edit mode, enter screen edit mode, and then again press the key which activates full screen editing. This makes the cursor on the IVC change to a solid non-blinking block, and this means that you are permanently in screen edit mode. As before, press the Return or Enter key to enter a line as input. When CP/M requests the next input character, screen edit mode will automatically be reactivated. To escape from this mode, press the key a third time. This feature is particularly useful when editing BASIC programs, as a LIST command can be followed by extensive editing of the lines displayed, without having to remember to enter screen editing mode for each line.

Screen Paging Control

Another improved feature of SYS is control over screen paging. If too many lines are output to the screen without any input being obtained from the keyboard, then it is possible that information might roll off the top of the display and be lost. Whenever this could occur, the following message is output:-

"*** Press ^C, ^S, R, W, K or Space ***"

If you press Control/C or Control/S then this character is returned as the next input character to the program being run.

If you press R then the screen paging feature is disabled until the next user input. For example it would start to operate again if Control/S was used to pause the output display.

If you press W then the screen paging feature is disabled until the next warm boot.

If you press K then the screen paging feature is disabled until the next cold boot, or until SYS is executed.

If you press a space then the next page of output is displayed.

MAP 256K RAM

SYS now provides full support for the MAP 256K RAM card, allowing virtual disk systems with up to a total of one megabyte of RAM. This is described in more detail elsewhere in this issue. SYS uses the new 32K paging method for the Nascom and Gemini GM811, and with the GM813 it uses the full memory mapping capabilities of the GM813 and of the MAP RAM. SYS provides warm boot off the virtual disk, and this speeds up this process considerably. It also has the advantage that you don't need to worry about having the correct data on the system tracks of your disks, as these are no longer used except on cold boot.

Restructuring of SYS

In order to allow the support of three different types of virtual disk, as well as the inclusion of two completely separate versions of the disk software, SYS has had to be restructured. It is now much more easily maintainable, as the different parts are stored in eight separate source modules. These are SYSB1.MAC to SYSB7.MAC, and SYSB6A.MAC which contains the alternative standard disk software. As usual the user has only to edit SYSB1.MAC, which now contains only the option switches and various helpful comments, and then submit SYSB.SUB, which does the assembly and link. This takes about five minutes, which is very fast considering the size of code which is being processed. M80 actually stops and thinks to itself for a bit when it has to generate all the relocation labels, so there is no need to worry if your system becomes silent. The M80 assembler and L80 linker are required, and I recommend release 3.44, which I know to work correctly.

Problems with SYS

Various problems have been reported by users of SYS, and most of these have been cleared up very easily. For example you should not try to run a Gemini computer using a SYS generated for a Nascom. It may be helpful to list a few problems which people have encountered which have proved to not be errors in SYS.

When a Nascom with SYS is switched on and SYS is executed the system may stop and wait for an input character before the configuration messages are displayed. This is caused by a simple hardware problem. The serial input port has supplied a spurious null on power-up which has activated screen edit mode. The solution is either to assemble SYS with the SKBD option set to FALSE, or connect the serial input line to ground so that no character is received. Remember that the serial input may be set to RS232 or cassette, and ground the appropriate line.

When spurious input characters appear on the screen there are several likely reasons. Check that the GEMINI flag is set correctly, and check the SKBD, NKBD, GKBD and VKBD options carefully. GKBD must be FALSE with the Gemini GM813, as there is no keyboard port. If you have a Gemini video card it is best to plug the keyboard into it and set VKBD to TRUE.

If the system crashes when SYS is executed then it may be configured incorrectly. For example it may specify a virtual disk when none is attached. Alternatively there may be insufficient memory for the SYS BIOS if the correct changes to MOVCPM have not been made (see above).

If a virtual disk is used and a different configuration of SYS is executed, then the directory of the virtual disk may become corrupt. The system should be switched off and on again to be certain of clearing this problem. This is because cold boot does not clear the contents of the virtual disk, if it appears to be initialised already.

Conclusion

I hope that you continue to enjoy using SYS. I have to admit, as Gemini point out, that it is really only suitable for computing enthusiasts, but that is who it was written for. Finally, my thanks to David Parkinson and Gemini for their help with the inclusion of the standard disk software.

Please direct any queries about SYS to the supplier, as I only have time to look into any really difficult problems, as with luck there aren't too many of those!

Suppliers note.

As the Gemini Winchester drivers have been incorporated with the kind permission of Gemini, it has been agreed that the version of SYS incorporating the Winchester drivers will only be supplied if the user produces evidence of owning a Gemini GM835 Winchester. This is unfortunate for those who have the odd Winnie knocking about in the junk box and wish to put it into use. On the other hand, it does allow those who already have a Gemini GM809/GM815 system running on a Nascom and who wish to add a Gemini GM835 to do so without scrapping the whole system and starting again. On the whole, Gemini's wish to protect their software is understandable.

SYS is available from HENRY'S RADIO, see their ad. in this issue.

VIRTUAL DISKS – ARE THEY USELESS?

by RICHARD BEAL

Are virtual disks a useful invention or are they a desperate attempt to find a use for lots of RAM? A virtual disk, also known as a memory disk or pseudo disk, is a way of using a large quantity of RAM as if it is a real disk. In CP/M terms the translation of track and sector to a memory address in the extra paged RAM is done by special code in the BIOS.

The SYS BIOS has supported a virtual disk for about a year and a half, using the original Nascom four 64K page system, but this was prohibitively expensive and was limited to a virtual disk size of about 128K. The introduction of the MAP 256K RAM card, no doubt to be followed by a similar product from Gemini, is already making large memory systems common, and virtual disks will be widely used as a result.

When I first used a virtual disk, I made it into CP/M drive P, and in fact on many systems it is known as drive M. The problem is that one tends to forget about it, and since the contents of the disk are lost when the power is turned off, a lot of time can be spent moving files from real disks to the virtual disk, and then later moving the new files back.

So for a long time I had the luxury of a virtual disk, but hardly ever used it. Then I started thinking about how it could be made more useful, and after much experimenting, came up with several ideas. The first was to reduce the time taken to warm boot by restoring the CP/M image from the virtual disk. This works well, and for the first time I began to find the virtual disk useful.

The next change was to prevent the virtual disk being wiped clean if I pressed Reset. This proved easy, as I simply put in a check to see if it was already initialised, and if it was then the initialisation was skipped.

Next I thought about how the speed of executing Submit files could be speeded up, since they are so amazingly slow. If the $$$.SUB file was written to the virtual disk instead of to a real disk, then the Submit overhead time ought to vanish. This needed several changes, because the virtual disk would have to be drive A, as I don't believe in modifying the BDOS. At the same time I didn't like the idea of confusing myself by moving all the real drives up by one to B and C, which was one suggestion. Programs like BACKUP and FORMAT would still refer to the real drives as A and B and I could imagine myself making some terrible errors. Therefore I decided to flip drives A and P, so that the real drive A would be called drive P and the virtual disk which was drive P would become drive A. Also I allowed drive P to be called drive M in case people preferred that.

This all seemed a marvellous idea until I tried it. The system silently warm booted up with drive A the virtual disk, but naturally there were no programs on it, so I had to log in to drive P! The solution to this problem was to make drive P the default logged in drive, so that on cold boot or after a drive select error, the system would come up with the prompt "P>". A strange sight. Now I really thought that I had solved the problem. But when I tried SUBMIT it didn't work! Of course the reason was that SUBMIT writes the $$$.SUB file to the logged in drive, and only executes $$$.SUB if it is on drive A. Now I could see that I was getting near to the answer. I had the source code of the excellent program EXSUB, which I had debugged earlier, and it is designed to be easily modified to force the writing of $$$.SUB to drive A regardless of the logged in drive. Success at last - using SUB was now a pleasure, and the EXSUB facility of being able to submit several commands without using a text editor to create a .SUB file became more useful.

But the final benefit of making drive A the virtual disk came unexpectedly with the use of CCPZ, which is a much improved CCP written in Z80 code and available from the CP/M Users Group. This CCP implements a very clever search for programs which you try to execute. For example if you are logged in to drive P, as user 4, and you try to execute a program, first it looks on drive P user 4. If this fails it looks on drive P user 0. If this fails then it looks on drive A user 0. And of course this is now the virtual disk. So if you put programs on the virtual disk then they will always run even if you type in the command while logged in to another drive. CCPZ has lots of advantages, including making the User number feature work sensibly, which is very useful particularly if you have a hard disk. In fact Gemini have had the great good sense to supply it with the MOVCPM for their Winchester system, instead of the standard CP/M CCP.

Several software products which make good use of disks to extend their ability to deal with large quantities of data benefit greatly from use with a virtual disk. A good example is WORDSTAR, which may be used to edit a text file of say 100K. It automatically reads and writes the file to disk so that a large

portion is in memory, but if you keep moving up and down the file the delays become painful. But if the text file is on a virtual disk, then WORDSTAR continues to be fast, even with vast files. Another example is the amazing VIZAPL which is a full implementation of APL for microcomputers. It is available for the Gemini IVC, which displays all the APL characters very beautifully. VIZAPL has the amazing ability to extend real memory using its own virtual memory technique to move little used data and procedures to disk. If the virtual workspace is put on to the virtual disk, then the speed improvement with large APL workspaces is very impressive. It is like having the memory addressing capability of a 16 bit microcomputer, but on an 8 bit Z80.

To summarise the advantages:-

(a) high performance exceeding that of any real disk, whether floppy or hard;
(b) no disk wear or noise in operation;
(c) ideal with CP/M if CCPZ used, and for SUBMIT operations if it is drive A;
(d) special benefits with some software such as WORDSTAR and VIZAPL.

And the disadvantages:-

(a) more expensive than real disks - but this is getting better;
(b) a power cut can be a disaster if you run for hours without backup;
(c) it can be boring moving files to and from the virtual disk.

So are virtual disks useless? No.

E.V. BEEPER – A MINI-REVIEW

by RICHARD BEAL

Recently I was using a conventional CP/M system (a Rair Black Box) and I found that it had a feature that I wanted (apart from a hard disk). To be accurate, the terminal had the feature, not the computer. It went BEEP. I hadn't realised that various bits of CP/M software go BEEP to warn you of things, and of course it is very easy to put

```
PRINT CHR$(7)
```

into a BASIC program. All you need is a beeper. I found that one already existed, from E.V. COMPUTING LTD. I have a Gemini intelligent video card (IVC), and this already supports a beeper by putting out a signal when it receives a Control/G (07H). The beeper detects this signal and goes BEEP. I ordered the beeper by phone, using a credit card, and the beeper arrived all the way from Manchester the next morning. Ten minutes later, with the help of very clear instructions, it was working perfectly. I had a nasty fright when I turned the machine on, because it went BEEP at once. In fact it now always goes BEEP whenever it is turned on or off. The BEEP is really more like a choked warble, and the manual describes how to modify the beeper to make it warble differently.

The instructions explain how it can also be used if you do not have an IVC, using a signal from the Nascom keyboard port. This requires some extra software to be patched in, and this would need a bit of work by the user, but it is well documented in the instructions.

I recommend this product, which works perfectly, for those who want a simple beeper. It does not play music or sing, but it is very reasonably priced at £12.50 plus VAT.

Queries to:-
E.V. COMPUTING LTD., 700 BURNAGE LANE, BURNAGE, MANCHESTER M19 1NA.
Tel. 061-431 4866

80-Bus + UCSD = Dynamite

by Mike York

Introduction

References have been made in 80-Bus News (and its predecessor) to "UCSD" and in the last issue of 1982, S. Monger mentioned that I had implemented it for the Nascom then confessed not knowing what it is. Working on the assumption that many other readers will not know what it is I am writing this short article to explain the origins, subsequent development and current state of the UCSD p-system and its implementation for 80-Bus computers. You may also find it worth your while to read a series of three articles in PCW (July - September, 1982) and an article in the first issue of Computer Answers (Nov/Dec 1982).

What is UCSD?

Many people know of UCSD Pascal. It was the first full-scale Pascal implemented on micro-computers (and was adopted by Apple as "Apple Pascal"). The original Pascal compiler was then used to develop a complete operating system for microcomputers that is not just a rival to CP/M but contains many features which CP/M users can only dream of. Subsequently other compilers (FORTRAN, BASIC, Lisp, APL, Modula-2 and the new INMOS language Occam) have been added and the only remaining attachment to Pascal is now historical (and the fact that the use of Pascal naturally influenced that history).

I was first introduced to UCSD two years ago and was so impressed that, after a year of uselessly wishing that someone would buy me some disk drives or sell an automatic cassette deck with UCSD as its operating system, when I finally got my disk drives, I decided to forget CP/M and go straight for UCSD on my Nascom and have been actively using it now for nearly a year.

The major and, I believe, unique feature of the system is its use of "p-code" or pseudo-code. P-code was originally specified as a pseudo-machine code to run on a hypothetical stack-oriented processor. This was to facilitate the compilation of Pascal in a single pass. (Subsequently a real microprocessor, the Western Digital Microengine, was developed to run p-code as its native code.) This p-code is subsequently executed by a p-code interpreter which is coded in the native code of the microprocessor being used. This use of p-code gives an incredible portability to UCSD software. As long as a p-code interpreter exists, any program compiled into p-code (including the whole of the UCSD operating system) will run on any microprocessor! Except in special circumstances, the same programs that run under UCSD on an Apple, IBM PC, Sirius, Sage II, etc. will also run on a Nascom or Gemini without change - and vice versa.

This sort of portability is impossible under CP/M without resorting to expensive "Z80 cards". (And just look at the mess CP/M has got into in upgrading to 16-bit: they have even had to produce their OWN Z80 card for the IBM PC to allow users to run their old software!)

Of course this portability has been bought at a price. P-code, because it is interpreted, runs slower than native code. Supporters of CP/M-based Pascal compilers have been quick to point out that the Pascal Benchmarks published in PCW run quicker under Z80 compilers than p-code compilers. Of course, the importance of execution speed is tremendously exaggerated by these critics. (During the last year I have not once been adversely affected by slow execution of p-code.) However, even if it were important, such critics are now out of date. There now exists a Native Code Generator which can translate a p-code program, wholly or partly, into Z80 native code. The resultant code file can then be saved and treated as any other p-code file - except that during execution, those passages which are time-critical will now run at native code speed. With UCSD,

therefore, you have the choice. Either maintain portability (and more economical use of memory) using p-code or speed up execution using native code. You can even keep two versions of the program: one purely p-code and the other optimised for speed.

There are, of course, other hardware features, besides the processor, which affect portability. UCSD tackles this problem the same way as CP/M -- you have a hardware-dependent BIOS especially written for the particular machine. However, even here, UCSD have gone one better than CP/M by simplifying the BIOS considerably (and calling it the SBIOS - Simplified Basic Input Output System). This is done, for instance, by standardising the Blocking/De-Blocking process in a way that is transparant to the SBIOS writer. Other advances over CP/M include the ability to redefine control characters to suit your console device, optional input queuing (type ahead, etc.) and flagging of "events" to signal concurrent processes, etc.

Mention of Concurrency brings me back to the compilers. UCSD is a multi-tasking operating system. You can run programs which execute several concurrent processes at once - switching between them as necessary. (It is not, however, a multi-user system -- although it can support remote linking to a host computer or networking.)

Other powerful features that are normally available only on mainframes include separate compilation of "Units", overlaying (swopping in and out of memory from disk -- with all variables protected) of "Segments", the use of "Libraries" of pre-compiled routines and the linking of routines from different compilers, or assembler, into one program.

Besides these advanced operating features, the Pascal compiler, itself, has several enhancements over the Jensen & Wirth specification and these enhancements have been the model for most of the other Pascal compilers developed since for CP/M. Most notable is the implementation of Random Access to disk files (left out by Jensen & Wirth who dealt primarily with sequential files) by the new intrinsic procedure "Seek" which enables reading and writing of random records in a file.

The only major feature of Jensen & Wirth (or the ISO standard) which is missing is the ability to pass procedure identifiers as parameters -- but this is not used very often anyway and is planned for future versions.

What does it do?
~~~~~~~~~~~~~~~~

UCSD will be useful to system developers, applications programmers and business users, and, if you have the hardware, graphics freaks. Number crunchers can choose four-word reals (instead of two) if they wish.

When you boot up you are presented with a prompt-line giving a menu of possible commands. (This can be altered to produce a customised "turnkey" system if desired.) These commands are implemented by a single key and the prompting system is followed with each of those commands in a tree-structure.

When you buy CP/M you get an editor and an assembler thrown in. However, if you have a Z80 the the assembler is useless and unless you are dependent on an old teletype terminal the editor will soon cause much gnashing of teeth and tearing of hair. To get a reasonable screen editor and Z80 assembler you could end up paying several hundred pounds extra.

With UCSD you get a very powerful screen editor which doubles as a modest word-processor, and a macro/conditional assembler (that compares favourably with M80) thrown in as part of the standard package. A line-oriented editor for those using teletype terminals is also included. If you want to download assembled
~~~~~~~~~~~~~~~~

machine code onto a different processor then a Cross-Assembler package is an extra. Printer spooling is also supported and a turtlegraphics package is available. A special utility to read and write CP/M files is also available enabling the transfer of data and text files from one operating system to another. This utility even goes one up on PIP by allowing you to reconfigure the block size, disk capacity, directory length, etc. for different disks. (With PIP you are stuck with the parameters specified in your BIOS.)

On the commercial side, UCSD still lags a little behind CP/M (in terms of the amount of applications software available), but that situation is changing rapidly.

Although originally developed for a university/educational environment, it was commercial interest which led to the setting up of SoftTech Microsystems to market the UCSD p-System. When implementations of UCSD for the Apple and SuperBrain became available - the Apple adaptation, in particular, proving very popular (outselling even Visicalc) they were the motivation for a wide range of business software. More recently, Sirius, IBM and Osborne have adopted UCSD. The latest state-of-the-art business micro, the Sage II, based on the Motorola 68000 and streets ahead of the other 16-bit micros, has adopted UCSD as its favoured operating system. In addition, many universities (including Oxford, Cardiff, Bath and Edinburgh) run UCSD as their favoured environment for micros. The portability and adaptability of the system means that it can run on even cheaper micros (e.g. Nascom/Gemini) and the price advantage must surely make UCSD on these micros a very attractive proposition.

A considerable amount of business software is now on the market (a brief list is available from me if you send an SAE (get my address from my Ad.)) for UCSD, including the most advanced accounting, modelling, word processing and database packages. (Some of these have even received the unprecedented bouquet of approval by IBM for their Displaywriter and Personal Computer.) Most of these packages were originally developed on an Apple. They were thus designed to run in limited memory. The demand for them to run on IBM, Sirius, Sage, etc. has now required that any hardware dependent features be removed and only the standard portable I/O facilities be used. There should, therefore, be no problems in running them straight away on a Nascom with 64K and some may run in 48K.

Nascom Implementation

When I set about implementing UCSD on my Nascom I was unemployed and desperately looking for some way to earn some pennies. From the beginning, therefore, I had my eye on the possibility of making the system available for sale.

I was immediately faced with the prospect of lots of different 80-Bus compatible disk controllers, video controllers, and printers, several different types of drives and several different disk formats. Since I had 8" drives (which I got cheap because they had been discontinued) I obviously had to write my own disk drivers anyway and so I decided to write a new boot ROM which would contain the drivers for virtually any mix of controllers, drives and formats. This is called AllBoot and more information can be had by sending an SAE to me. (I decided not to bother with the old Henelec controller which used the PIO and was thus too limited and incompatible with the Gemini GM809 and Lucas Logic FDC to make it worth my while. I have not yet looked at the new Gemini GM829 FDC/hard disk card either. [Ed. - from a 5.25" point of view it is 100 per cent compatible with the GM809])

Now for the actual implementation:
The version provided for the Nascom (and planned for Gemini) will be an implementation of the latest (version IV.1) Z80 Adaptable System. This allows for future extension of the SBIOS to accommodate a hard disk and provides a versatile

and comprehensive programming interface to the operating system to facilitate sophisticated applications programs. The SBIOS for Nascom has been implemented via AllBoot (mentioned above) that enables the disk drives to be configured under software control. Thus almost any legal combination of 5" and 8" drives with Nascom and Gemini(GM809) FDC cards and format (including the ability to use 48 tpi formatted disks on 96 tpi drives) can be catered for and reconfigured, at will, while the system is running. The system can be run (although with a couple of inconveniences) using the normal Nascom 16*48 display. However facilities for a Gemini IVC and/or serial console are provided as are "hooks" to enable a Lucas Logic AVC, or other CRT or keyboard of the user's choice, to be used.

The printer can be software configured to use either the PIO (for a Centronics type) or the UART (serial).

AllBoot sits at F000 - F7FF and RAM is required from FC00 for the SBIOS and F800 if Nascom video is used (can be changed on booting to any other address to which the VRAM select is decoded). A minimum of 48K (60K is more suitable) RAM is required below F000 and at least one disk drive (two is more sensible) with at least 175K.

The Gemini version will be essentially the same except that it will be totally RAM based (and with about 1K extra RAM at the top end of the memory map), the Gemini boot ROM paging out after loading, and the Nascom video option will obviously not be possible.

(Many thanks to Lucas Logic for their help with 5.25" disk drives, by the way, and also to Dave Hunt at Henry's for patiently answering countless queries.)

User Group
~~~~~~~~~~

About two years the UCSD p-System Users Society (USUS) was set up in the USA, followed shortly by USUS(UK) over here. They have a large library of programs (about 20 single density 8" disks' worth) in various formats available for the cost of the media and copying.

They also organise two full-scale conferences (usually at a University) every year and distribute a Newsletter rather like this full of technical tips and reviews of the latest aplications packages.

They can be contacted via:

Mark Woodman
Membership Secretary USUS(UK)
Mathematics Faculty
The Open University
Walton Hall
MILTON KEYNES
MK7 6AA

How to get it
~~~~~~~~~~~~~~

I almost forgot (he said lying through his teeth) you can buy the Nascom implementation (and soon Gemini/Galaxy/Quantum versions) from me, see my Ad. in this mag. Alternatively hassle your Nascom/Gemini dealer and tell him/her to order one thousand copies from me as soon as possible if they don't want to be left out of the rush.

RANDOM RUMOURS (& TRUTHS)

by S. Monger

Me again. And with 'not a lot' of new products to reveal. All is relatively quiet, although I think that there is a fair amount of behind-the-scenes activity in certain quarters.

It seems that anybody looking for a video card will soon be faced with a bewildering choice. In monochrome, for 80x25, there is the 'de facto' Gemini GM812 IVC. This is now about to be joined by the MAP VFC, a combined 80x25 video and 5.25" FDC card. It's good to see another company producing 80-BUS/Nasbus compatible cards, but it is a shame that they are trying to reinvent the wheel instead of bringing along totally new boards. The majority of dealers feel that this card is unlikely to break any sales records. As a video card it loses to the IVC by having no PCG (programmable character generator), no on-board CPU (for intelligence and secondary programming), only one screen format, no light pen socket, and an unbuffered keyboard input socket (as an option). One also has to consider that neither Nascom or Gemini are likely to include support for this card in their CP/M or other operating systems, and nor is Richard Beal likely to include it in his SYS BIOS program. Price of a 'video only' VFC is £110 against the IVC at £125.

As a disk controller the MAP VFC is claimed to be 100% 5.25" compatible with the Gemini GM809 and GM829 cards, no doubt so that MAP can say that CP/M is available (through Gemini) until the day they can afford their own Digital Research licence. It does NOT however have 8" floppy and SASI hard disk support, like the GM829, or an 8" option, like the GM809 and Nascom FDC. Thus at £115 'yer pays yer money and yer takes yer choice' against the Gemini GM829 and Nascom FDC at £145, or the soon to be discontinued GM809 at £125. (N.B. exchange (second-hand & Gemini retested) GM809s will be available from MicroValue dealers at £100 as/if people upgrade to GM829s.)

In fairness (wot, me fair?) to MAP, the card does make a little sense when bought in its combined form at £199, or when bought as a kit, and it may provide a desparately needed spare card slot, but the comments on software still hold, and it is a shame that they haven't put their efforts into something else.

In colour the choices become bewildering. In order of cost there is the Nascom AVC at £185 (10"x8", relatively slow, but with some excellent support software for Nascoms), the Climax at £199 (super fast vector drawing, but only just about to actually become available), the Baby Pluto at £299, and Pluto at £399 (very difficult to get hold of due to 'run-away success', also see Dr. Dark's review).

Finishing off on my video topic, I will also throw in the rumours heard that there will shortly(?) be available a low-cost colour card (circa £100) ideal for games and simple animation, and also another (more expensive) monochrome card, with more advanced graphics facilities than on the two current cards.

Lucas/Nascom have recently had a dealer meeting where they handed out pretty display cards. One advertised Nas-Net at 'only £39.95 per station'. Does this mean I can have a 10 station networked system for less than £400? I don't think that is what they meant somehow!

Conspiracy? Why else are Lucas/Gemini/MAP/Quantum in consultation? Well, look out for a new computer from one of these companies, that has the physical appearance of a product from one of the other companies, and contains boards from three of these companies. Puzzled? Well, there isn't anything revolutionary to get excited over, other than the fact that these people are actually talking to one another!!!

UCSD p-system for Nascom

The renowned UCSD p-system has now been adapted for Nascom microcomputers with virtually any combination of Nascom or Gemini drives, 5" or 8".

A complete operating system, with screen editor, macro assembler, optional Pascal, BASIC and FORTRAN, Native Code Generator and Turtle-Graphics, there is a comprehensive and highly portable range of business and applications software available and virtually complete software compatibility with other computers running UCSD -- including Sirius, IBM, Apple and Sage.

Basic p-system (includes screen editor and macro assembler & AllBoot).....£225
Compilers:
Pascal.....£140
FORTRAN.....£195
BASIC.....£125

AllBoot (All-purpose boot ROM for Nascom)....£25

Add VAT @ 15%.

Gemini/Galaxy/Quantum versions soon.

Large SAE to: Mike York, 9 Rosehill Road, LONDON SW18 2NY (Tel. 01 874 6244) for further details.

NEW FOR NASBUS/GEMINI
The MAP V.F.C.

80 Column x 25 line screen, and Floppy disk controller on a single 8" x 8" professionally built plug in card. Fully compatible with the current range of NASCOM and GEMINI products.
FEATURES: 80 x 25 paged memory mapped screen
Flicker free display
Standard ASCII character set
128 graphic chars or inverse video
Onboard software
Video switch & keyboard port options
5 1/4" floppy disk controller
Available separately as kits or built boards.

VIDEO CARD : £89 KIT £110 BUILT
FLOPPY CONTROLLER: £95 KIT £115 BUILT
VIDEO & FLOPPY : £165 KIT £199 BUILT
OPTION & EXPANSION KIT POA

We are pleased to announce that we are supporting our new VFC with a 5 1/4" floppy disk drive system using 96 TPI, SS/DD Teac half height drives, in a slim case complete with power supply and all cables.

SINGLE DRIVE SYSTEM (500Kb) £299 BUILT
DOUBLE DRIVE SYSTEM (1 Mb) £499 BUILT

256K RAM CARD-64K Ver £105 KIT £150 BUILT
(Expandable to 256K - expansion kits POA)

SOFTWARE - We can support most systems with CP/M software - Ring for details.

ALL PRICES EXCLUDE P&P (£1.50) & V.A.T.

MAP 80 SYSTEMS LTD.,
333 GARRATT LANE, LONDON,
SW 18. Tel 01-874 2691

HENRY'S INCREDIBLE CP/M UTILITES DISK.

All the things you ever wanted: The Disk cataloguing and file dating suites. File compare, string and byte search utilities. ASCII file compression and expansion programs. A new system independant disk repair utility. A new SUBMIT with fully interactive input. The revised DRI PIP.COM including all the official fixes. And many more. Almost all are standard CP/M utilities and will work on any CP/M system. Supplied in either Nascom and Gemini formats. The price of this gift ? A mere £15.00 + VAT (£17.50 + VAT in Gemini SD format as it's too big to fit on one disk).

RICHARD BEAL'S NEW SYS BIOSes

SYSN7, the ultimate SYS for the original Henelec/Gemini G805 CP/M disk system, suitable for either CP/M 1.4 or 2.2, Nascom 48 column video or Gemini IVC 80 column video.

SYSB15 Super SYS for the Gemini computers and Nascom/Gemini hybrid computers. Compatible with the Gemini GM809 controller card and the new Gemini GM829 SASI controller card. When used with the GM809, support for Shugart compatible and other 5.25" drives. Also GM829 support for 8" and the Gemini GM835 Winnie as well (Winnie drivers only supplied on providing evidence of owning a GM835).

Support is now provided for using Gemini 64K or Nascom 48K page mode RAM cards (maximum 128K) or the MAP80 Systems 256K RAM card (maximum 1M Byte) as a virtual disk.

Provides all the goodies of the previous SYSes and lots more. When used with 48 t.p.i. drives, gives limited read/write compatibility with Super Brain QD and DD formats, Rair, Cromenco, RML and Xerox formats, and copying facilities to/from Osbourne. And, of course, the original Gemini/SD compatible format. A superb piece of software, ask anyone who uses it.

Supplied as a full source listing for the Microsoft M80 V3.44 (or later) assembler. Please supply your full system configuration if you require SYS ready assembled for use. As this is support software, it's Richard's (and our) opinion that the price should be as low as possible, so it's £10.00 + VAT. Upgrades from earlier SYSes £5.00 + VAT (return the original disk).

NEW SUPER DISKPEN

DISKPEN has been rewritten and revised. This popular text editor/formatter now includes a 'HELP' facility, and new features for the print control of most popular printers, underline, bold, etc, (also user patchable for the less popular types). New features include block delete, better move commands, new cursor control, optional hyphenation, visible indentation (margin) setting and lots more. A major enhancement is the ability to handle overlay files so that Pen can use auxilliary packages such as the multi-column formatter (that printed this). Details on writing your own overlays is provided for the machine code programmer, allowing the user to write special versions for special purposes.

The new DISKPEN is suitable for use with Nascom/Gemini hybrids (using the Gemini GM812 IVC card), and all Gemini computers, and is available as a £15.00 + VAT upgrade (return your original disk) or to new purchasers at £45.00 + VAT (please supply your CP/M serial number). Versions of DISKPEN will shortly be available for the Mimi G801 and G802 and also Super Brain.

DRH 830114 E & O E

HENRY'S COMPUTER KIT DIVISION
Phone 01·402 6822

HENRY'S RADIO
404, Edgware Road, London W2 1ED.
Telephone orders welcome

Simply a better way to pay VISA

AMERSHAM COMPUTER CENTRE

80-BUS HARDWARE

GM811	Z80A CPU Board	£125.00
GM813	Z80A CPU + 64K RAM Board	£225.00
GM812	Z80A Video Controller Board	£125.00
GM802K	16KRAM Kit (64K Board)	£ 80.00
GM802	64K Dynamic RAM Board	£125.00
GM809	Floppy Disk Controller	£125.00
GM829	FDC/SASI Controller	£145.00
LUCAS LOGIC	Floppy Disk Controller	£145.00
GM803K	Eprom/Rom Board (Kit)	£ 55.00
GM816	I/O Board (RTC CTC 3*PIO)	£125.00
EV814	IEEE 488 Controller Board	£140.00
RB32KC	32K Cmos Batt/Backed RAM	£170.00
I0824	8ch - 8bit A/D Board	£120.00
PLUTO	High Res Colour Graphics board, 2 Screen Memorie (640h*288v) 3 bit Pixels	£339.00
BABY PLUTO	(320h*288v) 3 bit Pixels	£299.00
CC837	Colour Graphics Board, (256*256)16 colour pixel (PAL) display. (PAL + RGB)	£199.00 £220.00
NAS AVC	Colour Graphics Board 80col (320h*256v) Pixels	£185.00
NAS I/O	I/O Kit Not Populated	£ 45.00
GM810	8 Slot Motherboard 5A PSU	£ 69.50
GM806AK	Nascom buffer+mother board	£ 49.50

DISK SYSTEMS

GM825-1S	Single Drive Disk Unit	£350.00
GM825-2S	Double Drive Disk Unit	£575.00
LUCAS LOGIC	Single Drive Unit (inc FDC)	£470.00
LUCAS LOGIC	Double Drive Unit (inc FDC)	£685.00
GM835	5.4 Meg Winchester Sub-System	£1450.00

DISK OPERATING SYSTEMS

GM512	CP/M for Multiboard & GM815	£ 90.00
GM532	CP/M for Multiboard & GM825	£ 90.00
GM513	CP/M for Nascom & GM809/GM815	£100.00
LUCUS LOGIC	CP/M for LUCUS Disk System	£100.00
GM515	POLYDOS 1 for Nascom & GM805	£ 90.00
GM516	POLYDOS 2 for Nascom & GM815	£ 90.00
GM533	POLYDOS 3 for Nascom & Lucas	£ 90.00
GM534	POLYDOS 4 for Nascom & GM825	£ 90.00
LUCUS LOGIC	NASDOS for Nascom & Lucas	£ 60.00
LEVEL 9	Q-DOS for Nascom & GM805	£ 40.00

MONITORS

9"AVT	High Res, Green or Amber	£ 99.00
12"P12	High Res, Green or Amber	£110.00
12"M12	Metal cased,90 Deg,High Res	£150.00

COLOUR MONITORS Please ring for details

PRINTERS

EPSON		
MX80T/3	Dot/Mat/Tractor	£349.00
MX80FT/3	Dot/Mat/Friction/Trac	£389.00
MX100T/3	132col (as above)	£499.00
NEC		
PC8023BC	Dot/Mat/Tractor	£215.00
SMITH CORONA		
TP1	Daisywheel/12cps	£485.00

NASCOM SOFTWARE TAPES

LEVEL 9	Extension Basic Tape	£ 15.00
CCsoft	Nas-Graphpac Tape	£ 20.00
NASCOM	Pascal Compiler Tape	£ 45.00
SIGMA	Zeap 2.0 Assembler Tape	£ 20.00
GEMINI	Nas-Dis/Debug Tape	£ 20.00
L-Soft	Logic soft Relocator Tape	£ 13.00
MATHS PACKS	Mike's Basic Expander	£ 9.95
	Double Precision Package	£ 13.00
	Program Handler	£ 9.95

LEVEL 9 NASCOM GAMES TAPES

5 GAMES (Gunner/Wumpus etc)	£ 6.00
Missile Defence	£ 8.00
Asteroids	£ 8.00
Space Invasion	£ 7.00
Bomber	£ 5.00
Fantasy	£ 6.00
Nightmare Pork	£ 5.00
Colossal Adventure (32K)	£ 10.00
Adventure Quest (16K)	£ 7.90
Galaxy Invaders	£ 5.90
Life	£ 5.90
Super Gulp (requires 32K & Extension Basic)	£ 5.00

NASCOM FIRMWARE

LEVEL 9	Extension Basic (4*2708)	£ 25.00
NASCOM	Pascal Compiler	£ 75.00
GEMINI	Naspen Text Editor	£ 20.00
GEMINI	Nas-Sys 3 (2708/Nascom 1)	£ 20.00
GEMINI	Nas-Sys 3 (2716/Nascom 2)	£ 20.00
GEMINI	Imprint (For Imp Printer)	£ 20.00
BITS & PC's	Programmers Aid (N/Sys1)	£ 20.00
BITS & PC's	Programmers Aid (N/Sys3)	£ 20.00

GEMINI RPM/CPM SOFTWARE

MICROSOFT	Basic Interpreter	£195.00
COMAL 80	Structured Basic	£100.00
*GEM PEN	Text Editor / Formatter	£ 45.00
*GEM ZAP	Assembler (screen editing)	£ 45.00
*GEM DEBUG	Debug/Disassembler	£ 30.00
COPY SB	Superbrain to Gemini (DDDS)	£ 30.00
LIST/REPAIR	Recovers Lost Data Etc	£ 25.00
DATAFLOW	Information Processor	£125.00
*G BASIC	Graphics Basic (IVC)	£ 25.00
GEM GRAPHPAC	Links to Mbasic (IVC)	£ 35.00
COM-PAS	Pascal Generates M/Code	£120.00

*Tape also available.
When ordering disks please specify the format.

MEDIA

		EACH	BOX 10
SCOTCH C10	Cassettes	£ .60	£ 6.00
SCOTCH C30	Cassettes	£ .70	£ 7.00
DYSAN 1042D	D/S disk	£5.00	£ 43.00
DYSAN 1041D	S/S disk	£4.75	£ 40.00

80-BUS SYSTEMS

QUANTUM	QM 2000 System (2.4M Bytes)	£2250.00
GM903	GALAXY 2 System,2 Drives	£1495.00
GM904	GALAXY 2 System,1 Drive	£1275.00
GM905	GALAXY 2/2 drives(1.6M Bytes)	£1695.00
NASCOM 3	48K with Nas Sys 3 & Graphics	£ 549.00
NASCOM 2	Built & Tested (No user ram)	£ 285.00

WE ARE OPEN MONDAY TO SATURDAY 9-5.30. PERSONAL CALLERS WELCOME.

AMERSHAM COMPUTER CENTRE LTD.
18, WOODSIDE ROAD, AMERSHAM, BUCKS. HP7 OBH
Telephone: 02403 22307 Telex: 837788

ORDERS ACCEPTED BY MAIL AND PHONE.
ACCESS AND VISA CARD HOLDERS WELCOME.
Prices subject to the addition of relevant VAT charge + P&P