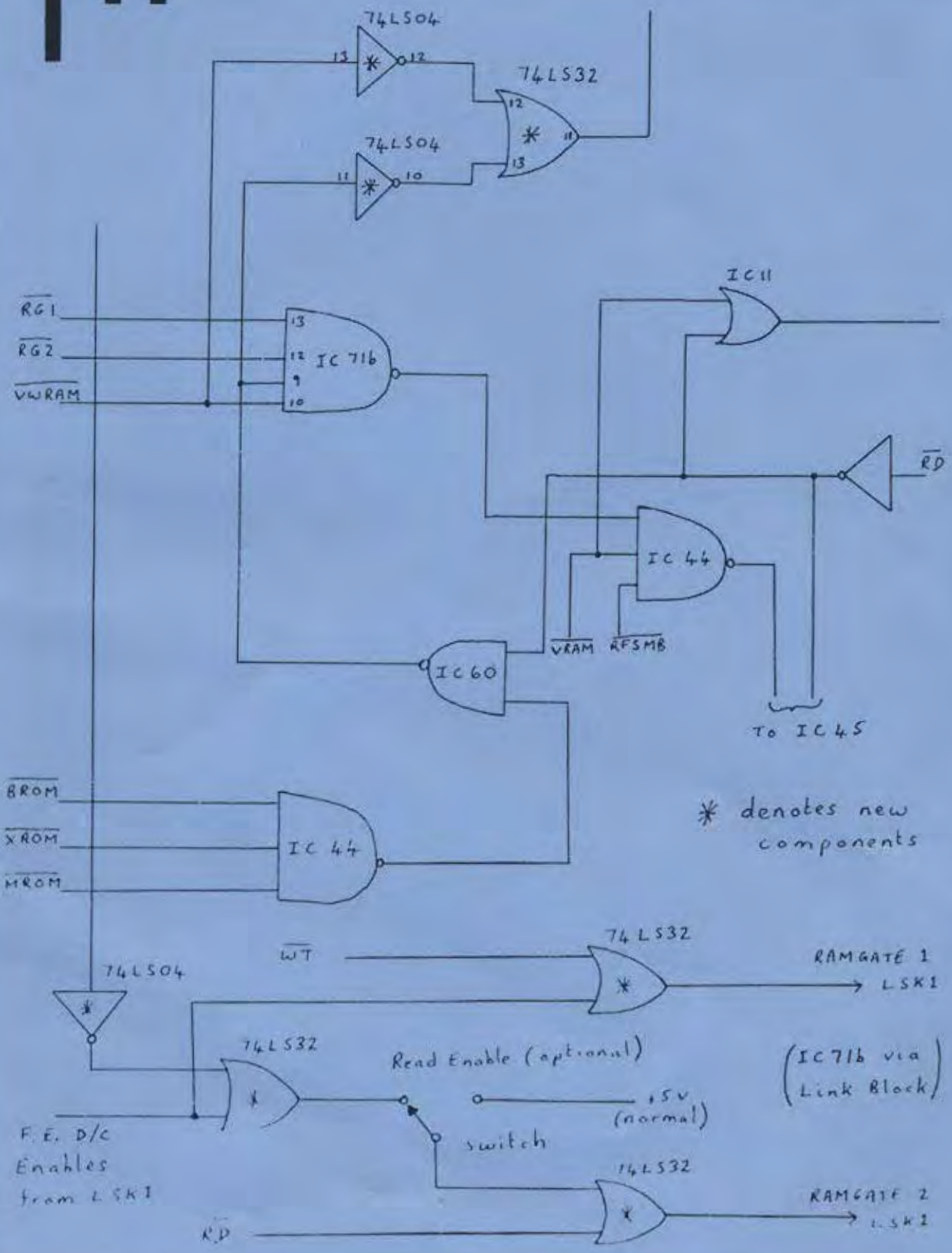
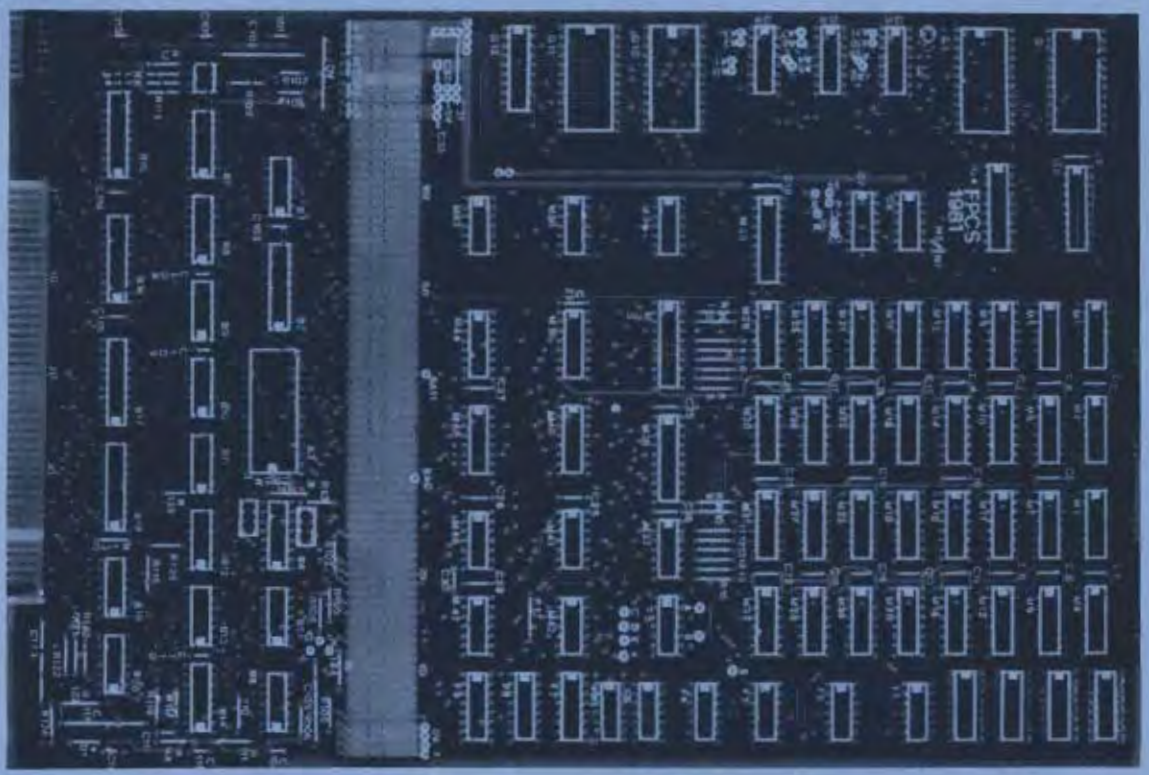
MICROPOWER VOLUME 2, NUMBER 3

A MAGAZINE FOR NASCOM USERS



JULY, 1982 £1.00



64 KILOBYTE RAM
and
BUFFER CARD
with PROGRAMMABLE GRAPHICS

This 64K RAM card is suitable for the Nascom 1 or 2. The double sided glass-fibre P.C.B., 302 mms (12 ins.) by 203 mms (8 ins.), holds up to 4 blocks of 16 Kb dynamic RAM (4116). When all four blocks are fitted the whole of the Z80 address field is occupied by RAM. The on board mapper allows parts of this address field to be selectively inhibited in either read or write mode, or both. The mapper divides the address field into 4K blocks, and any two selected blocks can be further subdivided into 2 x 2K blocks.

The graphics section is entirely separate from the dynamic RAM, but it can be mapped in at any chosen 2K boundary. It can use an EPROM (2716) to give a pre-programmed character set, or static RAM (2 x 4118, or 6116) to provide user-programmable characters.

For the Nascom 2 the memory and graphics section can be separated from the "buffer" section; the resulting 8 x 8 card can be plugged into a standard Nasbus (80-bus) edge connector For the Nascom 1 the bottom 8 x 4 ins. section of the card provides full buffering between the Nascom 1 43-way connector and Nasbus. In addition the following extra facilities are also provided:-

1. Power-on jump; this allows the processor to execute a program at any preset 4K boundary on power-on or reset.
2. Synchronised Reset; the reset pulse is synchronised with the processor M1 cycles, to prevent corruption of data in dynamic RAM
3. Wait state generator; one wait state can be added to memory or input/output access
4. ROM socket; a 28 pin or 24 pin socket can be placed at position B3, and via a series of links this can accommodate a 2716, 2732, 2764 or the standard Nascom Basic ROM
5. Input/output; a partial decode is provided which allows for 64 input/output addresses.

The 64K RAM card is available now, price £39.50, from

MICRO POWER Ltd.,
8/8A, Regent Street, Leeds LS7 4PE
Tel. (0532) 683186
Please add 55p p/p and V.A.T. at 15%.

CONTENTS

EDITORIAL

First, an apology for the late appearance of this issue. It should have had JUNE '82 on the front cover but, as it is JULY already, this wouldn't have been very accurate. Slapped wrist. Must try harder.

The second point I'd like to make is about our circulation level. We are doing very well at the moment but, ever ambitious, we would like to do better. If you have ever considered getting a subscription to MicroPower, now is the time to act. A subscription actually works out cheaper than buying the issues individually and it also means that you are guaranteed a copy of each issue. To assure you that the quality of the magazine won't be deteriorating in the future, articles that will definitely be appearing in the next couple of issues are reviews of the Hobbit cassette system, the 32K Cmos, battery-backed RAM card and the 64K RAM card with PCG. Hardware mods that we have include one to add 32K of paged EPROM to the N2 main board. We also have a selection of short programs and helpful hints that we hope to publish.

Final plea: we still need more articles, letters and comments to help us keep the magazine going. We are rapidly running dry of ideas to fill the magazine. If YOU send us a decent amount of stuff in the next couple of weeks (and ever after) it means that I won't have to plead with you in the next editorial. We do pay you for everything that we publish you know.

Happy Reading IJC

THE FRENCH CONNECTION

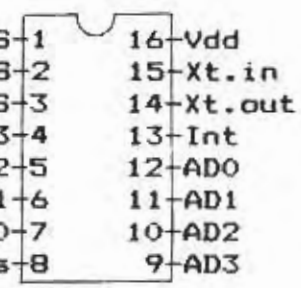
by G.R.Kirby

Quelle heure est il? Ever wanted to ask your friendly Nascom that question? Well with 4 integrated circuits and a few components it is possible to build a "real-time" clock for your Nascom using the National Semiconductor chip MM 58174.

The MM 58174.

The MM 58174 is a 16 pin device containing 16 internal registers programmed to give the time in tenths of a second, day of the week, day of the month and the month of the year. It does not however give the year, although, by telling it during initialization the year with respect to the last leap year, it will automatically keep the month of Febuary up to date (ie. 28 or 29 days). The circuit is controlled by its own internal oscillator, driven by an external quartz of 32768 KHz which I believe is a standard frequency for electronic time pieces (therefore, easy to obtain). The registers are addressed by a 4 bit bus (BCD). The following table gives the purpose and address of each register and also the pin connections.

REGISTER	AD3	AD2	AD1	AD0	MODE
R0 Mode Test	0	0	0	0	(W)rite
R1 Tenths of seconds	0	0	0	1	(R)ead
R2 Seconds (units)	0	0	1	0	R
R3 Seconds (tens)	0	0	1	1	R
R4 Minutes (units)	0	1	0	0	R/W
R5 Minutes (tens)	0	1	0	1	R/W
R6 Hours (units)	0	1	1	0	R/W
R7 Hours (tens)	0	1	1	1	R/W
R8 Days (units)	1	0	0	0	R/W
R9 Days (tens)	1	0	0	1	R/W
R10 Day of the week	1	0	1	0	R/W
R11 Month (units)	1	0	1	1	R/W
R12 Month (tens)	1	1	0	0	R/W
R13 Year 0-3 basis	1	1	0	1	W
R14 Stop/Go	1	1	1	0	W
R15 Interrupt	1	1	1	1	R/W



The power requirement of the MM 58174 is a single 5V supply, there is however a standby mode where voltage as low as 2.2V is sufficient to keep the clock running and the time up to date. In this mode it is not possible to read or write to the internal registers. Power consumption in "stand-by" mode is less than 10uA. There are, as you will realise, many possible uses for this powerful circuit. The following is a "real-time" clock using 2 Ni-Cad batteries to keep the clock running during periods when your Nascom is switched off.

The Circuit Diagram

As you can see, relatively few components are needed to build this "real-time" clock, the principal of course being the MM 58174 from National Semiconductor which is selected by the address decoder IC 2. IC 2 is a 1-in-8 decoder being low when

A6=1, A5=0, A4=0 and A7=0, giving access to the 16 registers from address 40H to 4FH (hex). The data-out, in BCD form, is present on the data lines D0-D3. R/W signals are directly connected to the control BUS of your Nascom.

IC 3 is only required by the "sporty" Nascoms running at 4 Mhz. (You mean all Nascoms don't run at 4 Mhz??-Ed). It provides a Wait signal of approxiamately 1uS to allow the MM 58174 time to output data to the Data Bus. The DBDR signal is not needed by the Nascom 2's and I am not too sure if it is in fact needed by all versions of Nascom 1's. (see Nascom Doc)

As previously mentioned, 2 Ni-Cad batteries are used to keep power on during computer "days off". These are kept charged via T1 and R1 when 5V is applied to the circuit. In "stand-by" mode, the diode D1 ensures that the 2.2V powers the MM 58174. It is necessary, when 5V operations cease, that the /WR, pin 3, is kept at a sufficiently high potential to stop any random writing to the MM 58174 registers. R4 provides this potential and the 2 NOR gates, connected in series, provide the necessary isolation from the computers Control Bus. Finally, the variable capacitor VC1 allows accurate callibration.

List of Components

R1 15 Kohm	R2 1 Kohm	R3 4.7 Kohm
R4 1 Kohm	R5 240 ohm	R6 500 ohm
R7 1 Kohm	R8 1 Kohm	
C1 68 pF	C2 100uF	VC1 50 pF
T1 2N2222	T2 2N2369	D1 1N4148
X1 Quartz 32,768 Khz		
IC 1 MM 58174	IC 2 74LS138	IC 3 74121
IC 4 7433		
Batt. 2 rechargeable Ni-Cad batteries		

Construction

I think anyone attempting to build the clock will already have sufficient skill to work out a simple Vero board layout. In view of the small number of components, it should not be difficult. Myself, I used a 10cm x 12cm wire wrap board. There was enough space for all the components and the Ni-Cad batteries housed in a plastic container (transistor radio type). As I have a "home built" case for my Nascom, I found no difficulty installing the board, wiring it directly into the Nascom Bus.

NasBus Connections

NAS-BUS PIN	SIGNAL	NAS-BUS PIN	SIGNAL
1	Ground	13	/DBDR(N1s only)
23	Wait(4Mhz only)	26	/IORQ
29	/Read	30	A0
31	A1	32	A2
33	A3	34	A4
35	A5	36	A6
37	A7	50	D0
51	D1	52	D2
53	D3	78	+5V

Software

If the unit has been built according to the circuit diagram, you will find that it is addressable at ports 64 to 79 (don't forget to switch to external port addressing). I enclose 2 demonstration programs written in Microsoft 80. The first is called to initialise the clock. Lines 100-190 input the date and time, and control the validity. Line 230 sets all bits of register 14 to 0 thus stopping the clock. Lines 220-380 set up the date and time in registers R4-R13. Note, I don't use register 10, day of the week. Lines 330-370 set up the correct value for the Leap year. Note, the astuce for detecting the year on a 0 to 3 basis in line 330.

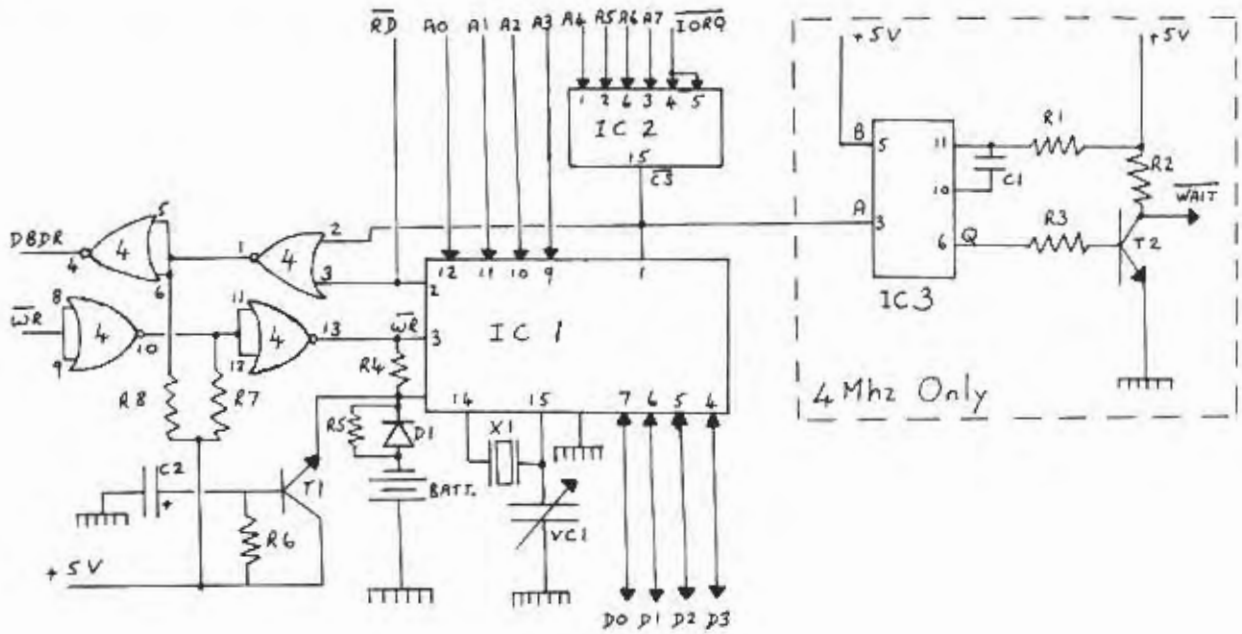
0=Leap	year		therefore set	register	13 to 8
1= "	"	+1	"	"	13 to 4
2= "	"	+2	"	"	13 to 2
3= "	"	+3	"	"	13 to 1

Finally, when all registers have been initialised, it remains to start the clock by sending a 1 to register 14.

The second program is a demonstration of how the clock can be read, once set up. Lines 520, 530 and 560 read data from the 11 registers, starting at register 2 (seconds (units)). Line 540 checks if data read is valid ie. that a value was present at the Data Bus during the address period. Line 550 leaves the BCD value. Lines 580-660 are just one example of displaying the date and time.

Conclusion

As you will no doubt appreciate, there are many possible uses for this clock, from simple time telling, although it does make rather an expensive clock, to sophisticated timing requirements in the case of automatic control functions. What ever your choice, you will always have the time!



```
10 '*****************************************************
20 '*                    DEMONSTRATION                    *
30 '* CLOCK SET-UP PROGRAM   G.R. KIRBY   MAY 82   *
40 '*****************************************************
60 PRINT CHR$(26)    :REM CLEAR SCREEN ROUTINE
70 PORT=64           :REM ADDRESS OF MM 58174
80 PRINT "To set up internal clock enter data as requested,"
90 PRINT "to start clock, hit Return."
110 INPUT "Enter date eg. 06/05/82 ";D$
120 IF LEN(D$)<8 OR LEN(D$)>8 THEN 110
130 IF VAL(LEFT$(D$,2))>31 THEN 110
140 IF VAL(MID$(D$,4,2))>12 THEN 110
150 REM INPUT TIME AND CHECK IF VALID
160 INPUT "Enter time eg. 16:59 ";T$
170 IF LEN(T$)<5 OR LEN(T$)>5 THEN 160
180 IF VAL(LEFT$(T$,2))>24 THEN 160
190 IF VAL(MID$(T$,4,2))>59 THEN 160
210 REM SET TIME AND DATE
220 OUT PORT,0      :REM INITIALISE
230 OUT PORT+14,0   :REM STOP CLOCK
240 OUT PORT+4,VAL(MID$(T$,5,1))   :REM MINUTES
250 OUT PORT+5,VAL(MID$(T$,4,1))   :REM TENS OF MINUTES
260 OUT PORT+6,VAL(MID$(T$,2,1))   :REM HOURS
270 OUT PORT+7,VAL(LEFT$(T$,1))    :REM TENS OF HOURS
280 OUT PORT+8,VAL(MID$(D$,2,1))   :REM DAYS
290 OUT PORT+9,VAL(LEFT$(D$,1))    :REM TENS OF DAYS
300 OUT PORT+11,VAL(MID$(D$,5,1)) :REM MONTH
310 OUT PORT+12,VAL(MID$(D$,4,1)) :REM TENS OF MONTHS
320 REM CHECK FOR LEAP YEAR AND SET ACCORDINGLY
330 X=VAL(MID$(D$,7,2)) AND 3    :REM LEAVE YEAR ON A 0-4 BASIS
340 IF X=0 THEN OUT PORT+13,8
350 IF X=1 THEN OUT PORT+13,4
360 IF X=2 THEN OUT PORT+13,2
370 IF X=3 THEN OUT PORT+13,1
390 REM CLOCK START ROUTINE
400 INPUT "Hit Return to start clock ";X
410 OUT PORT+14,1
440 '*****************************************************
450 '*   READ CLOCK PROGRAM   G.R. KIRBY   MAY 82   *
460 '*****************************************************
470 PRINT CHR$(26)    :REM CLEAR SCREEN
480 PORT=64+2         :REM ADDRESS OF MM 58174 SECONDS REGISTER
490 DIM D(12)
500 INPUT "Enter year eg. 82 ";Y$
510 PRINT CHR$(26)        :REM CLS
520 FOR X=0 TO 10
530 D(X)=INP(PORT+X)      :REM READ PORT
540 IF D(X)=255 THEN 530 :REM IS IT VALID?
550 D(X)=D(X) AND 15      :REM STRIP 4 MSB's
560 NEXT X
580 REM DISPLAY DATE AND TIME
590 PRINT D(7)*10+D(6);"/";    :REM DAY
600 PRINT D(10)*10+D(9);"/";   :REM MONTH
610 PRINT Y$;"  ";             :REM YEAR
620 PRINT D(5)*10+D(4);":";    :REM HOURS
630 PRINT D(3)*10+D(2);":";    :REM MINUTES
640 PRINT D(1)*10+D(0);        :REM SECONDS
650 PRINT CHR$(30)             :REM MOVE CURSOR UP ONE LINE
660 GOTO 520                   :REM UPDATE DISPLAY
```

PolyDos

T.I.J.Toler

Although I have had my Nascom 2 for over 2 years, I must declare that I am still a beginner and this article is written for those who, like me, find they have much to learn.

After having played about with the usual gimicks - "Space Invaders", "Stock Car Racing", drawing lines on the screen, etc., I decided to make the machine earn its keep or at least do something useful. My daughter, a professional photographer, wanted a system to keep tabs on her, literally, thousands of photographs, so, I decided to invest in a disc system. Having read up all the articles on discs and DOS, and all the adverts, off I went to the local Nascom dealer who sold me a Pertec, single density, FD250 dis c drive and a G809 Gemini controller card.

The question then was - What operating system? - CP/M or something else? The "something else" was PolyDos which the dealer said would enable me to use my own programs as well as everything else. So, for £80 plus VAT I got my PolyDos. It consists of 2 EPROMs, a 5 1/2 inch floppy with the controlling programs and a manual.

Well, home I went with my purchases. First, to connect the drive. The G809 is an 80-bus card. When my Nascom was built we hadn't allowed for an extension of the bus. However, fortunately, the pins on the main board edge connector had not been cropped and were long enough to allow a second piece of Vero board to be fitted. So with a bit of engineering, the G809 card was fitted and the ribbon cable from the Pertec drive plugged in.

Next, the PolyDos EPROMs had to be fitted. The manual gives several options for this. I used the Nascom 2 main board block B. On LKB6, you connect pins 8 to 12, 7 to 11, 6 to 10 and 5 to 9. Then on LKS1, the manual says "connect pins 6 to 7 for XROM and 6 to 10 for D000-DFFF. This rather threw me and I probably connected both. However, I inserted the EPROMs, PD2A in socket B5 and PD2B in socket B6. Power up and "Hey Presto", up comes "Boot which drive ?". (No one told me what "boot" meant!!)

Now the next stage is to hit "0" - which I did. Nothing happened, no joy on the disc drive - all dead as a doornail. So, off to the dealer with my equipment. Now, unfortunately, I don't know exactly what he did but I think he altered the connections on LKS1. They are now as follows:-

1 to 16
2 to 15
4, 7 and 10 all together
8 to 9

I suspect he also made some alterations to the RAM board as, somehow, you have to have some RAM available at C000H. However, it worked! Enter DIR;ELD, hit Enter and up comes the list of files on the master disc.

Now you HAVE to get down to reading the manual (several times) and it is as well to make a copy of the master disc, just in case of accidents!

First, you have to FORMAT an empty disc. This quite simple. Enter "FORMAT", hit enter and follow the instructions as they are displayed. Put the master disc back, enter "BACKUP", hit Enter and again follow the instructions. I then started entering my machine code files. They went in fine. The speed with which they come out is amazing. "Space Invaders" loads in about 2 seconds flat.

However, now my troubles starts, PolyDos supports a very useful disc basic system but when I tried to use it by entering "BASIC", I got a series of "error"s flashing and nothing else. So, once again to the dealer who now said that I needed 48K of RAM and not the 32 which I had. Nothing for it but to shell out for a new RAM board with 8 new 4116s and add the 16 4116s I already had. (One 32K RAM board going begging but without the 4116s - however, I might incorporate it later)

Well, at last with the new RAM board installed (I didn't even do the memory test) it all worked. PolyDos Disc Basic came up with "38851 bytes free". My BASIC programs could now be entered and run. Much more important for me were the commands to set up Sequential and Random files, but here the manual failed me somewhat

For a Random Access File, it gives an example of a format descriptor string - "I%"+CHR$(36)+"S"+CHR$(48) - indicating that each record consists of an integer and 2 strings of maximum length 32 and 48. As this occupies more than one line, how can one enter it or get it to scroll over? I hope some one will tell me it is just too simple and show me how to do it.

My other problem was with POLYZAP - the assembler program. Here you have to enter a "Source File Specifier" but the manual doesn't tell you (or, at least, I couldn't find it) how to start. However, the answer is:-

Start with a file name eg. TEST.TX
Enter EDIT TEST.TX to which the monitor responds "Press space to continue"
When you do this, the screen clears and a flashing cursor appears in the top left hand corner
Enter your program in nmemonics to be assembled, editting it as you go along
You can then assemble it by entering PZAP TEST.TX

Now, if all is well, (and it generally isn't) "PZAP" will assemble the program and output the results to the screen or the printer. If, as generally happens, there are errors, it will tell you what they are and you can start again. Now, I may not be doing it correctly and it has certainly caused me a lot of hassle, but, I am told, it is a very powerful assembler and you can throw away your ZEAP.

All in all I am really very pleased with PolyDos. It has tremendous potential and really makes the Nascom a production machine. I should be very interested to hear from any other Nascom user who has experimented with it.

Snowdinger 2

by G.J.Davies

In the first issue of Micropower (you did get your copy of the first issue??-Ed), Snowdinger, a circuit to effectively stop on-screen disturbances caused by indiscriminate processor access of the VDU RAM, was published. The circuit had one major disadvantage, it was for a Nascom 1 and my own computer is a Nascom 2. Hence after reading the article, the project was forgotten. Now having just finished my exams at college, I was looking for something to usefully occupy my time with. Since mowing the lawn was completely out of the question, I decided to look at the problem of converting Snowdinger into Snowdinger 2. This article is the result of some two and a half days solid work.

For those who either didn't get the first Micropower or have forgotten the theory behind Snowdinger, the basic approach is as follows:
Each line of the TV picture is 64 microseconds long and in every 64 microseconds, 16 are blanked (line blanking), thus if 64 characters are used for one display line, each will take up a time slot of 1 microsecond. Also 16 of the 64 will be lost due to the line blanking, leaving 48 characters to be displayed fully. Hence Nascom screens are 48 characters wide. For every character on a TV line there are 8 possible dots to be displayed, because each character cell is an 8*16 dot matrix, 16 TV lines are required to produce one row of 48 complete character cells. The VDU circuitry places an eight bit byte of data in the serialiser once per microsecond, where upon it is sent to the TV one bit at a time. When it is empty the next byte of data is loaded and the process goes on repeatedly for all 256 lines of the TV picture. If this period of 1 microsecond is split into two seperate halves, one half may be used to send the necessary data to the serialiser and the other half may be used by the Z80 to access the VDU RAM. Using hardware to synchronise the processor to these time slots of 500ns, no screen disturbance will occur.

The theory sounds simple enough, putting it into practice is however a little harder. The first thing that I noticed about Snowdingers circuit was that it required slight changes to work at 4 Mhz. If this could be avoided in Snowdinger 2, then the switches on the Nascom 2 could be used as normal. By drawing out all the clock waveforms on graph paper and looking in the Z80 technical manual at the timing for the read and write cycles, I soon discovered why Snowdinger needed circuit changes to work at 4 Mhz. Wait states are sampled on the falling edge of the 2nd clock period (T2) of both the read and the write cycle. Since there are twice as many falling edges in 4Mhz. a 500ns time slot has within it, two falling edges of the 4 Mhz clock, compared to one falling edge of the 2 Mhz clock. When no wait state is found, the following negetive edge of the clock terminates the cycle (whether read or write). In the case of the 2 Mhz clock, this coincides with the access time slot, but for 4 Mhz the read or write cycle will be terminated before the access time slot occurs (see diagrams for details of the waveforms).

I was a little concerned about the problem of access time for the VDU RAM so the next step was to make absolutely sure that this problem was minimised. The most critical cycle, timing-wise, is the read cycle because data is read in on the falling edge of the last clock of the cycle (T3). With the write cycle, data is valid rom the falling edge of T1 until slightly afterthe rising edge of T3 and may be written anywhere between the falling edge of T2 and the falling edge of T3. Hence as long as the falling edge of the T3 clock is before the rising edge of the access slot, there is less chance of a bad read. At 2 Mhz this is all well and good as the falling edge of the T3 clock can be placed about 375ns into the access period. This gives an access time somewhere in the region of 375 to 400ns. With the 4 Mhz clock, things begin to look dodgy because, with the T2 clock falling edge just on the falling edge of the access slot, there is only 250ns before the falling edge of T3. So access time to the VDU RAM must be better than 250ns in order to allow time for the address bus multiplexors to change the address bus over and let the RAM settle down. By placing the falling edge of the 4 Mhz on the falling edge of the access period, you are relying on the delay time of the gates to maintain a high output long enough to terminate the wait state. This is bad practice as the main clock only needs to be delayed by a slow gate and the wait state may not be terminated.

Now it became obvious that Snowdinger 2 would be completely different to Snowdinger, due mainly to using the same circuit for both 2 and 4 Mhz. Using the 1 Mhz clock ORed with VRAM a gated signal is produced for any access to the VDU RAM, ie VRAM is synchronised to the 1 Mhz clock. Finding a wait signal for the Z80 that would work at 2 and 4 Mhz was some what more difficult. However the final waveform is the result of ANDing the 2 Mhz with the 4 Mhz clock. For the Z80 to use this wait signal, both the clocks need to have there phases changed in order to sample the wait gate at the correct times. These phase changes are accomplished as follows:

1) The 2 Mhz clock is shifted back quarter of it's cycle by exclusive ORing it with the 4 Mhz clock.
2) The 4 Mhz clock is shifted back quarter of it's cycle by exclusive ORing it with the 8 Mhz clock.

The reason for shifting the 4 Mhz clock is, so as to keep it's falling edge away from the falling edge of the wait gate signal, hence avoiding any static hazards from slow gates, etc. This has the disadvantage of reducing the access time for the RAM by about 62.5ns, but makes the circuit more reliable. Figure 1 shows all of the waveforms needed to construct the circuit, however I soon discovered that this circuit will only work for a Nascom 1. A quick look with a 100 Mhz oscilloscope confirmed that Nascom 2s have different phase clocks. Figure 2 shows diagrams of the 4,2,1 and the serialiser load clocks for a Nascom 2.

The Nascom 2 uses a 74LS193, presettable 4-bit binary up/down counter. This circuits clock is positive edge triggered, hence all the clock outputs have different phases. At first sight this presents no problems, apply the theory to the new waveforms and design a new circuit. Unfortunately, changing the phases of the clocks also changes the position of the load

clock. This means that a new access time slot must be decided upon, because which ever half of the 1 Mhz clock is used, it will always interfere with the load clock. The new access slot must be between load clock pulses, but not too close to the start of a load clock pulse so as to interfere with video access timing to the RAM. Figure 2 shows the new access signal, this is produced by exclusive ORing the 1 and 2 Mhz then inverting the resultant signal. The wait gate is produced as before and the phases of the 2 and 4 Mhz clocks altered to suit this using the theory previously described. Again all the waveforms are shown in figure 2 so that they may be compared to the original clocks etc.

Figure 3 shows the circuit diagram, this is fairly straight forward and should present few problems to the constructor. The final connection to the Nascom is somewhat more involved but, providing that the instructions are followed carefully, there should be no real problems.

Construction.

In addition to the 4 integrated circuits used, IC 49 must be relocated on the Snowdinger 2 board. The socket for IC 49 must be of the wire-wrap type, with pins 2 and 6 cut short. This is so that the whole board can plug into Nascoms socket for IC 49, with the new 2 and 4 Mhz clocks connected, via wires, to pins 6 and 2 respectively. Any connections made to IC 49 should then be made to the wire wrap socket. The prototype used wire wrap sockets for all the connections but any method of construction may be used. Follow the circuit diagram carefully and check all your work as you go along.

Connection.

The board should have 6 wires for connection to the main board of the Nascom. In the circuit diagram of figure 3 these are marked 1 to 6. Using the following list make all your connections.

1) Remove IC 58 completely from the main board.

2) Remove IC 49 and place it on the Snowdinger 2, taking care to have it correctly orientated.

3) Remove IC 71, bend pins 2 and 4 outwards and then replace it.

4) Remove IC 69, bend pin 3 outwards and then replace it.

5) Take wire 1 and push it into pin 2 of IC 49's socket.

6) Take wire 2 and push it into pin 6 of IC 49's socket.

7) Solder wire 3 to pin 3 of IC 69.

8) Take wire 4 and push it into pin 9 of IC 58's socket.

9) Solder wire 5 to the processor end of R9 on the main board.

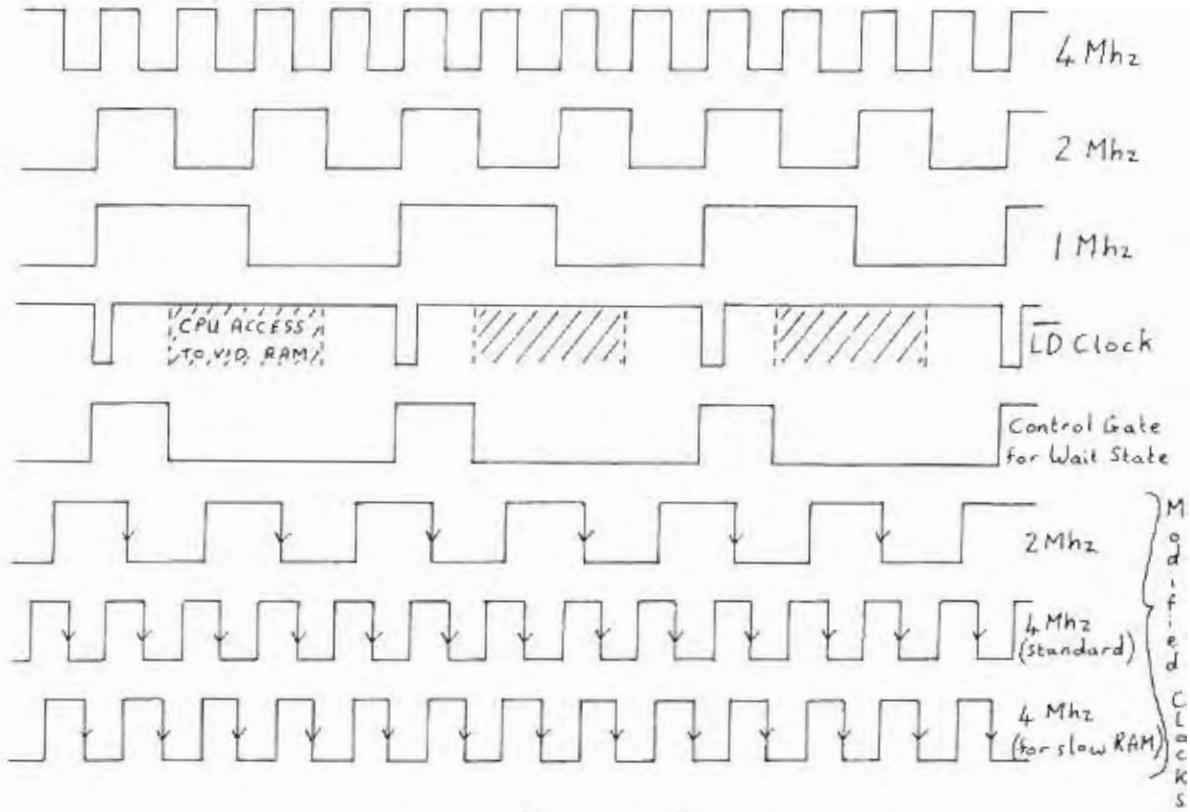
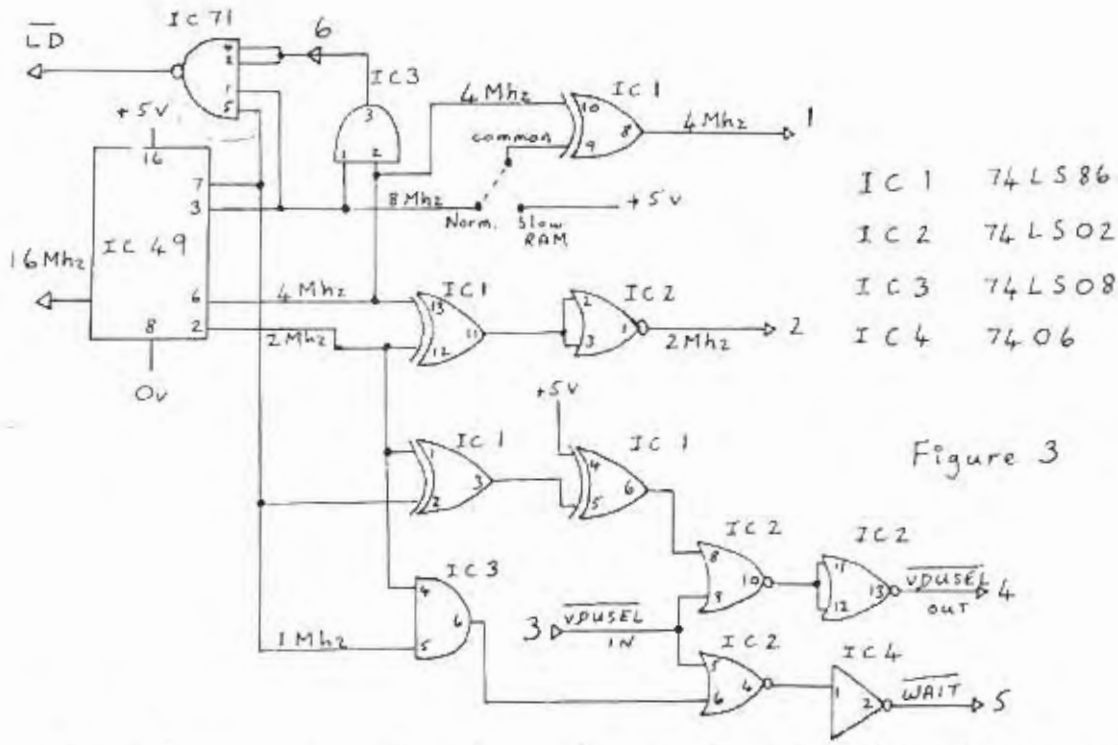


Figure 2



IC 1 74LS86
IC 2 74LS02
IC 3 74LS08
IC 4 7406

Figure 3

IC 49 is on the Snowdinger 2 board in a wire-wrap socket. Pins 2 and 6 of this socket are cut short so that, when the board is plugged into the Nascoms socket for IC 49, they do not make any connections. Instead wires 1 and 2 are placed in these pins on the main board.

10) Solder wire 6 to pins 2 and 4 of IC 71.

11) Finally place the Snowdinger 2 board into IC 49's socket using the long legs of the wire wrap socket.

Check all your connections and when you are satisfied that they are correct, try your Nascom at 2 Mhz. Test the circuit using the Tabulate command, then, if all is well, try the system at 4 Mhz. If the circuit worked well at 2 Mhz but not at 4 Mhz there are 2 possibilities:

1) There is a wrong connection on the Snowdinger.

2) The VDU RAM is too slow.

Check all your work, if no faults are found, it is likely that you have slow RAM. On the circuit diagram there is an alternative connection for one of IC 1's gates which is in the 4 Mhz signal path. Making this change will increase the access time at 4 Mhz but it will increase the risk of static hazards as mentioned previously. Alternatively you could get a faster 4118 RAM. This should be considered as a last resort as the first solution is simpler and cheaper.

Hopefully, by now your screen is clear at both 2 and 4 Mhz. Try playing Invaders or some other program that makes dynamic use of the screen, you'll wonder why you put up with screen flash for so long.

NOTE - Sheet 1 in the Nascom 2 manual shows IC 7 as an inverter. This is an error as IC 7 is an 81LS97 which is a buffer. Nascoms clock is not the inverse of that supplied by the Snowdinger 2, but a buffered version. One small alteration that can be made to improve the shape of the clock is connecting a 220 ohm resistor from pin 5 of IC 11 to +5 volts (pin 14 of IC 11).

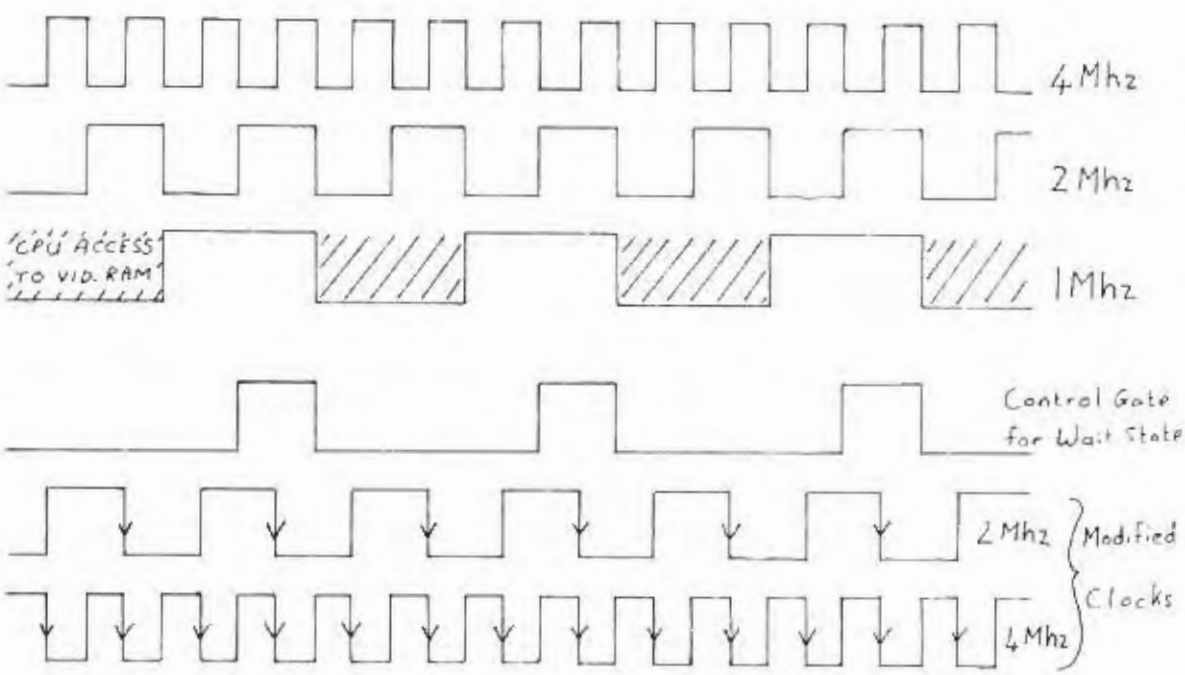


Figure 1

AN INTRODUCTION TO CP/M

by Chris Blackmore

CP/M stands for Control Program for Microprocessors. CP/M is an operating system, as the glossy magazines will tell you. They rarely say what that means though. In his "CP/M Handbook", Rodney Zaks says that the purpose of an operating system is "to execute user commands and allow the user to conveniently use all of the hardware resources provided by the computer".

Most readers of Micropower will already be familiar with one of the best operating systems for Z80 based computers available, Nas-Sys, which converts an otherwise useless lump of expensive electronics into a Nascom computer. When you switch a Nascom on, it automatically runs Nas-Sys which gets the system ready for you to use it, clearing the screen and setting various options, and then it waits for a command. Nas-Sys commands consist of a letter, usually followed by at least one hexadecimal number. Using these commands, you can load programs and data from tape, change and debug programs once they are loaded, save programs and data to tape, run programs and much much more. All of this is achieved using only 2K of EPROM, which is a remarkable example of compact programming.

What Nas-Sys does not do for its users (and this is not suprising, given its size!) is keep a record of where on a tape a given program or data file may be found or the name of the program or file, and it can not control a disc system without considerable extension. Its ability to control the screen and keyboard of a Nascom is, of course, superb. But what would happen if you tried to use Nas-Sys to run a TRS-80? or an RM380Z? This would be a silly thing to attempt, even though all three are Z80 based micros and will run each others BASIC programs with very little conversion work. Each machine has its own operating system, which has been designed to make the most of the available hardware.

If an operating system is to run on more than one machine, it will have to have its abilities reduced to the level of the "lowest common denominator", or use routines which have been custom written for the machine it is to run on. CP/M was originally written for use on an Intel system that used the 8080 chip. Thus, it can be used on machines that use both the 8080 and the Z80. The main bulk of CP/M is the same for all the machines it is run on, only the routines referred to as the "primitives" are custom written for the particular machine by the implementer. These routines are the ones that read the keyboard, print characters on the screen or elsewhere, and transfer data to and from disc units. In any CP/M system, there is RAM from address 0000 upwards, for as far as you can afford. At address F000H there is an EPROM or ROM containing a very short program which loads CP/M from the disc unit into the memory and then hands control over to CP/M.

This automatic loading of the operating system is called "booting the system" for reasons far too obscure to be of interest here. On most systems, it is done automatically for you, but on others it is not. To show off my worldly wisdom, I

will tell you that on a Research Machine 380Z you have to type B. CP/M is loaded into the top part of the computers RAM and when it is run it clears the screen and prints a system prompt. In some versions it also prints a system message such as "48K CP/M" before the prompt. The prompt looks like this:

```
A>
```

This means that drive A is active and that CP/M is waiting for orders.

Nas-Sys uses single letter commands and there are quite a lot of them. CP/M is far larger than Nas-Sys and more complex but has suprisingly few commands. These are usually referred to as the "built-in" commands and there are 5 of them in version 1.4 of CP/M. Strictly speaking there are more commands than this if you count the instructions that can be given to the system by pressing the Control key in combination with some of the alphabetical keys. The built-in commands are DIR, ERA, REN, SAVE and TYPE. To issue a command to CP/M you would type one of these commands usually followed by the name of a file, sometimes with parameters as well and press return. Before I go into detail about what the commands accomplish, it is first necessary to say something about file names, as these will not be familiar to the majority of Nascom users. Some machines use filenames as part of their tape handling system and one sometimes sees programs which enable the Nascom to put a name at the start of a tape and search for the name when the tape is loaded.

CP/M file names have two parts which are seperated by a full stop. The first part is the name of the file and the second part, called the extension, is used to show the type of file referred to. For example, a file called HELLO.BAS would almost certainly be a BASIC program called "HELLO". If a command is to refer to more than one file, an ambiguous file name can be specified using the characters "?" and "*". The question mark represents any single character that could appear in the file name and the star represents up to eight characters in the file name or up to three characters in the file type. For example, the ambiguous file name MARVIN.* would match with the files MARVIN.BAS and MARVIN.INT as well as MARVIN.COM, and *.BA? would match with HELLO.BAS, HELLO.BAK, DATA.BAD and many more. To put it another way, *.* means the same as ????????.??? .

The command ERA, not suprisingly, is used to erase files from the disc you are working on. If you type ERA *.BAS all the files with that extension will vanish from the disc. If you type ERA HELLO.BAS only the file with that name will be erased. And if you type ERA *.* , CP/M will ask you if you are joking because that would erase all the files on that disc.

Before you start to erase things it is necessary to know what is actually there. The command DIR enables you to do this. If you type it on its own, CP/M will list all the files on the disc. If you follow the DIR with an ambiguous file name, all the file names that match will be listed.

The name of a file can be changed using the REN command. To change the name of the file HELLO.BAS, for example, you might type REN GREETING.BAS=HELLO.BAS. The new name is always first and the old name last.

The SAVE command is used to transfer a program or data from the computers memory to disc. This command has to be told how much of the memory you want it to save, in 256 byte pages. The area saved starts at 0100H, and a typical example of the command is SAVE 5 SPLAT.COM which would copy the section of RAM from 0100H to 05FFH onto the disc and give it the file name SPLAT.COM in the directory.

The final built-in command is TYPE which types out a file. It will put it on the screen and if CP/M has been told that the printer is on, it will send it there as well.

So, how do we tell the system to send its output to the printer? This is acomplished using one of the Control key commands mentioned earlier, Control P. There are several of these commands besides Control P. Control C will (usually) reboot the system, Control H works like the backspace key, Control M is the same as the enter key and Control S stops and starts the console display so that you can read all of the directory, for example, when it would otherwise be scrolled off screen. All these commands are covered in the two introductory books that I suggest new CP/M users buy, in great detail. The books are "The CP/M Handbook" by Rodney Zaks and "Using CP/M" by Judi Fernandez and Ruth Ashley. Of the two, I prefer the first but this a matter of taste. If you like the kind of book where you are expected to fill in the answers to the questions as you go along, then the latter book is excellent too. What neither of them tells you, much to my disappointment, is how the machine code addict can make use of CP/M in his programs. The information is in the CP/M documentation. Sadly, it is terse in the extreme and does not make several important points at all clearly. In future articles I hope to give an insight into the ways in which the routines in CP/M can be used by your programs.

The description of CP/M that I have given so far does not make any mention of how to run a program. It could not be simpler, as all you do is type the name of the program and press the enter key. The system checks what you have typed and if it is not a built-in command, it looks on the disc for a file of that name with a .COM extension. Suppose you type INVADERS and press enter; there is no built-in command with this name, so CP/M looks on the disc for a file called INVADERS.COM. If it finds it, it loads the file into memory at address 0100H onwards and jumps to the start of it. A program of the kind described is called a transient program because it is not in memory all the time. Several transient programs are supplied with CP/M when you buy it, all intended to make the system easier to use. The transient programs supplied with the Gemini implementation of CP/M that I use, and doubtless supplied with all the other implementations of the system are as follows:

ASM.COM	This is an asembler, more of which later.
DDT.COM	This is a debugging aid also mentioned later.
DUMP.COM	A program for tabulating files in hex.
EBASIC.COM	A compiler for BASIC, not without limitations, as it does not support POKE and PEEK commands. Quite good file handling though.
ED.COM	A text file editor I shall malign later.
ERUN.COM	The run-time part of the BASIC compiler.

FORMAT.COM	A program used to make new discs ready for use.
LOAD.COM	A program used to convert the output of the assembler from an ASCII listing of hex into a .COM file, ready to run.
MOVCPM.COM	A program used to tailor CP/M to systems with differing amounts of memory.
PIP.COM	Used to transfer files from one device to another and to join files together. Can do an awful lot and is very useful.
STAT.COM	Not only does this one give you statistics about the size of files, it is also used to tell CP/M which devices are available for it to use.
SUBMIT.COM	A program to allow you submit a file filled with commands, which CP/M will carry out in sequence, usually described as batch mode.
SYSGEN.COM	A program used to copy CP/M itself onto a new disc.

Both Zaks and Fernandez and Ashley give good descriptions of how to use these programs and there would be little point in going over the same ground again. There are a few things the dedicated machine code "freak" like myself needs to know about some of them, however. To start with, the text editor was designed (it seems) to be used with a teletype rather than a VDU. It is a very powerful editor, but not at all easy to use, and the only valid reason for continuing to use it would be poverty. Even masochists draw the line somewhere before ED.COM! The assembler also displays its historical origins clearly, as it uses the hideous Intel mnemonics that were developed from the 8080 mnemonics when the Z80 first appeared. If you don't recognise SHLD and STA as being Z80 mnemonics, you will not find this assembler at all easy to get on with. I have no idea why it can't put its output on the disc without the help of LOAD.COM.

This information came as quite a shock to me when I started to use CP/M, as I had not seen any such critisms in the press before I bought the system. At first sight, there would appear to be considerable difficulty in store for the user who hopes to do more than just run other peoples' programs. The advantages of CP/M are such that these problems should not be considered overwhelming. The cure is a simple one, provided you can afford a little more software, after the already considerable cost of the disc system itself. Several software packages consisting of an editor, assembler and debugging aids are available - the one I use is sold by Hisoft. Their editor gives back the ability to move the cursor all over the screen, at the expense of not running on other CP/M systems, which is no problem, and the assembler recognises sensible mnemonics. The debugger has several excellent facilities absent from the standard DDT.COM , and has a better display than Nas-Sys when single stepping. It also includes a disassembler which is very useful.

In my next article I will describe how to use the CP/M input and output routines in your programs, with some examples.

16K CMOS Memory Extension for the Nascom 2 Main Board

Paul Anderson

If you, like me, are one of those people who purchased a 48K (Ram B) memory card and then kicked yourself for not waiting until the 64K (Ram C) became available and now wish you had the full 64K to run your disc system etc., all is not lost. Here are the modifications required to turn those 8 (spare) 24 pin DIL sockets on your Nascom 2 into that extra 16K.

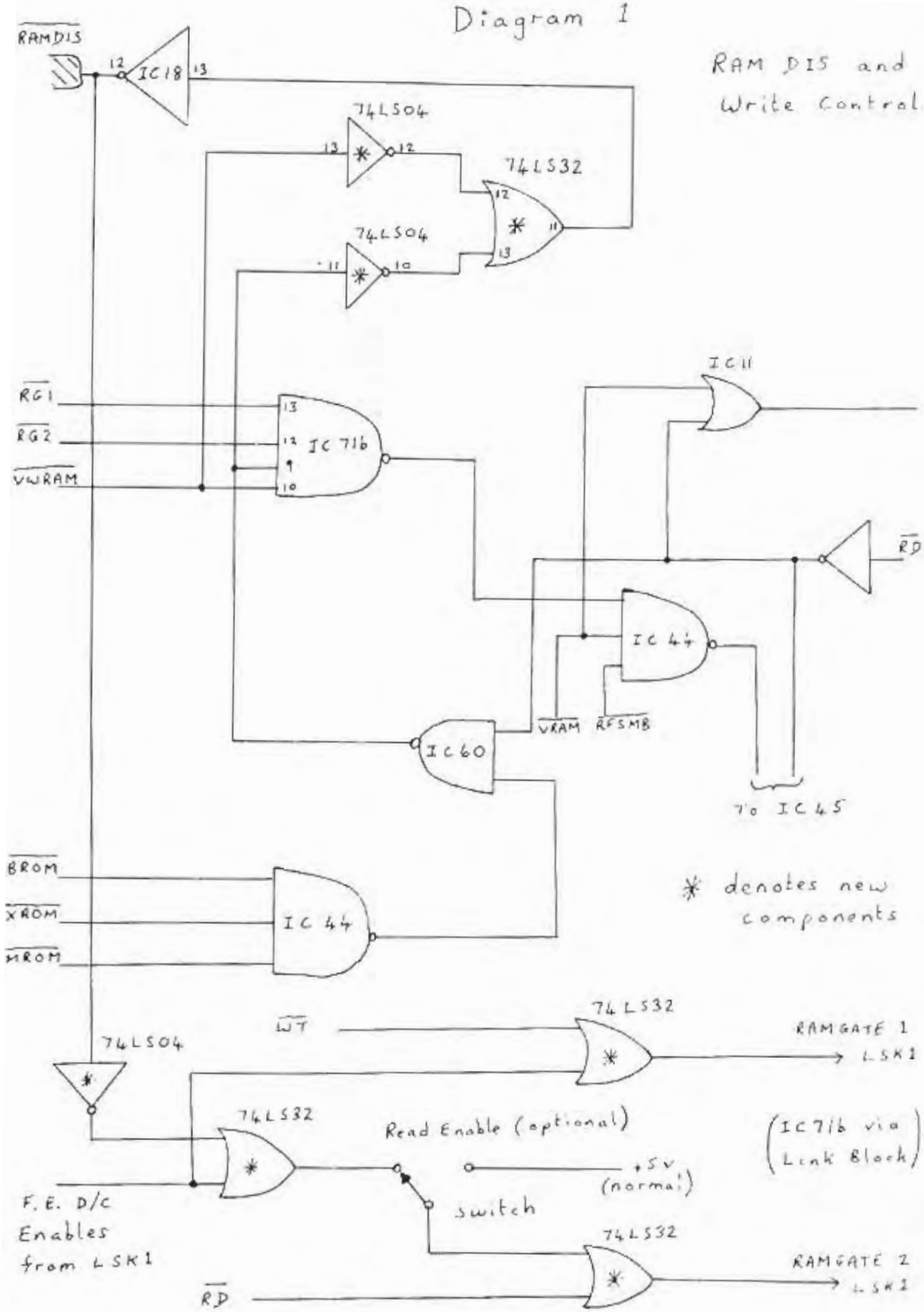
I will introduce you to an amazing little beasty called the HM6116-4, manufactured by Hitatchi, available a year ago for the princely sum of £30 (yes THIRTY POUNDS) each - have I frightened you? - but now available for around £4 (yes FOUR POUNDS) each in 10-up quantities. Each of these devices is 2K x 8 bits (16K), so you will need 8 - one for each socket!!! This will cost you about £40 including VAT. This may sound a lot but when you consider that to upgrade by that 16K you will have to sell your 48K card and buy a new 64K (anyone seen any secondhand 64K card!!), plus the cost of the extra RAM, I suspect it will cost about this amount anyway.

As I said earlier, the HM6116 is an amazing device. The fastest version will run at over 8Mhz and the low power version consumes about 0.1uA in standby mode. I used the standard version (cheapest) and out of a batch of 20 samples???, they all exceeded their quoted performance by an enormous margin, particularly in terms of Power Down (standby). For the standard version, 20uA is quoted which, for 8 chips, means 160uA which would be OK for battery backup. However, of the samples I have checked, the worst device had a standby current of only 5uA (only 3 were above 1uA) and the typical values were between 0.3uA and 0.7uA (almost two orders of magnitude inside spec. and better than that quoted for the low power version). So IF VE VERE TO POWER ZEM from one MEMPACK battery (3.6V 150mA hours), VE VOULD BE ABLE to operate them for around 15000 hours on one recharge, assuming a total consumption of less than 10uA for your 8 chips, thats approaching 2 YEARS. In practice, 1 year is about right allowing for the self-discharge of these batteries. If VE VERE to float charge the batteries at about 1 milli Amp for 4 hours a week (I'm sure your Nascom is on for more than that each week, if not there's a little mod...), then the batteries will remain virtually fully charged for the life of your Nascom. (Old Nascoms never die, they just get mod'ed-Ed)

Now, battery backed up RAM opens up many interesting possibilities. Firstly, it can be used just like ordinary RAM, as workspace and to hold programs, etc, secondly, it can be used as "LOADABLE EPROM" and if a Write Disable is incorporated, the content is as safe as if it were in EPROM, but can still be changed virtually instantaneously when required. Also, when developing programs, they can be stored in the RAM safely whilst the program is tested, or you can go away for a tea break and and not have to worry about mains dropouts or the wife unplugging your machine to use the Hoover. I am convinced it is preferable to using just normal RAM and I am seriously considering fitting 64K of Cmos RAM.

Diagram 1

RAM DIS and Write Control



So, after this extended introduction, I will proceed to enlighten those interested parties on how it is done. It is based on using up the 8 EPROM/RAM sockets on the Nascom 2 main board. The 8 sockets will accomodate most byte wide, 24 pin devices and the HM6116s fall into this category. For straight forward RAM expansion without battery back-up and Write Disable, fitting is very similar to fitting 4118s in these positions except that a higher order addressing is required to access all the available memory. The only additional circuitry is to provide RAM-Dis onto the Nascom 2 to disable this new RAM if you are using external RAM or ROM at the address chosen to locate your new RAM. It is suggested that this stage be done first anyway, to check that the RAM is operating correctly, before introducing the complexities of the power down and Write Disables. Refer to the link block diagram in your manual (you have got a manual, haven't you!!) and connect pin 8 to pin 4 (chip select), pin 6 to pin 2 (output enable to read enable), pin 5 to pin 1 (write enable) and finally, loop pin 7 (address line A10) along each pin 7 and pick up address line A13 off the data bus (best done from pin 12 of IC 47 (N2MD)). This has the effect of addressing through all the lower 1Ks of each chip and then through the upper 1Ks of each chip (the upper and lower halves of each chip are seperated by 8K) and is the simplest way to do it. Finally, to make the whole thing go, connect the RAM-gates to the blocks of addresses required eg. D000 to F000 would give you 16K of RAM starting at D000 and finishing at FFFF so, if you are using a CP/M type system, then set your 48K card to go from 0000 to CFFF and you will have 64K on-line by connecting either RAM-G1 or RAM-G2 to block address C000 + D000 + E/F000 strapped together (the last 8K block is normally used to enable the BASIC ROM). To provide RAM-Dis onto the p.c.b. requires two ICs, one type 74LS32 (quad OR) and one type 74LS04 (hex inverter) and these devices will be used to provide the Write Disable control as well (you might as well go the whole hog!). In my system, these are mounted piggy back, on top of two other conveniently situated 14 pin ICs close to the link block LKS1, to which most of the connections will be made. Remove IC 18 (74LS06), bend up pin 13 and then replace the IC. For the remainder of the connections it is best to refer to circuit diagram 1 alongside the Nascom 2 Memory circuit diagram, top left corner, where all the original circuitry will be found. I would suggest that you now test, very carefully, the operation of your new RAM, both with EPROM cards and other peripheral cards occupying the same address space in both enable and disable in case there is any interaction. My own system has been operating for many months and I have had no problems. NOTE that the linking is altered so that the Write Enable is linked to the Write Disable switch.

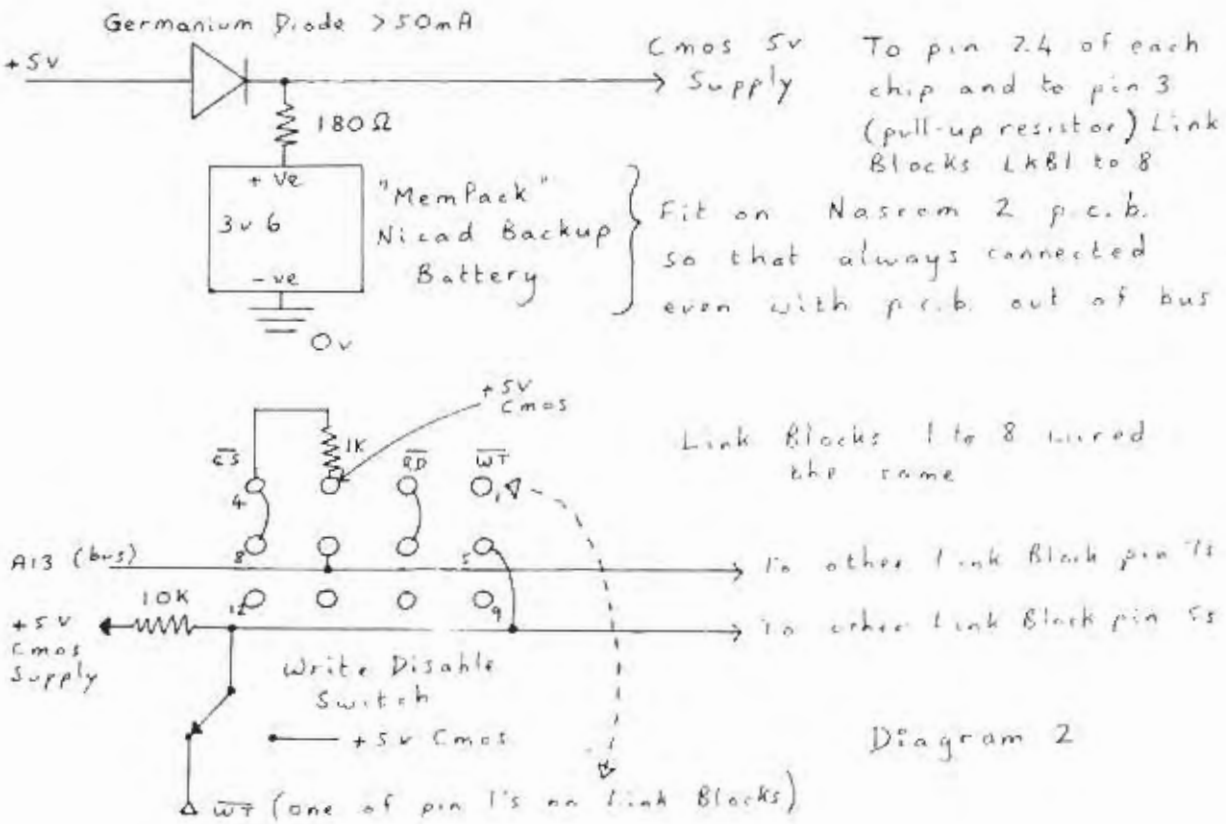
Now for the tricky bit, battery back up cooection. It is not difficult, just time consuming and some care is necessary to ensure that the correct connections have been made. The idea is to isolate all of the +5V connections to pin 24 of each chip and also, to pin 3 of the associated link blocks. This can be done by cutting about five tracks and adding a few wire links. I would have included details but it now very difficult to gain access to the underside of my N2 card and I have no record of what I did - however, I would be pleased to reinvent the wheel if anyone wishes to take the plunge. Otherwise, lifting IC legs

is probably the easiest. Check that you still have +5V on your Monitor and BASIC ROM and other RAM/ROMs in the vicinity. (I didn't when I did it.)

Next, obtain a 74LS156 and fit it in place of IC 46 (which is at present a 74LS155 of which the 74LS156 is an Open Collector version). This is necessary to allow correct power down operation. Fit 8 of 1K resistors between pin 3 and pin 4 of each link block LKB1 - 8. This provides the Open Collector pull-up to the new battery backed, 5V supply, when it is fitted.

Now for the back-up supply. This is best done by reference to diagram 2 where the normal 5V supply can be obtained from the test pins near the bus connector and where the new 5V supply is connected to the new RAMs and pull-up resistors ONLY. (It is worth noting that another HM6116 could be fitted in place of the Monitor and Video ROM/RAMs and, by the connection of the back-up supply and the addition of Write strobes, you would be able to change Monitors and Video/Graphics at will and not loose their contents on power down, therefore mimicking the present ROMs.) Write Inhibit switching is recommended as there can be no guarantee that spurious writes to the Cmos RAM will not occur during power up and power down. However, to date I have not experienced any problems in this respect and have tried to provoke spurious errors without "success". To be certain, ALWAYS DISABLE WRITE BEFORE POWER UP OR POWER DOWN. I have not thought of a failsafe, software controlled Write Disable. If it could be incorporated, it would open a few more avenues for use.

Having now had this facility on my system for some while, I now wonder how I got by before it was fitted, and the time spent getting it right was well worth it.



ARE FLOPPY DISCS TOO EXPENSIVE?

Why not consider the HOBBIT floppy tape? This Nascom approved product is fully automatic and uses the Philips Digital Cassette recorder with a data transfer speed of 750 bytes/sec! Each cassette can hold 101K bytes organised in up to 138 files!

BRIEF SUMMARY OF COMMANDS

When a file is required, the message "name" will be displayed after the command has been entered.

- W Write a file
- L Load a file into memory
- R Read a program file and autostart
- D Delete a file
- C Change a file name
- K Kill: Delete all the files
- M Mount:Read index into memory
- N List the names of all files and the number of free blocks
- E End: Rewind cassette ready for removal
- F Format a cassette
- B Save a BASIC or NASPEN file
- Z Save a ZEAP file
- X Return to monitor
- S Select a drive
- T Transfer a file between drives

The HOBBIT operating system is supplied in a 2716 or two 2708s. It is compatible with all Nascom monitors. It can be supplied at any address you require. (D000 standard) Price £99 + VAT

MICROSOFT BASIC UPGRADE KIT

This enables BASIC to read and write HOBBIT files using "PRINT" and "INPUT" statements rather than PEEK and POKES. Price £10 + VAT

Why not find out more about these and other products by contacting David Tucker, Ikon Computer Products, Kiln Lake, Laugharne, Dyfed, Tel:099-421-515

SCAL SPACE, DF 69

This subroutine call outputs a single space by loading the accumulator with £20 and calling the ROUT restart, £F7. The section of code which does this is the end of routine TBCD3 (DF 66), which outputs a space after the contents of HL have been printed in hexadecimal.

SCAL CRLF, DF 6A

A carriage return/line feed is output by loading the accumulator with £0D and calling restart ROUT. The code used is the end of the next, 'Error message', subroutine.

SCAL ERRM, DF 6B

This subroutine prints the word 'Error', followed by a carriage return to position the cursor at the start of the next line. This call is very useful for reporting input errors in machine code programs. Just print the type of error and call DF 6B; this saves six bytes ('Error' + CR/LF) by the use of two.

SCAL TX1, DF 6C

The main use of this subroutine call is to print out the header on each block of data for tape input/output. The start address of the block is in the HL register pair, the length of the block in E, and the block number in D. After the header data has been received from or sent to tape it is printed out using DF 6C; the routine sums H, L, D and E into the C register, and the value obtained is used as a checksum by the Read and Write routines. SCAL TX1 really consists of a single routine which prints out the contents of HL by a call to TBCD3 (DF 66) and then exchanges the HL and DE registers. Because this section of code is preceded by a relative call to itself (D7 00), it is carried out twice. The result is that first HL is printed, HL and DE are transposed, then DE is printed and HL and DE are restored to their original positions. If you use this subroutine and need the checksum, you must first set register C to zero.

SCAL SOUT, DF 6D

This routine sends up to 256 bytes of data to the serial output. On entry HL must contain the address of the start of the data to be sent, and the number of bytes must be in B. The C register is first set to zero, then each successive byte is added into C and sent to the serial port. using SCAL SRLX, DF 6F. On exit from the routine B is zero,

C contains the checksum for the data, and HL points to the next byte following the data. Note that if B is zero on entry, 256 bytes will be output.

SCAL XOUT, DF 6E

XOUT provides facilities for communication with external serial devices. On entry the contents of the accumulator are saved on the stack and HL is set to point to $XOPT (£0C28), which is the byte used to store the flags for the various options available with XOUT. The value stored at $XOPT is controlled by the X command; it is also affected by characters input via the XKBD routine.

The state of bit 7 of XOPT is first tested. If bit 7 is 1, the character is not output and the routine terminates by resetting bit 7 to zero (so that only one character at a time is suppressed) and recovering the accumulator from the stack. If bit 7 is zero, the character is output by a call to an external serial output routine , XSOP, before the routine is terminated as above.

XSOP outputs the character in the accumulator by a relative call to a subroutine, XSOP0. Here the accumulator is ORed, which sets the zero flag if the accumulator contained a null (00). In Nas-Sys 1 the subroutine is terminated if a null is found; thus the Nas-Sys 1 External output routine will not send nulls. In Nas-Sys 3 however, the Return on Zero instruction has been omitted, and nulls can be output.

A second effect of the OR A instruction which starts XSOP0 is to set the parity/overflow flag if the number of bits in the accumulator which are '1' is even; otherwise the parity/overflow flag will be reset. If the parity of the accumulator was odd, the output routine now changes the state of bit 7; the result is that the accumulator now contains a code of even parity, the bottom seven bits (bits 0 - 6) representing the ASCII character to be output.

Next bit 0 of £0C28, the external option byte, is tested; if this bit is zero, the character is to be output with even parity, and the routine jumps straight to the final section, where SRLX is called. However, if bit 0 of £0C28 was 1, bit 7 of the accumulator is first inverted, so that all output is of odd parity.

After return from subroutine XSOP0, the routine is terminated, unless the character that has just been output was a carriage return. In this case, bit 4 of the option byte is checked, and if it is zero the accumulator is loaded with the line feed code (£0A); this is then output by a 'fall through' to XSOP0.

SCAL SRLX, DF 6F

This routine, which handles all the serial output, is extremely simple, because all the data serialisation routines, including the addition of start and stop bits, and the data transmission, is performed by the UART chip. The contents of the accumulator are first saved on the stack; the accumulator is then output to port 1. An input/output operation is decoded on the Nascom 2 by a I/O PROM, IC26, which uses address lines A0, A1 and A2. and buffered Read and Write lines RDB and WRB to select bytes of data stored in the PROM. This data then controls the keyboard and UART. When data is output to port 1 the 'address' used by the PROM is 00001001 (binary), selecting the 10th byte in the PROM, which has the value £EF. When this is placed on the PROM outputs, all the lines remain 'high' except D4, which is connected to the Transmitter Buffer Register Load line on the UART. Thus the data in the accumulator, which has been placed on the data bus by the Port Output instruction, is transferred to the UART transmitter. The UART then procedes to send out this data in serial form, adding the necessary start and stop bits.

Meanwhile, the Z80 processor has nothing much to do. It sits in a loop, reading data out from port 2 and testing bit 6. Decoded by the I/O PROM, the Port 2 read enables the UART status flags. Bit 6 of the data bus is connected to the line which signals that the transmitter buffer is empty. Thus bit 6 going low informs the processor that the data (including the stop bits) has been transmitted. The value initially in the accumulator is then recovered from the stack, and the routine terminated.

You will note that the data bus is connected to the UART input, output and status registers; which particular registers are used by any one operation is determined by the control bytes output by the I/O PROM.

SCAL SRLIN, DF 70

This subroutine checks the serial input port to see if a character has been received. If no character is available it returns with the carry flag clear; otherwise, the carry flag is set and the character is loaded into the accumulator. Once again, the routine is very simple, because the processor merely gets the data from the UART, which has to do all the deserialisation. The first operation is to read port 2. As in the case of SRLX. this enables the UART status flags. Bit 7 of the data bus is conected to the Data Received line, which goes high when the UART has obtained a data byte. The byte read in from port 2 is rotated from the accumulator into the processor carry flag. Thus if no data has been received the carry flag is reset, and the routine ends. If the Data Received line is high, the carry flag will be set by the rotate, and the processor then reads port 1 into the accumulator. This read operation, decoded by the I/O PROM, pulls the UART Receiver

Register Disable line low, which allow the data in the Receive Buffer to pass on to the data bus and thus to the accumulator.

SCAL NOM, DF 71

All input/output in Nas-Sys is controlled by tables of routines. If you want to change the sequence of operations carried out on output you merely have to set up a new table, which must consist of a list of the numbers of the subroutine calls to be carried out terminated by a null, and then tell Nas-Sys where this table starts by loading the address into the HL register pair and calling NOM.

The address of the new table is first saved on the stack and the address of the current output table is loaded into HL from the output table pointer at £0C73. The contents of HL are now exchanged with the top two bytes of the stack; thus HL now contains the new output table address, and the stack holds the current address. Finally, the new address is stored at £0C73, becoming the address for all subsequent output operations, the old address is POPped of the stack into HL, and the subroutine ends.

SCAL NIM, DF 72

This subroutine operates in the same manner as DF 71 to change the address of the input table, which is stored at £0C75.

SCAL ATE, DF 73

This subroutine call is used by input and output procedures to get the routine numbers from the appropriate tables and call them in turn. On entry to ATE the address of the appropriate table pointer, £0C73 for output and £0C75 for input, should be in HL. However, before ATE is used within Nas-Sys, the original contents of HL are saved on the stack; consequently, if you try to use ATE from a machine code program you will find that there is an extra POP instruction at the end which throws away your return address, creating havoc. Of course, there is no need to call ATE, because the standard input and output routines in Nas-Sys, SCAL IN (DF 62) and ROUT (F7), jump directly to ATE after saving HL.

The address stored at the table pointer is loaded into DE, and the routine numbers are then picked up successively from the table until the null that marks the end is detected. The routines in the table are called differently by the two versions of Nas-Sys. In Nas-Sys 1 the routine number is stored at ARGC (£0C0A) and the routine executed by calling SCALJ (DF 5C). In Nas-Sys 3 the number is transferred to the E register and the routine is called via SCALI.

nascom

BRITISH AND BEST

SYSTEMS		ADD-ONS	
48K NASCOM 3	£499.00	Castle Tape Interface.	£17.50
8K NASCOM Micro Ed.	£399.00	Milham A/D Converter.	£49.50
NASCOM Single Drive.	£470.00	Screen Weave Eliminator	£8.75
NASCOM Dual Drive	£685.00	Screen Flash Eliminator.	£14.75
NASDOS Disc Operating System.	£45.00	Programmer's Aid.	£28.00
CP/M Disc Operating System.	£100.00	Arfon Speech Board	£85.00
Hitachi Tape Recorder	£25.00	W.S. Big Ears.	£45.00
12" Green Screen Plastic Monitor.	£99.00	PMG Soundbox	£40.50
12" Green Screen Metal Monitor.	£125.00	CMOS 2K Ramboard.	£95.00
Epson MX80 F/T2 Printer.	£399.00	CMOS 32K Ramboard.	£185.00
TEC 1550 Printer	£699.00	I.O. Systems Pluto high res. Graphics Board.	£399.00
		Colourgraphics NASCOM AVC On Demonstration Now	£155.00
		RAM B 64K Upgrade	£30.00

All Prices inc. VAT

NASCOM
Networking up to 32 Machines on one master

SOFTWARE
Over 100 items in stock covering business, educational games.

NAS-SYS/NAS-DOS
Suppliers: - NASCOM – applications packages.
Dove Computer Services – polished system, software.
Program Power – superb games and utilities.
Mike York – extensions to BASIC.

Business & Leisure Micro Computers

16, The Square, Kenilworth, Warwickshire CV8 1EB. Telephone: (0926) 512127.

VISIT OUR GREATLY EXPANDED SHOW ROOM

---

new software for nascom from dove computer services

DCS-DASH Design Aid for Screen Handling. Full 'Print Using' facilities for Basic together with superb screen formatting. Make your programs totally professional. Supplied on high quality tape or disk for only £22.00 + V.A.T. *Nascom approved.*

DCS-DASH IVC All the features of DCS-DASH plus full support for the Gemini IVC 80 × 25 video. £25.00 + V.A.T.

DCS-DASH AVC For the Nascom Advanced Video Card. Available shortly.

DCS Utilities For DCS-DOS2 and Nas Dos. Greatly improved disk utilities from the authors of Nas Dos. Send for details.

DCS-MOS Multitasking Operating System. An amazing operating system which runs up to 7 tasks at once. Supplied complete with a print spooler and a real-time clock. Compatible with Nas-Sys 3, DCS-DOS 2 and Nas Dos. On tape, disk or EPROM for £22.00 + V.A.T. *Nascom approved.*

DCS-SORT Sort and Index. Sorts DCS-DOS 2 and Nas Dos files into any sequence and creates indexed files. Up to three sort fields, ascending or descending and record selection. Sorts can be incorporated into your software. Supplied on disk for only £25.00 + V.A.T.

AVAILABLE SOON – PROFESSIONAL APPLICATION SOFTWARE

Send for Fact Sheets on all our products. All our software is available post free by mail order – or from your Nascom dealer.

Dove Computer Services,
5, Dove Close, Newport Pagnell, Bucks.
Tel. Newport Pagnell (0908) 612465

---

io research ltd.

"PLUTO" COLOUR GRAPHICS PROCESSOR

Pluto is a self-contained colour display processor on an 8" x 8" NASBUS and 80-BUS compatible card featuring:

- Own 16 bit microprocessor
- 192 Kbytes of dual-ported display memory for fast flicker - free screen updates. (Outside of the host address space)
- 640(H) x 288(V) x 3 planes (8 colours) – 2 screenfulls
  OR
  640(H) x 576(V) x 3 planes (optional extra)
- Fast parallel I/O interface usable with ALMOST ANY MICRO. Only single +5v supply required.

Pluto executes on-board firmware providing high level functions such as:

- Fast vector draw – over 100,000 pixels/sec. Lines can be drawn using REPLACE, XOR, AND, OR functions
- User-definable characters or symbols
- Spare display memory with memory management facilities for allocating symbol storage space or workspace
- Rectangle Fill and copy using REPLACE, XOR, AND, OR plus 5 other functions
- Fast access to single pixels
- Write protect memory planes during copy
- Double-buffered screen memory for animated displays
- Complex polygon colour fill

Pluto is expandable. An expansion board will be available later this year to give Pluto up to 8 memory planes with no loss of resolution. S100 Interface now available.

AVAILABLE NOW. ONLY £399 + VAT (p&p free)
Dealer and OEM enquiries invited.

6 Laleham Avenue, Mill Hill, London NW7 3HL
Tel: 01-959 0106

---

io research ltd.

BABY PLUTO

320(H) x 288 (V) x 8 COLOUR DISPLAY

The power and performance of Pluto but with 96 Kbytes of memory and half the resolution. An ideal match for low cost colour monitors. INCREDIBLE VALUE AT ONLY £299 + VAT.

A/D BOARD FOR NASCOM

- 8 input channels
- 30 microsec conversion
- Over voltage protection
- Prototyping area
- 8 bit resolution
- Sample and hold
- Full flag/interrupt control
- NASBUS compatible

Price £120 + 15% VAT (post free)

EPROM PROGRAMMER

- Programs 3 rail: 2708/2716
  Single rail: 2508/2758, 2516/2716, 2532/2732
- Software supplied for Read/Program/Verify
- Can be used with other machines with 2 parallel ports

Price £63 + 15% VAT (post free)

6 Laleham Avenue, Mill Hill, London NW7 3HL
Tel: 01-959 0106

Screen Reverse

A. Marshall

This circuit, for both Nascom 1 and 2, gives port control over Background Colour (0), Reverse Alphanumerics (1), Screen Blanking (2) and Graphics Switching (3). The latter may be required when graphic codes have been used in machine code programs assuming that no graphics are available. (eg. the Othello program in an early edition of INMC)

The circuit in figure 1 shows how this is achieved using only 3 TTL chips, and the truth table of figure 2 shows how the Exclusive-OR gate can be used as an inverter. If A is low then C=B but if A is high then C=/B.

In the following description of the construction, the ICs referred to within the square brackets, [], are on the Nascom 1 board and those in round brackets, (), are on the Nascom 2. Note that I have a Nascom 1 and that the Nascom 2 connections are taken from the circuit diagram, but I believe them to be correct.

The prototype was constructed on Veroboard and the layout is shown in figure 3. The power supplies and port connections were taken by a 6-way ribbon cable to a 16 pin DIL header plug which was plugged into Skt a, Port 4. As the prototype was added to an Econographics board, the Serial In connection was made to the bent up pin of IC 5/6 and the Serial Out connection plugged into the socket. [On a Nascom without the Graphics, IC 15/9 provides the Serial In in the same way.] (Nascom 2 IC 65/9)

The switch or link on the Econographics board between socket 1/20 and IC 6/12 is replaced by the A7 wire connected to the socket side and the Graphics Enable connection made to the IC 6 side, [A7 to IC 17/19, GE not connected. Als o connect IC 17/18 to IC 20/12] (A7 to IC 67/19 side of LSW 2/9 and GE to IC 66/18 side). The Graphics clock is connected to the wire link to the side of IC 6/11 [IC 17/11] (IC 67/11). With the Econographics board fitted, the Veroboard can be mounted on to it, but if not, then a wire wrap socket could be used to plug into the PIO socket instead of using ribbon cable and a DIL header.

After connecting the board to your Nascom, the display will be black on white as, all the inputs are held high by the pull-up resistors but as yet, the Port will not be controlling. To initiate the Port, type O 6 F and the screen will go blank as the output of the Port is zero. Type O 4 F and the display will return but now under Port control. To change the display under program control, send the appropriate value to the Port.

PORT BIT

0	low gives black background
1	low gives reverse alphanumerics
2	low blanks the display
3	low disables the Graphics ROM

Exclusive-OR
Truth-Table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Figure 2

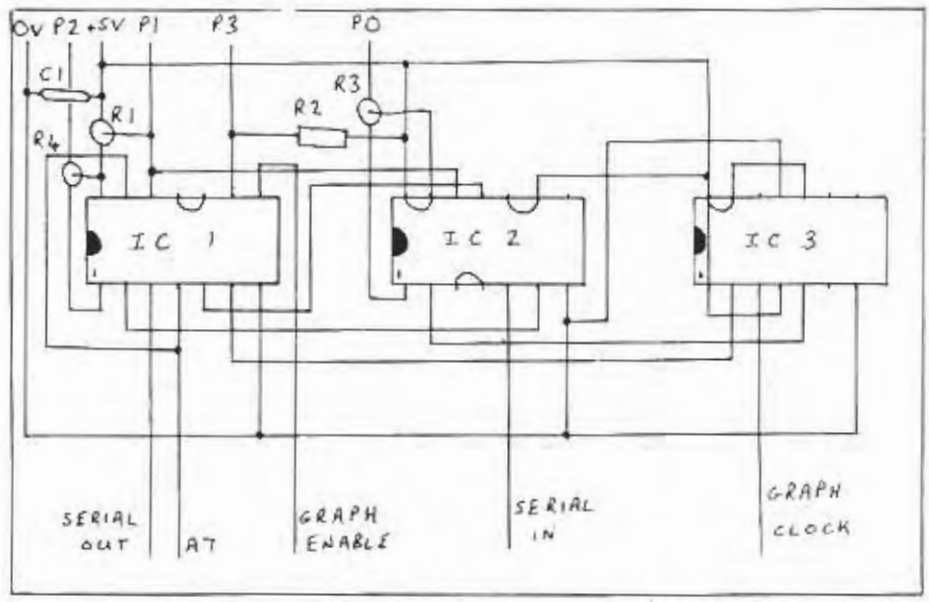


Figure 3 Suggested Veroboard Layout

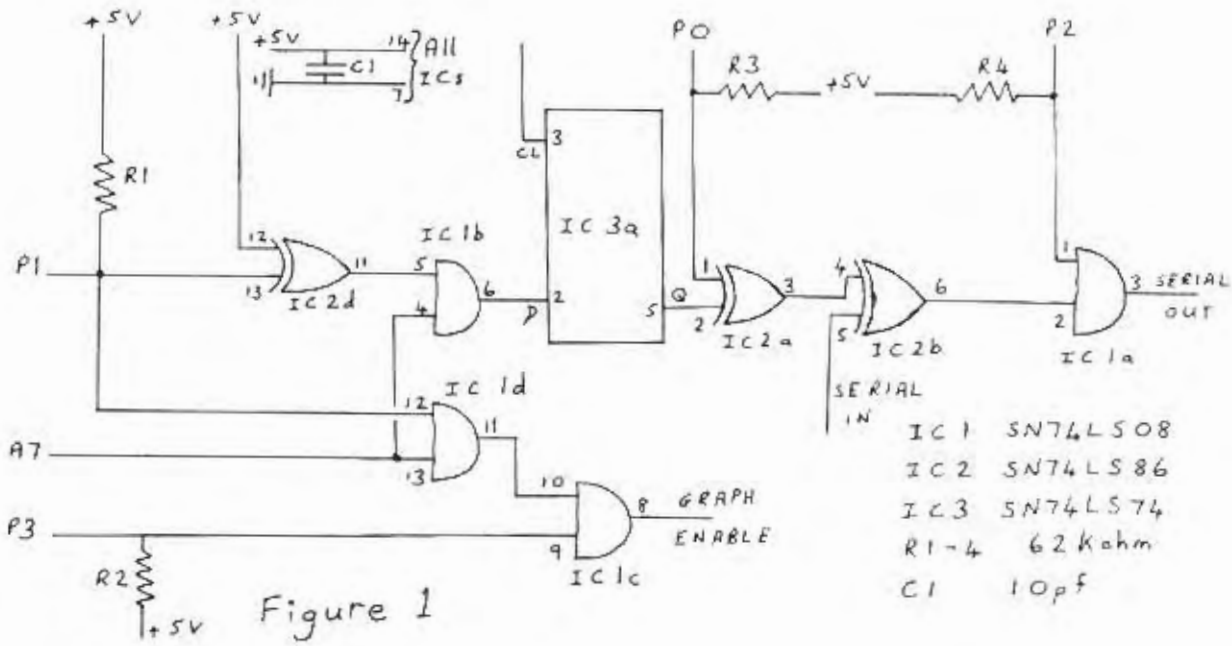


Figure 1

Facilities 0, 2 and 3 are self-evident but the Reverse Alphanumerics requires some explanation. The graphics bit, Bit 7, is re-routed to provide the reverse signal, so that Graphics and Reverse Alphanumerics cannot be on the screen at the same time. The difference between ASCII "A" (41) and ASCII reverse "A" is 80 so that with Bit 1 of the Port low, C1 will give reverse "A".

Programming in machine code is simple, as all that is required is the addition of 80 to the ASCII code for a letter or number. This can be done using the comma before a letter in the Modify mode and then pressing the Graphics key with the letter if you have the modified keyboard described in Micropower, Volume 1, Number 2. (A superb modification. The most difficult part being the cutting out of the keyboard case for the extra keys. Also, if you, like me, do things and then think about them, please note that it is the zero key and not the O key that is modified.)

Programming the Reverse Alphanumerics in BASIC is simple if you have the Graphics key but some string manipulation is required if you haven't, in order to set Bit 7. The following is a suggestion but could, I am sure, be improved.

```
5 REM SET UP ORIGIN AND DESTINATION STRINGS
10 A$="Hello":B$=""
20 FOR I=1 TO 5
25 REM SET BIT 7 OF EACH CHAR. IN ORIGIN STRING
30 B$=B$+STR$(ASC(MID$(A$,I,1)+128))
40 NEXT I
```

The Reverse Alphanumerics work regardless of the screen background, so that black letter on a white background are produced if the screen background is black and vice versa. One thing that I should mention about RA is that the top of each character is on the top row of the line so that if, for instance, "E" is in RA below a normal line, then the top bar will tend to merge with the line above. However, this a very cheap modification and some snags can be expected. For perfectionists, a similar commercial product is available costing, I believe, about £40.

==========

Othello

by A. Brown

"Not another Othello", I hear you say. Yes, it is, but this is a very nice version, written in machine code for Nas-sys, it executes at 1000H and has a nice feature called "Hint" which you get by pressing the Space Bar on your move. Beware though, it plays a mean game.

```
1000 C3 20 16 4F 54 48 45 4C 4C 4F 20 20 62 79 20 41   . .OTHELLO  by A
1010 20 42 72 6F 77 6E 43 6F 6D 70 75 74 65 72 3D F5    BrownComputer=.
1020 F6 F7 FF 01 09 0A 0B 00 17 20 2D 36 18 23 2A 35   .........  -6.#*5
1030 00 01 02 03 04 05 06 07 0A 11 14 1B 1E 25 28 2F   .............%(/
1040 32 39 3C 43 46 47 48 49 4A 4B 4C 4D 21 DB 08 11   29<CFGHIJKLM!...
1050 30 00 DD 2A 39 0D 0E 08 06 08 DD 7E 00 77 23 23   0..*9.....~.w##
1060 DD 23 10 F6 DD 23 DD 23 19 0D 20 EC 21 09 0D 34   .#...#.#.. .!..4
1070 C9 21 59 0B 22 29 0C 3E 1B F7 22 29 0C C9 DF 62   .!Y."). >..")...b
1080 D0 18 01 CF FE 1B CA 20 16 C9 09 3A 07 0D EE 7F   ....... . ...:....
1090 BE C0 09 BE 28 FC EE 7F BE C9 21 06 0D 36 01 ED   ....(...!..6..
10A0 5B 41 0D 2A 3F 0D ED 53 3F 0D 73 23 72 EB 36 00   [A.*?..S?.s#r.6.
10B0 23 36 00 23 3A 3D 0D 77 23 36 00 23 22 41 0D C9   #6.#:=.w#6.#"A..
10C0 2A 3D 0D ED 5B 39 0D 19 7E FE 2E C0 11 1F 10 1A   *=..[9..~.......
10D0 13 B7 C8 4F 17 9F 47 D5 E5 CD 8A 10 CC 9A 10 E1   ...O..G.........
10E0 D1 18 EC AF 32 06 0D ED 5B 41 0D 2A 43 0D 23 23   ....2...[A.*C.##
10F0 73 23 72 23 22 43 0D 22 3F 0D 21 0B 0D 3A 0A 0D   s#r#"C."?.!..:..
1100 BE 3E 00 30 2A 32 3D 0D CD C0 10 3A 3D 0D 3C FE   .>.0*2=....:=.<.
1110 08 38 F2 3E 0A 32 3D 0D CD C0 10 3A 3D 0D C6 07   .8.>.2=....:=...
1120 32 3D 0D CD C0 10 3A 3D 0D C6 03 FE 46 38 E6 32   2=....:=....F8.2
1130 3D 0D CD C0 10 3A 3D 0D 3C FE 4E 38 F2 3A 06 0D   =....:=.<.N8.:..
1140 B7 C0 2F 32 3D 0D CD 9A 10 C9 78 87 87 80 87 81   ../2=.....x.....
1150 32 0D 0D C9 DD 21 B4 1A 3A 0D 0D DD BE 02 28 0F   2....!..:.....(.
1160 DD 5E 00 DD 56 01 AF BA 28 0B D5 DD E1 18 E9 DD   .^..V...(.......
1170 22 45 0D AF C9 3E 01 C9 3A 07 0D 77 EE 7F ED 42   "E...>..:..w...B
1180 BE C0 EE 7F 77 18 F5 ED 5B 39 0D 21 5A 00 44 4D   ....w...[9.!Z.DM
1190 19 22 39 0D EB ED B0 2A 45 0D 23 23 7E 32 3D 0D   ."9....*E.##~2=..
11A0 FE FF C8 2A 3D 0D ED 5B 39 0D 19 3A 07 0D 77 11   ...*=..[9..:..w.
11B0 1F 10 1A 13 B7 C8 4F 17 9F 47 E5 CD 8A 10 CC 78   ......O..G.....x
11C0 11 E1 18 EE C5 01 04 00 CD 7E 10 10 FB 0D 20 F8   .........~.... .
11D0 C1 C9 06 00 04 D6 0A 30 FB C6 0A 4F C9 3A 3D 0D   .......0...O.:=..
11E0 CD D2 11 21 9B 08 11 40 00 19 10 FD 09 09 3A 07   ...!...@......:.
11F0 0D 4F 06 03 36 2E CD C4 11 71 CD C4 11 10 F5 C9   .O..6....q......
1200 3A 07 0D EE 7F 32 07 0D C9 CD 97 11 3A 3D 0D FE   :....2......:=..
1210 FF C8 CD DD 11 CD 4C 10 CD DD 11 3A 09 0D FE 3D   ......L......:...=
1220 D8 C3 FC 15 CD 00 12 2A 15 0D 2B 22 15 0D 21 0B   .......*..+"..!.
1230 0D 35 2A 43 0D 2B 56 2B 5E ED 53 41 0D 2B 56 2B   .5*C.+V+^.SA.+V+
1240 5E 22 43 0D ED 53 45 0D 2A 39 0D 11 A6 FF 19 22   ^"C..SE.*9....."
1250 39 0D C9 AF 32 00 0D 32 01 0D 06 4E ED 5B 39 0D   9...2..2...N.[9.
1260 1A CB 7F 20 0C FE 2E 28 08 21 00 0D B7 20 01 23   ... ...(.!... .#
1270 34 13 10 EC C9 2A 3B 0D 1A 13 B7 C8 4F 17 9F 47   4....*;.....O..G
1280 F6 01 09 7E C9 CD 7E 10 3A D9 0B EE 40 32 D9 0B   ...~..~.:...@2..
1290 AF 32 03 0D 21 00 00 22 0F 0D CD 53 12 3A 00 0D   .2..!.."...S.:..
12A0 21 01 0D 96 32 02 0D 21 0A 0D 3A 09 0D 86 FE 3D   !...2..!..:.....=
12B0 D2 3A 13 2A 39 0D AF 32 0E 0D 22 3B 0D 7E CB 7F   .:.*9..2..";.~..
12C0 20 62 FE 2E 11 1F 10 28 32 CD 75 12 28 56 CB 7F    b.....(2.u.(V..
12D0 20 0D FE 2E 20 F3 09 7E CB 7F 28 ED 87 18 06 6F   .... ..~..(....o
12E0 87 87 85 ED 44 2A 3B 0D CB 46 20 02 ED 44 4F 17   ....D*;..F ..DO.
12F0 9F 47 2A 0F 0D 09 22 0F 0D 18 CE CD 75 12 28 24   .G*...".....u.($
1300 FE 2E 28 F7 CB 7F 20 F3 09 BE 28 FC 7E FE 2E 28   ..(... ...(.~...(
1310 EA CB 7F 20 E6 3E 01 CB 46 20 02 3E FF 21 03 0D   ... .>..F .>.!..
1320 86 77 18 D7 2A 3B 0D 23 3A 0E 0D 3C FE 4E 38 87   .w..*;.#:...<.N8.
1330 21 03 0D CB 7E 28 01 34 CB 2E 3A 02 0D 21 03 0D   !...~(.4..:..!...
1340 86 5F 17 9F 57 2A 0F 0D 19 ED 5B 13 0D A7 ED 52   ._..W*....[....R
1350 22 11 0D 7D A7 CB 7C 28 0A 11 81 FF ED 52 30 0B   "..}..|(.....R0.
1360 7B 18 08 11 7F 00 ED 52 38 01 7B 21 07 0D CB 46   {.......R8.{!...F
1370 28 02 ED 44 C6 80 32 04 0D DD 2A 45 0D DD 77 03   (..D..2...*E..w.
1380 C9 CD 87 11 CD 85 12 CD 48 12 C9 ED 4B 43 0D 11   ........H...KC..
1390 00 00 60 69 4E 23 46 72 2B 73 AF B8 C8 ED 43 45   ..`iN#Fr+s.....CE
```

```
13A0 0D CD 81 13 2A 43 0D ED 4B 45 0D 5E 23 56 AF BA   .▀..*C.▀KE.^£V
13B0 28 0B D5 DD E1 3A 04 0D DD BE 03 30 06 70 2B 71   (.▀▀:..▀.0.p+q
13C0 C3 92 13 EB C3 AB 13 3A 08 0D EE 7F 32 07 0D AF   ▌..▀▌.:..▀.2..
13D0 32 0B 0D 21 1D 0D 22 43 0D 21 B4 1A 22 41 0D CD   2..!.."C.!.."A.▀
13E0 E3 10 DD 2A 3F 0D DD 7E 02 32 05 0D FE FF 28 60   ▌.▀*?.▀~.2..▀.(`
13F0 CD 71 10 EF 20 59 4F 55 52 20 4D 4F 56 45 20 3E   ▀q.▀ YOUR MOVE >
1400 20 00 CD 83 10 FE 20 28 64 FE 31 38 2A F7 D6 31    .▀...▀ (d▀18*▀▌1
1410 FE 08 30 23 47 DF 69 CD 83 10 FE 20 28 4F FE 41   ▀.0£G▌i▀...▀ (O▀A
1420 38 15 F7 D6 41 FE 08 30 0E 4F CD 4A 11 CD 54 11   8.▀▌A▀.0.O▀J.▀T.
1430 B7 20 04 CD 09 12 C9 CD 71 10 EF 20 49 4C 4C 45   .▀..▀▀q.▀ ILLE
1440 47 41 4C 20 4D 4F 56 45 20 21 21 00 DF 5D 18 A0   GAL MOVE !!.▀].▀
1450 CD 71 10 EF 20 46 4F 52 46 49 45 54 20 4D 4F 56   ▀q.▀ FORFIET MOV
1460 45 20 20 3F 00 CD 83 10 FE 59 20 F9 C9 3E 80 32   E  ?.▀...▀Y ▀▀>.2
1470 11 0D CD 8B 13 21 D9 0B 36 4F CD 71 10 EF 20 48   ..▀..!▀.6O▀q.▀ H
1480 49 4E 54 2C 20 54 52 59 20 3E 20 00 2A 43 0D 5E   INT, TRY > .*C.^
1490 23 56 13 13 1A 06 30 04 D6 0A 30 FB C6 4B 4F 78   £V....0.▀.0▀KOx
14A0 F7 DF 69 79 F7 DF 5D C3 C7 13 08 2A 29 0C 70 21   ▀▀iy▀▀]▌▀..*).p!
14B0 29 0C 34 3E 3D F7 08 FE 0A 38 0A 47 AF 3C 27 10   ).4>=▀.▀.8.G<'.
14C0 FC DF 68 18 02 DF 7A DF 69 DF 69 C9 3A 08 0D 32   ▀▀h..▀z▀i▀i▀:..2
14D0 07 0D 3A 09 0D FE 01 CA A9 15 AF 32 0B 0D 21 00   ..:..▀..▀.2..!.
14E0 00 22 13 0D 22 37 0D 21 B4 1A 22 41 0D 21 1D 0D   ."..."7.!."A.!..
14F0 22 43 0D 21 17 0D 22 15 0D 06 08 36 00 23 10 FB   "C.!.."....6.£.▀
1500 CD 85 12 2A 11 0D 22 13 0D 21 0B 0D 34 CD E3 10   ▀..*.."..!..4▀.▀
1510 3A 0B 0D 21 0A 0D BE DC 8B 13 2A 43 0D 22 45 0D   :..!..▀..*C."E.
1520 2A 45 0D 5E 23 56 7A B7 28 3D ED 53 45 0D 2A 43   *E.^£Vz(=▀SE.*C
1530 0D 73 23 72 3A 0B 0D 21 0A 0D BE 38 16 21 0C 0D   .s£r:..!..8.!..
1540 BE 30 38 2A 45 0D 23 23 7E 21 30 10 01 1C 00 ED   ▀08*E.££~!0....▀
1550 B1 20 28 CD 87 11 CD 00 12 2A 15 0D 7E 23 23 77   ▀ (▀..▀..*..~££w
1560 2B 22 15 0D C3 09 15 3A 0B 0D FE 01 C8 CD 24 12   +"..▀..:..▀.▀.$.
1570 2A 15 0D 23 23 7E 2B 2B C3 84 15 CD 81 13 3A 04   *..££~++▀..▀..:.
1580 0D 2A 15 0D BE 38 1C 28 1A ED 44 23 BE 38 91 28   .*..8.(.▀D£8.(
1590 8F 77 3A 0B 0D FE 01 C2 20 15 2A 45 0D 22 37 0D   ▀w:..▀.. .*E."7.
15A0 C3 20 15 CD 24 12 C3 20 15 21 26 10 11 04 00 3A   ▀ .▀$.▀ .!&....:
15B0 08 0D B7 20 01 19 ED 5F 07 07 07 07 E6 03 5F 19   .. ..▀_....▀._.
15C0 22 37 0D C9 CD 71 10 3A 0A 0D C6 04 47 21 09 0D   "7.▀q.:..▀.G!..
15D0 3E 3D 96 B8 30 06 32 0A 0D 32 0C 0D CD CC 14 21   >=.0.2..2..▀.▀.!
15E0 D9 0B 36 4F 2A 37 0D 22 45 0D 23 23 7E FE FF 20   ▀.60*7."E.££~▀.
15F0 07 3A 05 0D FE FF 28 04 CD 09 12 C9 CD 71 10 EF   .:..▀.(.▀..▀q.▀
1600 53 43 4F 52 45 20 20 00 CD 53 12 3A 00 0D 06 7F   SCORE  .▀S.:....
1610 CD AA 14 3A 01 0D 06 00 CD AA 14 CD 83 10 18 FB   ▀.:....▀.▀...▀▀
1620 31 00 10 21 30 17 22 39 0D AF 32 3E 0D 32 09 0D   1..!0."9.2>.2..
1630 21 25 17 06 65 36 FF 23 10 FB 21 30 17 0E 08 06   !%..e6.£.▀!0....
1640 08 36 2E 23 10 FB 23 23 0D 20 F4 DD 2A 39 0D 3E   .6.£.▀££. ▀▀9.>
1650 FC DD 77 F5 DD 77 FE DD 77 4F DD 77 58 DD 36 21   ▀▀w▀▀w▀▀wO▀wX▀6!
1660 00 DD 36 22 7F DD 36 2B 7F DD 36 2C 00 3E 0C F7   .▀6".▀6+.▀6,.>.▀
1670 21 03 10 11 D9 0B 01 13 00 ED B0 21 99 08 01 40   !...▀....▀...!...@
1680 08 23 23 0C 71 10 FA 21 99 08 11 40 00 01 30 08   .££.q.▀!...@..0.
1690 19 0C 71 10 FB CD 4C 10 CD 71 10 EF 42 4C 41 43   ..q.▀▀L.▀q.▀BLAC
16A0 4B 20 4F 52 20 57 48 49 54 45 20 20 3F 00 CD 83   K OR WHITE  ?.▀.
16B0 10 FE 42 28 07 FE 57 20 F5 AF 18 02 3E 7F 32 08   .▀B(.▀W ▀..>.2.
16C0 0D 32 EB 0B 21 16 10 11 E2 0B 01 09 00 ED B0 CD   .2▀.!....▀....▀▀
16D0 71 10 EF 4C 4F 4F 4B 2D 41 48 45 41 44 20 20 31   q.▀LOOK-AHEAD  1
16E0 2D 36 20 3F 00 CD 83 10 FE 31 38 F9 FE 37 30 F5   -6 ?.▀..▀▀18▀▀70▀
16F0 D6 30 32 0A 0D C6 06 CB 3F 32 0C 0D CD 71 10 EF   ▀02..▀.▀?2..▀q.▀
1700 46 49 52 53 54 20 4F 52 20 53 45 43 4F 4E 44 20   FIRST OR SECOND
1710 3F 00 CD 83 10 FE 46 28 07 FE 53 20 F5 CD C4 15   ?.▀..▀F(.▀S ▀▀..
1720 CD C7 13 18 F8 00 00 00 00 00 00 00 00 00 00 00   ▀▀..▀..........
```

COORDINATE LIFE

by P. Whittaker

The last issue of Micropower contained a hex dump of a machine code program headed "Coordinate Life". This article is an attempt to describe what the program does, how it does it, and how to use it.

Life, the computer version that is, has been around for several years now. It plots the evolution of a colony of cells which obey a set of simple rules. By changing the rules you can produce wide variations in the behaviour of the colony, but the rules first published by J. H. Conway seem to produce the most interesting patterns. Conways rules are:-

1. The cells occupy a rectangular array (like the squares of graph paper), and each cell has 8 neighbours
2. Each cell is in one of two possible states, alive or dead!
3. If a living cell has no live neighbours, or only one, it will die in the next generation
4. A living cell with four or more live neighbours will die in the next generation through overcrowding
5. An new cell will be formed in the next generation on any empty cell with exactly three live neighbours.

Although you can follow the evolution of simple patterns with pencil and paper, it is much easier, and more interesting, to use a computer to display the complex behaviour that can be produced by these simple rules. However, whatever the size of the display selected, the most fascinating patterns always seem to become too big for the screen. For example, using Pixel graphics the Nascom screen can only display a 96 x 48 cell array, and many patterns soon expand beyond these limits.

You can overcome the problem of a small display size by letting the screen act as a window into a larger array. by using the cursor keys to move the window asound you can inspect any part of the array at will. You then run into another problem - the memory capacity needed by a large array. If each cell is represented by one byte, a 100 x 100 array would need nearly 10K of memory. Even if each cell was represented by the state of a single bit, an array 1000 x 1000 would need more memory than the Z80 is capable of addressing.

This program overcomes the capacity problem by only storing the coordinates of the living cells. Because in any colony the proportion of active cells is small, and tends to decrease as the total population grows (i.e., the area covered by the colony increases faster than the population), you can store the coordinates of the active cells in quite a

small table - it rarely exceeds 4K. Of course, you never get anything for nothing. The software to follow the evolution of the colony is much more complicated ; you have to search the table for adjacent cells, count them, and them apply the rules to build up a coordinate table for the next generation. In addition, you have to write a routine to map the coordinate table to the screen. However, you only have to scan the cells in the table; in the more usual life program all cells, alive and dead, are scanned. The result is that this version of life is much faster for small colonies - in fact you have to slow it down if there are less than 20 cells, or you can't follow the changes. The break even point seems to be a population of 100, when the program runs at 6 generations a second with a 4Mhz Z80, which is about typical for the normal version.

Now to the program. It is started by E1000, and immediately jumps to £1503; the region between contains tables and subroutines. At £1503 the screen is cleared, various workspace parameters are initialised, and the screen header is written to the top line. One of the subroutines called in the initialisation procedure is SETPIX, £12F3. This writes the Nascom 2 pixel set to a programmable character generator. My PCG sits from £0000 - £07FF, so the address where the pixels are situated is £0400 - £07FF; if you have a PCG at a different address, just change £12F4 to the necessary value. If you still use characters in ROM, SETPIX should not affect your system, unless you have RAM down at £0000, and the program will run perfectly happily with ROM graphics.

When you run the program you will be presented with a blank screen, except for the header on the top line and a flashing cursor in the middle. To enter a pattern, use keys Q, W, A, S, Z and X. Each time you press one of these keys the state of the corresponding pixel at the cursor will change. Each time you turn a cell 'on' its coordinates are entered in the table (the coordinates of the bottom left pixel are displayed on the top line), and when you turn it 'off' it is deleted from the table.

With the cursor keys you can move the whole screen left, right up, and down. Note that the screen image moves, but the cursor remains stationary in the middle.\ In effect, you are moving a window around, looking into an array. If you move the pattern off the screen its coordinates are still in the table, so when you move the window back the pattern will reappear. You can thus build up very large and complex patterns; however, it is best to experiment with small patterns initially.

To help you there are several patterns stored in the program - you can make these appear at the cursor by entering SHIFT/A - Z. You can enter these standard patterns at several different positions. You can also rotate and reflect the whole screen by pressing R and F. By combining direct entry and the standard patterns you can quickly build up complex arangements.

Such an arrangement can be stored for later recall by entering U followed by S (for User pattern Save). These patterns are stored in a reserved region of memory starting at £1A55. A series of patterns can be stored until the reserved space is full. If the pattern has been stored successfully the screen will clear, but if there is insufficient space for further storage the pattern will remain on the screen. You can clear the user space by entering UC; UW will write the data in the user space to tape, and UR will read a tape back in.

To follow the evolution of an arrangement just press G. If part of the pattern moves off the screen you can stop the process at any stage by holding down the space bar; then you can move the window to the area that interests you and restart. If the whole pattern interests you, but it becomes too big for the screen, you can zoom in and out with I and O, after stopping the evolution, and restart. The zoom factor is displayed on the top line. Whenever the evolution is stopped, you can clear the screen and start to re-enter a new patterns by entering J.

Many life patterns move about the screen while maintaining their shapes (for example, standard patterns f, w, x, y and z); you can follow this movement by setting the display to 'pan'. If you type P, then press the cursor keys to indicate the direction (left followed by up will give a diagonal movement), and finally enter a number between 1 and £0C for the number of generations between each screen movement, when you start the evolution the screen will follow the pattern at the specified rate. Entering P0 will turn off the pan option.

If you don't want to think up your own patterns, entering Y will generate small random patterns for you to follow. However, only a small proportion of such patterns do anything interesting.

The remaining commands, H and L, switch between high and low resolution. You can only use them if your system is fitted with a PCG. In high resolution each cell is represented by a single dot, so on an unzoomed display the resolution is 384 x 240 (the top line still contains the header). By using the Zoom facility you can increase this to more than 3000 x 1900! However the code is written for for a PCG situated at £0000, so some modification will be rquired to make it work with different systems.

Finally, here are the picturesque names by which the standard patterns are usually known:-

a The Acorn	b Blinkers	c Collision
d Diehard	e The Eater	f Flying Machine
g Glider Gun	m Methusulah	n Newgun
o Octagon	p Pentadecathlon	q Queen Bee
r R Pentomino	s Spaceship Gen.	t Puffer Train
w Glider	x Light Spaceship	y Mid Spaceship
z Heavy Spaceship		

Nascom & Gemini USERS

NEW 32K C.M.O.S. BATTERY BACKED RAM BOARD

FEATURES:

EFFECTIVELY REPLACES EPROMS.
Does away with the inconvenience of EPROM programming and the compromise of assigning valuable address space to ROM.

ON BOARD RE-CHARGEABLE Ni-Cad BATTERY RETAINS MEMORY FOR OVER 1000 Hrs.
Battery is automatically charged during power-up periods.

HIGH SPEED OPERATION up to 6 MHz WITHOUT WAIT-STATES.

FULLY NASBUS[1] and GEMINI-80 BUS[2] COMPATIBLE.

PAGE MODE SCHEME SUPPORTED.
The board can be configured to provide one 32k byte page or two completely independent 16k byte pages.
Complete pages of 64k bytes are simply implemented by adding more boards on to the bus.

SOFTWARE and/or HARDWARE READ/WRITE PROTECTION.
4K blocks in either page are link selectable to be aligned on any 4K boundary.

FULLY BUFFERED ADDRESS, DATA AND CONTROL SIGNALS.

MEMORY I.C. SOCKETS ARE LINK SELECTABLE TO SUPPORT ANY 24 PIN 2k byte MEMORY I.C.s.
Thus the board can support up to 32k bytes of any mixture of cmos, nmos rams or 2716/2516 eproms.
All options are link selectable using wire links plugged into gold-plated socket pins, avoiding the risk of damage and the inconvenience caused by soldering links directly to the board.
The printed circuit board is manufactured to the high quality demanded by industrial users and conforms to B.S.9000.
The board comes assembled and tested and is supplied with a minimum of 2k bytes of low-power cmos ram. Fully documented.

AVAILABLE NOW!

PRICES:

Board with 2k bytes	£99.95
Board with 16k bytes	£149.95
Board with 32k bytes	£184.95
Bare Boards (circuit diagram supplied)	£45.00
HM6116-LP/3 Very low power 2k cmos memory I.C.	£6.50

Please add 95p P&P and VAT @ 15%

[1]nasbus is a trademark of nascom microcomputers a division of LUCAS LOGIC
[2]trademark of GEMINI MICROCOMPUTERS LIMITED

cheques and P.O.s to:
MICROCODE (CONTROL) LTD.
41a MOOR LANE, CLITHEROE, LANCS. BB7 1BE.
For further information Phone (0200) 27890

Please supply me with the following: Price

Total enclosed £

Address:

POST CODE

PROGRAM POWER PROGRAM POWER PROGRAM POWER PROGRAM POWER

NASCOM 1&2 OWNERS!

ELDORADO GOLD

MIN. 16 K/G.

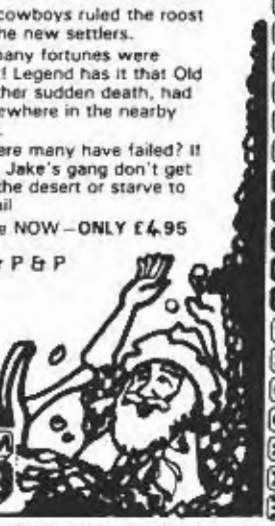
INTRODUCING A GREAT NEW M/C—CODE ADVENTURE FROM PROGRAM POWER

Welcome to a past age, when cowboys ruled the roost and indians terrorised the new settlers.
In ELDORADO COUNTY many fortunes were made—and nearly as many lost! Legend has it that Old Bill McCluskey, who met a rather sudden death, had built up a vast treasure somewhere in the nearby territory.
Can you end up a rich man where many have failed? If the marauding Apaches or Big Jake's gang don't get you, you may die of thirst in the desert or starve to death in jail

Begin your Western Adventure NOW—**ONLY £4.95**

Please add 55p order P & P + VAT at 15%
Send SAE for our Full Catalogue
PROGRAM POWER
5 Wensley Road
Leeds LS7 2LX
Tel. (0532) 683186

PROGRAM POWER



PROGRAM POWER PROGRAM POWER PROGRAM POWER PROGRAM POWER

nascom

BRITISH AND BEST

SYSTEMS		ADD-ONS	
48K NASCOM 3.	£499.00	Castle Tape Interface.	£17.50
8K NASCOM Micro Ed.	£399.00	Milham A/D Converter.	£49.50
NASCOM Single Drive.	£470.00	Screen Weave Eliminator.	£8.75
NASCOM Dual Drive.	£685.00	Screen Flash Eliminator.	£14.75
NASDOS Disc Operating System.	£45.00	Programmer's Aid.	£28.00
CP/M Disc Operating System.	£100.00	Arfon Speech Board.	£85.00
Hitachi Tape Recorder.	£25.00	W.S. Big Ears.	£45.00
12" Green Screen Plastic Monitor.	£99.00	PMG Soundbox.	£40.50
12" Green Screen Metal Monitor.	£125.00	CMOS 2K Ramboard.	£95.00
Epson MX80 F/T2 Printer.	£399.00	CMOS 32K Ramboard.	£185.00
TEC 1550 Printer.	£599.00	I.O. Systems Pluto high res. Graphics Board.	£399.00

SOFTWARE
Over 100 items in stock covering business, educational games.

NAS-SYS/NAS-DOS
Suppliers:- NASCOM – applications packages.
Dove Computer Services – polished system, software.
Program Power – superb games and utilities.
Mike York – extensions to BASIC.

Business & Leisure Micro Computers

16, The Square, Kenilworth, Warwickshire CV8 1EB.
Telephone: (0926) 512127.

VISIT OUR GREATLY EXPANDED SHOW ROOM

Neo-3D Globe.(???)

This program was sent to us by a Mr Carl Whalley of Blackburn and plots a very pretty picture.

```
10 CLS
20 M=11:L=1
30 FOR A=1.81818 TO 0 STEP -0.32
40 L=L+0.2:M=M-0.2
50 FOR D=1 TO 40 STEP 0.1
60 X1=(SIN(D)*M*A*L*1.8)+48
70 Y1=(COS(D)*20)+22
80 X2=(SIN(D)*42.4145)+48
90 Y2=(COS(D)*M*A*L*0.848767)+22
100 SET(X1,Y1)
110 SET(X2,Y2)
120 NEXT D:NEXT A
130 GOTO 130
```

If you alter the function in the SIN and COS evaluation you can get some very interesting effects.

If anyone else has any of these nice little programs rolling around at home, we would appreciate your sending them in so that we can print them out and hang them on the stark office walls that we have here at Micropower.

Nasprint Modification.

If you try to use Nasprint with the RAM version of ZEAP, you will find it keeps giving the message "uninitiated". This is because it tests memory location 0C79 to check that "DC" has been put there by ZEAP. However, the RAM version puts "1D" there instead when cold started. Also, after checking that ZEAP is started, Nasprint attempts to return control by jumping to D003. The call for RAM ZEAP is 1003. Therefore, to make Nasprint work properly with RAM ZEAP we must change the locations B16F and B17D in Nasprint to 1D and 10 respectively. This can be done using the Modify command if Nasprint is in RAM but an EPROM programmer is required if you have an EPROM version.

Steve Stubbs, Inverurie

Private Ads.

FOR SALE - RAM A card (4Mhz) with 32K RAM. In full working order with no wait states. £70 o.n.o.
Telephone Brian Oliver - Erith (Kent) 35868

FOR SALE - NASPEN (Nas-sys version) in two 2708 EPROMs: Bits & PCs Toolkit also in two 2708 EPROMs: and finally Nas-sys 1 Monitor ROM. All with their original instructions sheets. Real bargains at £11 per item. (Selling to help finance CP/M on disks).
Ring Steve Stubbs on Kemnay 3070 (STD code 04674)

FOR SALE
RAM A card with 32K RAM. Runs at 4Mhz with no waits. £60
Also RAM A card with 16K RAM. 4Mhz and no waits. £50
Also Castle Interface with original documentation. £8
Ring Ian on Leeds (0532) 683186

PROGRAM POWER PROGRAM POWER

* * * NEWSFLASH * * *

NASCOM SOFTWARE PRICES are

T U M B L I N G

Look Special Low Prices

SPACE GAMES

Moon Raider	£ 6.95
Invasion Earth	£ 6.95
Startrek II	£ 6.95
Galaxian Attack	£ 6.95
Lunar Lander Supreme	£ 6.95
Jailbreak in Space	£ 6.95
New Fase	£ 6.95
Starship Command	£ 6.95
Super Startrek	£ 4.95
Cliff Invasion	£ 4.95
Alien Labyrinth	£ 4.95

OTHER GAMES

Sargon Chess Pack	£35.00
The Keys of Kraal	£ 6.95
Dungeon Quest	£ 5.95
Backgammon	£ 5.95
Draughts	£ 5.95
Graphic Golf	£ 4.95
Eldorado Gold	£ 4.95
Serpent	£ 4.95

APPLICATIONS SOFTWARE

Nasprint '80	£12.95
Wordease WP (2532/2732) (2716 £21.50/2708 £24.50)	£19.50
Basic File Handler	£12.95
Wirral Pilot	£12.50
XTAL BASIC	£35.00
Nascount Finance	£ 6.95
Club Membership	£ 6.95
Vortex	£ 6.95
Constellation	£ 4.95
Music Box	£ 4.95
Indexed File Handler	£ 4.95
Mini Toolbox	£ 4.95
Nascii	£ 4.95
Sound Chip Demo	£ 4.95
Mathspack	£ 5.95

HARDWARE etc.

Cottis Blandford	£14.90
AY-3-8910 Interface	£10.95
AY-3-8910 Sound Chip	£ 6.45
Games Graphics ROM	£ 7.45
Graphics Rom Adaptor	£ 7.45

N.B. Special Offer DEDUCT £1 per program when ordering two or more.

Please add VAT at 15% plus 55 pence per order for post & packing.

MICRO POWER Ltd., 8/8A Regent Street
Chapel Allerton, Leeds LS7 4PE

PROGRAM POWER PROGRAM POWER