80-BUS NEWS

JULY–AUGUST 1984 VOL. 3 ISSUE 4

- DIY EPROM ERASER
- UPGRADING N1 KEYBOARDS
- ACCESSING BULLETIN BOARDS

The Magazine for GEMINI & NASCOM USERS

£1.50

July – August 1984. **80-BUS NEWS** Volume 3. Issue 4.

CONTENTS

SUBSCRIPTIONS

Annual Rates

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

Subscriptions to `Subscriptions' at the address below.

EDITORIAL

Editor : Paul Greenhalgh Associate Editor : David Hunt

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

ADVERTISING

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

PRIVATE SALES

Free of charge to Subscribers by sending to:

ADDRESS: 80-BUS News,
c/o Gemini Microcomputers Ltd.,
Unit 4, Springfield Road,
Chesham, Bucks. HP5 1PU.

Letters to the Editor

HS-1N Micro Cassette

Adrian Perkins writes in the March-April 1984 issue of 80-BUS News about his defection from HS-1N to disk, and mentions a problem with ageing drives. As co-designer of the project, I have been using the same two drives regularly since 1980, and have found them remarkably reliable - hence I have not yet succumbed to the pressure to move to disks. They do, however, tend to drift away from the specified speed with age, and difficulties may arise when an old tape is edited.

HS-1N tapes are initialised by writing a blank catalog at the beginning of the tape; apart from this the tapes are not formatted in any way. At this stage all 28 remaining 2k blocks are available, and when a file is saved the blocks at the beginning of the tape are filled first, always back-spacing one block and running forwards before writing the next block (to allow for a check-read and to get the tape running up to a consistent speed).

If large files are used (say 20k source files, occupying 10 HS-1N blocks on tape) then it may be possible to fit only two on one side of a tape. When the first file is deleted, blocks 1 to 10 on the tape are released for re-use, and if the drive is running at a different speed corruption may take place.

The most likely problem to arise involves a slowing of the drive. In this case, when ten new blocks are saved there may be a residue of the original tenth block which is not overwritten by the new tenth block. Attempts to read the eleventh block (the first block of the original second file) may fail because of this, but you may be able to recover by advancing the tape over the garbage thus:-

1. Read the new file in the normal way (tape stops after block 10), using Drive A.
2. Return control to NAS-SYS.
3. Run the tape forward a fraction by entering 0 78 1 and 0 78 0 one after the other (previously set up on the screen using the cursor keys). Use 0 F8 1 and 0 F8 0 if yours is one of the earliest HS-1Ns.
4. Warm start HS-1N (E D003).
5. Try to load the "lost file" in the normal way.

If the drives are running faster than before, the tenth block of the new file may overwrite the first block of the previous one, and this block is therefore lost for ever. By using the above technique, and editing the HS-1N catalogue workspace, you may be able to recover the later blocks. Data loss from fast drives is less likely than difficulties with slow drives, as the new file must extend across the entire inter-block gap on the tape, before it can corrupt the next block.

I should stress that there is no need to worry about reading tapes from a slow drive on a fast one, or vice-versa, only about editing tapes on a drive with a different speed. It is best to use a blank (either new or deleted) tape when saving edited versions of old files.

Yours sincerely, Dr M.D. Hendry, Cupar, Fife.

Lucas, NasDos, and Colour

Perhaps the following news/viewpoints would be of interest:-

a) There have been several recent letters suggesting that Lucas are not very helpful on some occasions. I can only say that over the last 6 months or so, I have written to them 3 times asking for help or information on various topics, and on each occasion I have received a courteous and helpful response.

They have even agreed to give me a list of registered NasDos users to assist in expanding the NasDos Users' Circle. As to what have they been doing - well, I can answer that at least in part. They have been developing a CAD system now called Lotti II. As a professional engineer, I can say that this is an excellent product and could well transform their sales forecasts. If any readers are thinking of buying a CAD system, then make sure you see Lotti before parting with green stuff.

b) The NasDos Users circle is now actually happening - thanks mostly to Roger Dowling. Hopefully this circle will grow as it becomes better known.

c) The recent article in your magazine comparing various colour cards for NasBus/80-BUS was very useful - it was the final straw that prompted me to invest £150 on the Nascom AVC. Incidentally, I had CP/M 2.2 installed at the same time with the Business & Leisure switch to select either CP/M or NasDos. This is extremely handy since it allows files to be transferred from NasDos disks to CP/M disks by moving to middle memory before switching formats. They still need modifying to run under CP/M of course - should keep me busy for the next 999 years! I have already got NasSys 3 working under CP/M by following the instructions in Chris Blackmore's Monitor.COM article which was found in an early Nascom N.....tter, as modified by Chris Bowden in the next issue - thanks to both gents.

Yours, Colin Case, Rugby, Warks.

PS: The survey was a great idea.

Lucas/Nascom attitutude; 80-BUS irregularity

Thanks for the latest issue of 80-BUS News (vol 3 issue 2) and the questionnaire. I hope that lots of readers return them.

I'm not at all surprised to read about the demise of the Nascom Newsletter - which appears to reflect the generally `couldn't care less' attitude of Lucas/Nascom to Nascom owners. Mr Scadden may have had more luck than most in that the above firm actually acknowledged his letters - they ignore my requests for information or pricelists and seem unwilling to supply spares in less than trade quantities. A great pity!

I would also like to reinforce Mr Scadden's plea for more regular issues of the magazine; perhaps an increase in the part-time editorial staff would help so that pressure of work at Gemini would not cause the erratic appearances to which we have had to accustom ourselves. From the potential and actual contributors' point of view, I hope that the replies to the questionnaires will be published and that we will be told what types of article are required, and, having put our minds in gear, and articles on tape or disk, that you will let us know whether or not the contribution is suitable and roughly when it will appear. This would do much to avoid the annoyance which contributors experience with the present system, to which M. Dasnoy referred. In spite of previous vicissitudes, I do feel that the magazine is very useful and it has a great deal more potential now that it is effectively the only one catering for 80-BUS users.

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

[Ed. - It is certainly our intention to publish a summary of the information gleaned from the questionnaires, and hopefully this will be in the next issue. We are also forever hopeful that we can achieve improved regularity of the magazines, although this has never been our strongest point!! Finally, it is our intention during 1985 to implement a much faster feedback system to potential contributors on whether an article is to be published or not.]

HELP !!!

Seeing various things in the last 80-BUS News reminded me that I had never got going a V&T tape system that I have. I wonder if there is any reader in the NW who has one and could advise. I am at the level of an intelligent dodo as far as this sort of thing is concerned.

Yours, Peter Copping, Manchester, (061-881 0050)

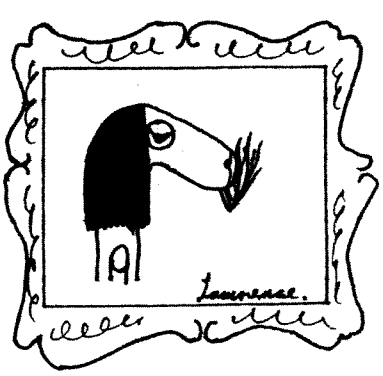
Booby Prize

As an added incentive for 80-BUS readers to fill in their questionnaires, may I suggest a personally signed picture of your friend and his, the mighty Lawrence, as a first prize.

For those poor people who do not win the first prize, you could give them a personally signed picture of Lawrence as a consolation prize.

Listen, here's an idea. How about a personally signed picture of Lawrence as a booby prize!

Yours, Anon



FREE SAMPLE.

(SORRY FOR THE POOR ART-WORK, BUT I WAS "TIRED AND EMOTIONAL" WHEN I DREW IT!) - Hic.

IMP PRINTER RIBBONS

P.A. Cooper of Brentwood, Essex, in Apr-Jun issue, was enquiring about re-inking ribbons for his Imp printer. I have just replaced the ribbon in my Imp with a nylon fabric refill for an Epson MX-100 cartridge. The refill was obtained from Inmac (UK) Ltd, £2.40 (less VAT) each. The minimum postage was £2, so unless you are buying several refills or other items, one refill is a bit expensive.

To change the ribbon, lay out a sheet of newspaper, carefully prize off the top of the cartridge and remove the old ribbon. The refill comes wound as a double coil of ribbon. I put one coil in the cartridge and the other on the table and feed the ribbon through the feed wheel and the slots. I replaced the lid and wound in the ribbon from the table, the coil in the cartidge stops it packing too loosely. Once the ribbon was wound in, the other coil was placed on the table, and the process repeated. The lid was finally glued on with a few dabs of 'UHU' ready for next time. I hope this may be of some use to Imp owners.

Yours, Ian Bissett, Chalfont-St-Peter, Bucks.

PolyDos Languages ?

Does anyone know of the existence of a high level language other than BASIC for PolyDos. I have no grumble about the PolyDos BASIC (it is in fact quite good), but there comes a time when one needs something a little faster and structured, eg, PASCAL or if one is really feeling brave, 'C'. I have the Hi-Soft PASCAL HP4, but it is on tape (after disks, tape is too frustrating

for words ...). I have been trying to "diskify" it, but not really with any success. One is left with the impression that, inspite of the published `Alteration Guide' for HP4, Hi-soft never really intended such a modification.

Yours truly, W R S Webber, Lewisham, London,

NasDos Users

Further to the note that was included in Dr Dark's article in the Nov/Dec issue of 80-BUS News it may interest some readers to know that there is now an active Nas-Dos Users Group circulating a single disk between members which is proving, so far, to be reasonably successful, despite only having 7 members (1 in Australia). That's one up on the 6 members of the PolyDos Users Group run by `Angry' of Tonyrefail!.

Anyone who would be interested in participating within the group can obtain details by sending a SAE to me at the address shown.

Yours truly, Roger Dowling, 393 Blackfen Road, Sidcup, Kent, DA15 9NJ

MOCSAN

Seeing two examples of entirely home-brewed single-board micros recalled an unfinished conversation with two other NASCOM-users during which we mused on the possibility of a home-constructed NASCOM replacement. It was not contemplated that it should be a threat to either the LUCAS or GEMINI organisations, but that it could provide a single-board solution to desire for a machine which could cannibalise a NASCOM for hardware, software and firmware, to produce a more modern, 40/80 col. x 25 lines monitor-display, driven by external RAM and ROM, have on-board DOS and hi-res graphics, and find room for all these via a reduced chip-count in the connectve logic area.

Provided that the constructor re-uses all of the existing bits of NASCOM entirely for personal use, the proprietors of firmware and software can raise little objection to the concept, and with 20,000 machines and a host of subsidiary boards already dedicated to NASBUS and 80-BUS, all concerned should be rather enthusiastic that anyone remains committed to preserving the existing user-base on which their sales and profitability are pre-dictated.

Tentatively coded `MOSCAN', for `Mode Of Computing Simulating A Nascom' (the reverse spelling is **purely** coincidental), it was contemplated that a dedicated group should raise a specification for the board and overall memory-map, then work with a small number of prototypes and a larger number of software-oriented enthusiasts to design cut-and-patch links between existing systems-software, NASCOM BASIC and its TOOLKIT or LEVEL 9 extensions. Although this would not necessitate violation of copyright, because it is feasible to publish patches without listing the orignal code, there is little doubt that proprietors of the most useful programs would find themselves in better standing with the NASCOM **cognoscenti** if they were to permit limited licence to users, contributors and magazine-editors to quote illustrative examples of their wares.

At the participants' level, the program would consist of acquiring a £40 double-sided,through-plated, bare boad, using their existing machine to load existing programs into RAM to cut-and-patch them into comprehensive system-firmware, burn these into an agreed standard size of on-board EPROMs, then to strip down their old machines for buiding into the new board, monitor and PSU and enter a new phase of NASBUS/80-BUS existence!

Would anyone like to comment? Better still: would LUCAS, GEMINI and the other 80-BUS board-manufacturers like to enter into a co-operative venture to write the specification and make the bare board available?

Yours truly, Bert Martin, 15 Grey Mill Close, Shirley, Solihull, B90 4TE

80-BUS - BBC

Could someone please tell me (via the Newsletter) whether anyone has interfaced an 80-BUS system to a BBC B via the 1 MHz tube?

Yours truly, Paul Young, London, W2.

[Ed. - I wonder if Mr Young would like to contact another of our subscribers who also sounds like he's from the group Band-Aid - he just happens to be called Phil Collins !!]

BBC BASIC

Having read Vol 3 Issue 1, I felt compelled to write regarding Dave Hunt's Bits and especially his article ".. and BBC BASIC" which I found rather confusing.

After reading this several times I finally concluded that the article refers to BBC BASIC as found in BBC machines, Electrons, etc. Is Dave Hunt aware that there is a very good version of BBC BASIC available for Z80 based machines called BBC BASIC(Z80)?

I must admit I was unaware of this fact until May of this year.

In April of the previous year, I had saved hard and managed to put together a Multiboard system working under RP/M and using CCSOFT's GBASIC with GRAPHPAC. Then I added a disk drive and purchased CP/M. Looking around for a suitable (and above all, cheap) BASIC language became the next priority.

There seemed to be nothing advertised under about £250 + VAT (MBASIC) and I was almost resolved to buy COMAL or PASCAL instead, when a very helpful Mr Creutzberg at OFF RECORDS in Clapham told me of this BBC BASIC(Z80) which was retailing at £125 + VAT.

I purchased a copy and have been most satisfied. The program is marketed by M-Tec of Norwich and was written by R T Russell. It is supplied with many useful utility programs and HELP files with many example programs. It has its own Assembler too!

All the BBC BASIC commands are supported with the exception of the colour, graphics and sound commands, but I understand there are moves afoot to link these with the colour graphics boards eventually.

It would be interesting to hear from other Gemini users who have BBC BASIC(Z80) and to see a review of it in 80-BUS NEWS. I would write one myself but have no word processing facilities as yet.

It is interesting that there has never been an advertisement for BBC BASIC(Z80) in 80-BUS NEWS and indeed no mention of it. I do not read many computer mags, I have only ever seen it once advertised there and then only in a very small space.

I can only say that I would recommend this language to anyone and feel that Dave's comments were a little derogatory. Speaking as someone who has only ever used BASIC, I found it easy to use, powerful and challenging. I agree with him that the actual BBC machine is next to useless. Having tried to write a useful program on one I found that with the DOS and Hi-Res mode in place, there was about 7K of free memory left and a 2-D array I had to use meant there was no room to run the program. The same program transferred to my Gemini ran perfectly well, giving the same screen image without Hi-Res just on the IVC and there was still 30K free memory left to play with!

Many thanks for an excellent publication even if it is often frustratingly late!

Yours truly, Chris Hellen, Colchester

A CHEAP AND CHEERFUL EPROM ERASER

P. D. COKER

There are a number of ways in which redundant EPROMS can be erased before reprogramming - such as exposing the uncovered window to bright sunlight for a few days or for a rather shorter time to short-wave ultra-violet radiation. It should even be possible to use a series of flashes from a xenon photographic flash tube although I haven't tried this method. All commonly avaliable EPROM erasers tend to use a quartz UV tube and the design which follows uses this method, allowing up to 6 devices to be erased at one time.

The time taken to erase an existing program from an EPROM is dependent upon the wavelength and intensity of the radiation which is employed, the type of EPROM and, to a lesser extent, upon the age and type of device; generally, an exposure time of 15 - 20 minutes with a standard 8 watt tube 1" from the EPROM is sufficient although double this time can be given without undue damage being caused. A shorter time will not fully erase the contents to FF (hex). Some EPROMS may take much longer than 20 minutes, due to their method of manufacture, but these are the exception rather than the rule and can sometimes apply to some Texas types. Sadists who erase their EPROMS with the aid of a combined UV/heat lamp are asking for trouble since the devices will be at risk of death from heatstroke - and unless they take precautions, their eyesight may be affected permanently and adversely by the high level of UV radiation (up to 300 watts).

A programmed EPROM has a pattern of charges placed in its memory locations (which are effectively twin gate MOS FETs), where, owing to the excellent insulating power of the silicon dioxide layer between the two gate electrodes, it can be retained for very long periods of time (in excess of 10 years according to most manufacturers). The effect of UV light is to allow this charge to leak away - which is why a programmed EPROM must have the `window' covered; it is made out of a high-silica glass which allows UV light to pass through - most ordinary types of glass block the short wavelength (253.7 nm) ultra-violet. Additionally, EPROMS don't like high temperatures (above 70 deg. C) which is another good reason for keeping them cool while erasing them and for fitting your computing engine with a decent fan if it acts as a part-time convector heater.

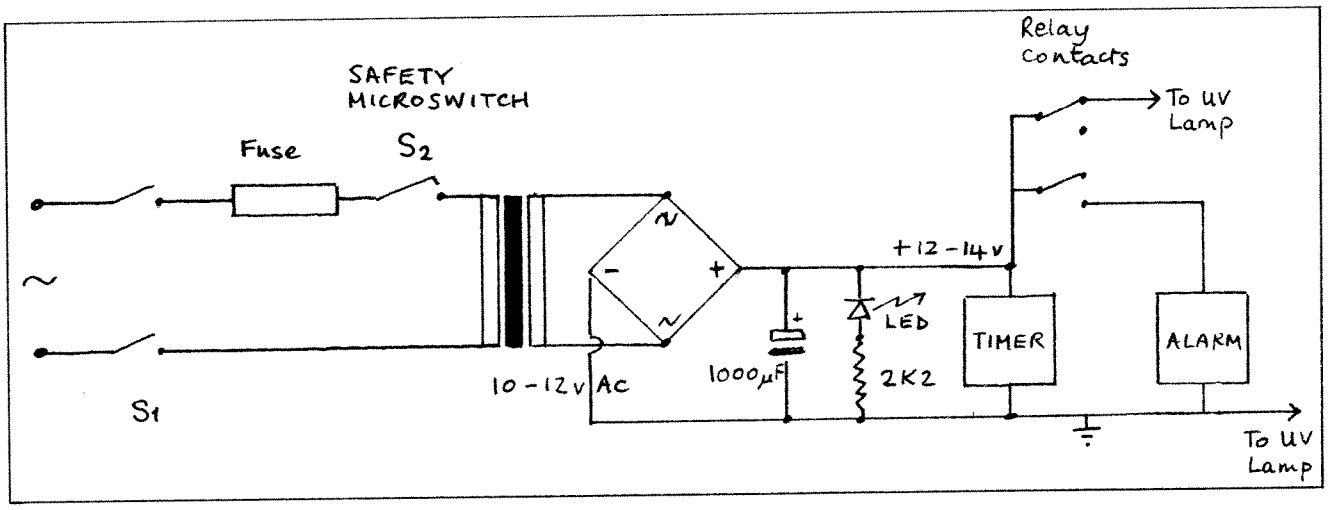
A typical device in the 27XX range requires a radiation dose of about 15 Joules/sq. cm (1 Joule is 1 Watt/second) at an intensity of about 0.012 Watts/sq.cm and for a new UV tube, this would be given by about 20 minutes exposure at 1" from the light source. Tubes lose emission intensity as they age but this is not likely to be much of a problem in this application since the exposure time can be increased if needed. The only drawback is that in order to check the state of erasure, the devices must be taken out of the eraser and checked in the programmer, and replaced if necessary for another dose of sun tan! A device which will not erase properly is likely to have one or more of the following attributes:

a. Made by Texas - try a longer erasure time (up to 2 hours).
b. a dirty window - clean off the muck with a solvent and try again.
c. open or short-circuited - had it been plugged in the wrong way round when it was last used?

If the last is true, then throw it away. Incidentally, new, unused EPROMS should be blank, with FF in all locations. If not, your dealer should change

it since erasure followed by programming may not give a satisfactory result. It is also recommended by most manufacturers that an erased device should be allowed to cool down for a time (20 - 30 minutes) before it is reprogrammed since this improves the long-term stability of the data. A device which has inevitably been heated up in the erasure process will probably program satisfactorily but may not retain all the information for the guaranteed period.

The design of this eraser is very simple and additional features can be added very easily. Many camping and motorists' shops sell single (12") tube, battery operated fluorescent lights for about £5 - £6 which will operate off either 6 or 12 volts DC (I used the 12 volt type in my prototype). These include a ringing choke single transistor convertor and usually take about 1 amp. Watford Electronics (and some other component suppliers) sell 12" UV tubes for about £9 which are a plug in replacement for the standard tube which is supplied in the light. A simple 12 volt 1.5 amp power supply can be constructed for a small sum and with the aid of a light-tight box and a microswitch to act as a safety power-off interlock, a basic but serviceable eraser is yours for less than half the price of a commercial equivalent.

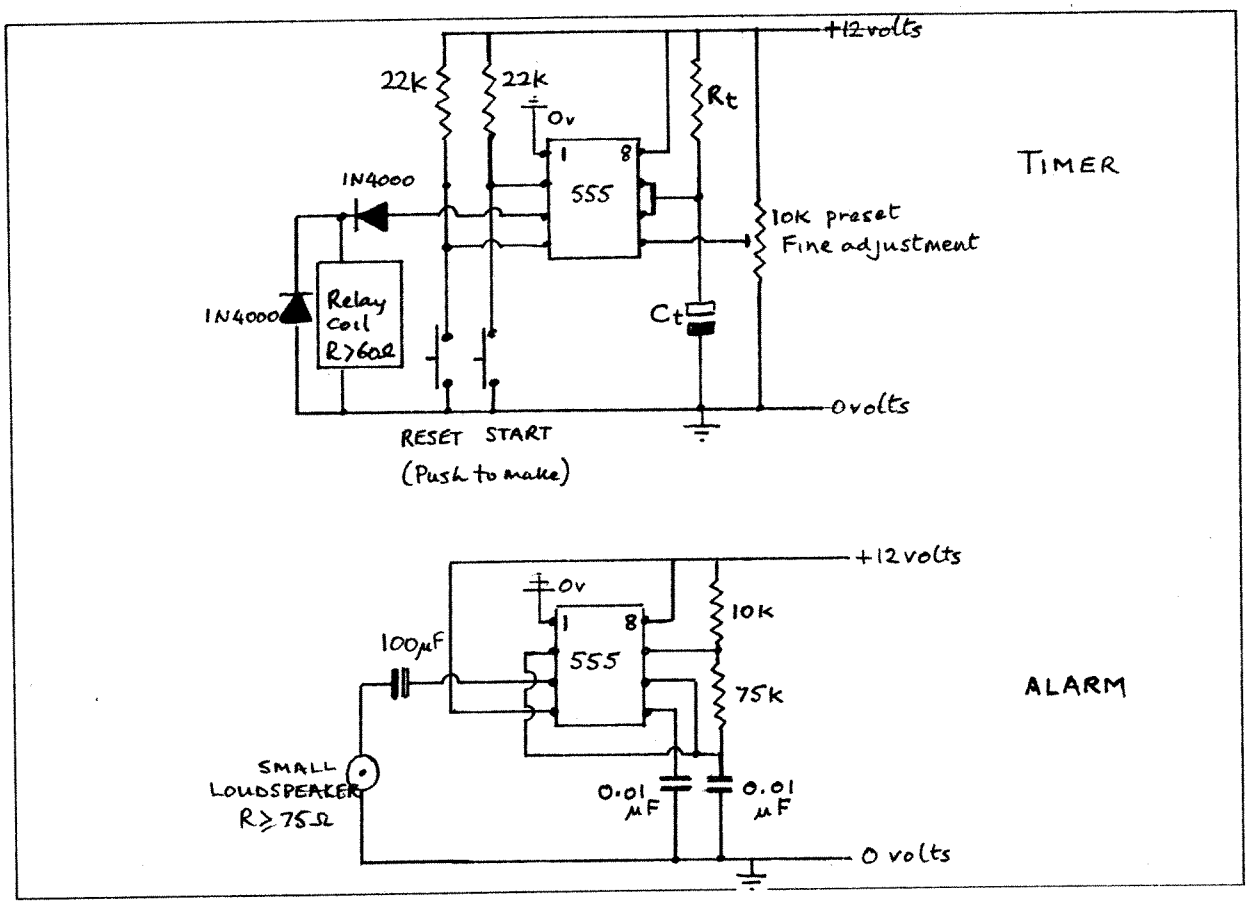


The PSU need not be stabilised or heavily smoothed (1000 - 2000 uF is adequate) and it is best to use a light-tight metal box, earthed to avoid problems with static electricity, into which the PSU and modified light can be fitted. The purpose of the microswitch which is connected in the mains live lead and positioned so that it can only pass current when the box is closed is to avoid excessive doses of UV radiation damaging the operator's eyes and it should NOT be omitted under any circumstances. Mains connections MUST also be well insulated.

Most of these camping lights have a metal chassis, and a flimsy plastic case and removable diffuser which are not needed for the eraser - and should be mounted so that the tube is about 1.5" above the tray which will hold the EPROMS. These should be placed with their pins embedded in a piece of conductive foam or into a piece of polystyrene tile covered with aluminium foil. Either method shorts out the pins and prevents damage to the device during irradiation. The device carrier should be earthed to the case.

BELLS AND WHISTLES

For the absent minded, some sort of timer is needed, and for simplicity, one with a fixed period of 10 or 20 minutes before it switches off the supply will be described. Some sort of audible indication is a good idea, particularly for those enthusiasts who are deeply into meditation (or assembly language).



The timer and audible warning can be made up from a couple of 555 timer i.c.s, one connected as a long period monostable pulse generator and the other as an astable multivibrator. The pulse generator operates a relay with two pairs of SPCO contacts (coil resistance greater than about 60 ohms) and allows one set to supply current to the lamp for the predetermined period (10 or 20 minutes). The 10k preset resistor may require some slight adjustment to compensate for the tolerances of the timing capacitor and resistor. For a period of 10 minutes, Ct is 47 uF and Rt is 11.5 megohms while for 20 minutes, Ct is 100 uF and Rt is 11 megohms; these values are probably accurate enough for most purposes. At the conclusion of the timing period, the relay de-energises and the other set of contacts supply current to the astable. A tone of about 1kHz is produced by the high impedance loudspeaker or a ceramic resonator which will continue until the power supply is switched off or the timer reset. The 555 ic is very suitable for these applications since it can operate a load of up to 200 mA directly and works well from a 12 volt supply line.

A more sophisticated version which enables the user to check on the state of erasure was featured recently in Elektor (no. 108 - April 1984) - the magazine whose projects usually work first time!

Doctor Dark's Diary – Episode 22.

Chaining programs in Hisoft Pascal.

One of the great things about Poly-Data's Compas Pascal is that it lets you chain programs using the "execute" procedure. Hisoft Pascal does not have this useful facility built in, so I have hacked (at last, Heloise, a correct usage of the verb "to hack"!) a way round the problem, which I hope will be found useful by all ambitious programmers sooner or later. It uses one of the features of CP/M to work its crafty trick, and it was a devil of a job to make it work properly at first. When I originally had the idea, I thought it would be possible to poke the name of the next command file into the command buffer, and then just return to CP/M, but that didn't work. CP/M just wiped the buffer, and sat there with its usual smug "A>" prompt. The second attempt worked, eventually, and is based on the way each line of a "submit" file is stored on disk as a file full of commands is processed. If you save a file called "$$$.SUB" on a disk in drive B: and then put it in drive A: you will see that CP/M reads the file and tries to use it as commands. It also erases the file. To create such a file without the bother of using an editor, it should be possible to use PIP, as on the Sirius I use at work, where "PIP BLABLA.TXT=CON:" puts whatever you type into the file BLABLA.TXT until you type a control Z. For some reason, this will not work on my system. (Now that I come to think of it, I may have been using MS-DOS, where you use "COPY CON: BLABLA.TXT", which explains why it refuses to work on Marvin!) So, I wrote a few files using an editor instead, and remembered all the trouble I used to have trying to get "submit" to work, before I gave up trying to use it altogether. I may be wrong, but I do not think I have ever seen it said that the commands in submit files have to have spaces in front of them. Neither have I ever read that command lines in Submit files need to be padded out with spaces the way I have done below. They must, though, with my CP/M. I wonder why? Anyway, here are a couple of programs:-

```
PROGRAM chainone;
VAR
  filel : TEXT;
  i : INTEGER;
BEGIN
  WRITELN('PROGRAM CHAIN ONE RUNNING');
  REWRITE(filel,'  $$$      .SUB');
  WRITE(filel,' CHAINTWO');
  {It won't work without the next line!}
  FOR i := 1 TO 24 DO WRITE(filel,' ');
  WRITELN(filel)
END.

PROGRAM chaintwo;
VAR
  filel : TEXT;
  i : INTEGER;
BEGIN
  WRITELN('PROGRAM CHAIN TWO RUNNING');
  REWRITE(filel,'  $$$      .SUB');
  WRITE(filel,' CHAINONE');
  FOR i := 1 TO 24 DO WRITE(filel,' ');
  WRITELN(filel)
END.
```

If you compile them, and run either, you will see them chaining each other alternately for ages. Unless you were not on drive A:, of course. CP/M insists on this, unfortunately, so I would like to know how to make drives A: and P: (my MAP-80 ramdisk) change their names by fiddling with the BIOS. If it can be done by some sort of poke from a Pascal program, that would be extra neat. Anybody know how?

Another little problem now arises. Programs that are so huge that they have to be broken up and chain each other are going to have lots of variables in them, which will be lost when CP/M takes over at the end of each program. Easy! Make sure the program saves the ones the other programs will need as a disk file. Compas lets you do it an easier way, you have to be using global variables, which are all the same for each program, and declared in the same order, and they will still be there when the next program is loaded, it says in the manual. I don't think it sounds like a very safe way to go about things, as putting a new variable in one program will mean altering all of the other programs in a system, and if you forget, and try to run them, all hell will be let loose, as programs start assuming one variable to be another. So I will use files. In fact, given the speed of ramdisks, and the fact that the file you pass the data in can be on any drive, this is the easy way to do it. As it happens, there is a lot to be said, as your programs grow ever bigger, for designing them to use files anyway, rather than arrays. It gives you more room, although it does need the sort of fast access Winchesters and ramdisks give you.

And now the same thing in Assembler!

Just to show it can be done, here is one half of the same thing in assembler. To make the other half, you want almost the same file again, with the exceptions that the message in line 40 should refer to Program chain 2, and the file name in line 44 should be altered to CHAINONE. The space before the file name is vital. I should still like to hear from anyone who knows why this is...

```
                        1 SYSTEM EQU  £0005
                        2 FCB    EQU  £5C
                        3 TBUF   EQU  £80
                        4 CR     EQU  £0D
                        5 LF     EQU  £0A
                        6 EOF    EQU  £1A
                        7 ;Say which program is running.
0100 0E09               8        LD   C 9
0102 114301             9        LD   DE PROGID
0105 CD0500            10        CALL SYSTEM
                       11 ;Make a file control block.
0108 218301            12        LD   HL NEWFCB
010B 115C00            13        LD   DE FCB
010E 012400            14        LD   BC 36
0111 EDB0              15        LDIR
                       16 ;Create a file with that name.
0113 0E13              17        LD   C 19 ;DELETE FILE
0115 115C00            18        LD   DE FCB
0118 CD0500            19        CALL SYSTEM
011B 0E16              20        LD   C 22 ;MAKE FILE
011D 115C00            21        LD   DE FCB
0120 CD0500            22        CALL SYSTEM
                       23 ;Transfer file contents to buffer.
0123 215E01            24        LD   HL MSG
0126 118000            25        LD   DE TBUF
0129 012500            26        LD   BC MSGEND-MSG+1
012C EDB0              27        LDIR
                       28 ;Write buffer contents to disk.
012E 0E15              29        LD   C 21 ;WRITE SEQUENTIAL
0130 115C00            30        LD   DE FCB
0133 CD0500            31        CALL SYSTEM
                       32 ;Close the file or it will vanish!
0136 0E10              33        LD   C 16 ;CLOSE FILE
0138 115C00            34        LD   DE FCB
013B CD0500            35        CALL SYSTEM
                       36 ;Return to CP/M.
013E 0E00              37        LD   C 0
0140 CD0500            38        CALL SYSTEM
                       39
0143 50                40 PROGID DEFM "Program chain 1 running."
015B 0D                41        DEFB CR LF
015D 24                42        DEFM "$"
                       43
015E 20                44 MSG    DEFM " CHAINTWO"
0167 20                45        DEFM "                "
017F 0D                46        DEFB CR LF EOF
0182 00                47 MSGEND DEFB 0
                       48
0183 00                49 NEWFCB DEFB 0          SUB"
0184 24                50        DEFM "$$$
018F 00                51        DEFB 0 0 0 0 0 0 0 0 0 0 0
0197 00                52        DEFB 0 0 0 0 0 0 0 0 0 0 0
019F 00                53        DEFB 0 0 0 0 0 0 0 0 0 0 0
                       54
```

Note that this program is very "basic" in that at no point does it check to see if any disk errors have happened, which in my opinion is not at all good practice. The reason I am allowing myself to get away with this sloppiness is that it has saved a lot of space. Be sure to put error checking in any serious program using these techniques, or you stand to lose serious files!

More Sneaky Tricks in Hisoft Pascal.

Suppose you want your program to erase a file. There is no way of doing that in standard Pascal, unless some committee has extended it without telling any of us! Hisoft Pascal has the useful predefined function "CP/M(V1,V2)", which interfaces directly to the CP/M routines in the BIOS that you can normally only get at with machine code programs. It puts the number V1 MOD 256 in register C, and V2 in register pair DE, then calls address £0005, in precisely the way you would program such an act in assembler. You would, wouldn't you? Any result that CP/M returns will be passed as the value of the function. The first program will erase a file. You don't have to specify the file name in the program, as V2 can be the name of a variable. As it stands, the program will only erase files called "TEST.DAT", but you can experiment to your heart's content. (Or until the daft thing erases an important file, at which time you will need a file recovery program! Do not make the mistake of saving the file recovery program on the disk you are trying to repair!)

```
PROGRAM ERASE;
CONST
  FCB = £5C; {File Control Block}
VAR
  I, DUMMY : INTEGER;
BEGIN
  FOR I := 0 TO 35 DO POKE(FCB+I,0);
  POKE(FCB+1,'TEST    DAT');
  DUMMY := CPM(19,FCB)
END.
```

If there was a returned value, it would be in the variable DUMMY, which you would be able to test to see if the erasure had been done properly, but I have not bothered. The next program changes the name of a file. Again, there is no reason why the names of the files should be constants. Experiment more!

```
PROGRAM RENAME;
CONST
  FCB = £5C;
VAR
  I, DUMMY : INTEGER;
BEGIN
  FOR I := 0 TO 35 DO POKE(FCB+I,0);
  POKE(FCB+1,'TEST    DAT');
  POKE(FCB+17,'TEST    BAK');
  DUMMY := CPM(23,FCB)
END.
```

I am still thinking about how to add a random file facility to Hisoft Pascal. So, presumably, are Hisoft, as they announced that they intended to produce one some time ago.

END.

A Pascal Print Routine

by P. A. Forrester

Dr. Dark has frequently extolled the virtues of Pascal as a programming language and I admit to sharing his enthusiasm; however, the programs which have been published in 80-BUS News so far have been mainly for illustrative purposes rather than to achieve anything useful. The following program is offered in the hope that it may be useful to members experimenting with BLS Pascal. The excellent editor provided with this compiler is slightly flawed by there being no simple method of printing the contents of the source buffer on a serial printer equivalent to that offered by the ZEAP U command. The recommended way is to mark the beginning and end of the buffer with control symbols and then direct the output to a user-defined print program. I think that a better solution is to use a separate program which gives a more complete control of the print process. My solution is to keep a copy of the compiled code generated by the following program at the top end of the memory and call it whenever I wish to print the source code of a new program.

The program assumes that the source code starts at 4000H and uses the pointer at 0C82H to locate the end. When called, the printer BUSY line, which is connected to bit 7 of Port 0, is checked to ensure that it is ready to receive a character. The edit buffer only uses 0DH for carriage-return (CR), so the program has to add an 0AH to produce a line feed (LF). It also counts the number of CRs transmitted and when this equals the value held in 'lines.page', a number of CRLFs is sent which is determined by the value stored in 'spaces'; this gives automatic pagination. The values can be set at the start of the program or the default values of 60 and 8 used. Printing can be stopped temporarily by pressing the space-bar or aborted by ESC. As the text is transmitted, the procedure 'Writewindow' displays the next 32 characters in a ticker-tape fashion in the centre of the screen and gives a good indication that all is progressing well. The procedure produces a slight delay to the print rate of an Epson FX-80 run at full speed when written in Pascal. This can be overcome by using a machine-code version called as a CODE routine; the ZEAP segment shows how this works. I hope the workings of the program can be followed from the source code. A further refinement would be to provide a means of printing only part of the source buffer, either by asking for start and finish addresses, or by inserting markers into the text. However, the program then starts to become quite complex by the time you have devised a means of entering numbers in hex or a suitable means of coping with markers which stray into the negative integer region.

```
; This code is used by the PASCAL
; program PASCALPRINT as the CODE
; Procedure Writewindow(pointer);
; it pokes 20H characters to the screen
; starting at the current pointer position
; to create a moving 'ticker tape' display
;
;
LEN     EQU £20  ; length of screen window
SCRN    EQU £910 ; start of screen window
WSP     EQU £C92 ; workspace stack pointer
;
```

```
LD HL (WSP)
LD DE -2
ADD HL DE    ; get pointer
LD E (HL)    ; load (HL)
INC HL       ;      into HL
LD D (HL)
EX DE HL
LD DE SCRN
LD BC LEN
LDIR         ; copy to
RET          ;      screen
```

```
PROGRAM PASCALPRINT;

{This program sends the edit buffer of a PASCAL program to a serial printer
 first checking on the busy line connected to Port 0 bit 7. After 'lines.page'
 lines, a new page is produced by outputting 'spaces' lines feeds. Printing
 can be stopped temporarily by pressing the space bar, and discontinued by
 pressing Escape. The text being sent to the printer is displayed in a 'ticker
 tape' window on the screen. The compiled program must have the PASCAL run-
 time system resident in RAM to run properly. Note there is a problem in
 dealing with large files where the size extends to $8000 and beyond, since
 then the integer then represents a negative number. A solution is to print
 the file to $7FFF and then to print the second half using negative numbers}

VAR        screen,count,lines.page,spaces,bufferend:INTEGER;
           Ans      :STRING[1];

PROCEDURE CLS;
CODE $EF,$C,0; {clear screen using NasSys RST 28H}

{Check that printer buffer is not full}
PROCEDURE buffer;
BEGIN
WHILE inp(0) AND $80 <> $80 DO {nothing}
END;

PROCEDURE ready; {Check serial port is ready to send}
BEGIN WHILE inp(2) AND 64<>64 DO; buffer END;

PROCEDURE Delay; VAR I:INTEGER;
BEGIN FOR I:=1 TO 10000 DO END;

PROCEDURE Writewindow(point:INTEGER);
VAR screen,I:INTEGER;
BEGIN
  screen:=$910; {set start of screen window }
  FOR I:=0 TO $20 DO BEGIN IF mem[point+I]=$D
                           THEN mem[screen+I]:= 7 {bell}
                           ELSE mem[screen+I]:=mem[point+I]
                     END
END;

{Alternative version of Writewindow using machine code insert
PROCEDURE Writewindow(pointer:INTEGER);
CODE    $2A,$92,$C,$11,$FE,$FF,$19,$5E,$23,$56,$EB,
       $11,$10,$9,1,$20,0,$ED,$B0;
}

PROCEDURE Printloop(pointer:INTEGER);
VAR char,K:INTEGER;
BEGIN
      WHILE keyboard=$20 DO{nothing}; {stop while Spacebar pressed}
      char:=mem[pointer];
      IF char=$D THEN
              BEGIN
                IF count = lines.page {end of page}
                   THEN
                    BEGIN
                      count:=1; {reset counter}
                      FOR K:=0 TO spaces DO
                         BEGIN
                            ready;out(1,$D);ready;out(1,$A)
                         END;
                    END
                   ELSE
                    BEGIN
                       count:=count + 1; ready;
                       out(1,$D);ready;out(1,$A)
                    END
              END
             ELSE
               BEGIN
                 ready; Writewindow(pointer);
                 ready; out(1,char)
               END
END; {of Procedure Printloop}

PROCEDURE Printbounds(start,finish:INTEGER);
VAR I:INTEGER;
BEGIN
I:=start;
WHILE (keyboard<>$1B) AND (I<=finish) DO
   BEGIN
     Printloop(I); I:=I+1
   END
END;

BEGIN {main body of program}
  screen:=$910;
  lines.page:=60; spaces:=8; count:=1;
  CLS; WRITELN;WRITELN;WRITELN;WRITE('     ');
  WRITELN('Do you wish to change lines/page from 60');
  WRITE('     ');
  WRITE('or page spacing from 8?     Ans Y or N ');
  READ(Ans); IF Ans='Y' THEN
      BEGIN
        WRITELN;
        WRITELN;WRITE('     ');WRITE('Enter new Lines/page value ');
        READLN(lines.page);
        WRITELN;WRITE('     ');WRITE('Enter new space value ');
        READLN(spaces); CLS;
      END;

  Delay; CLS;
  bufferend:=mem[$C82] + 256*mem[$C83];
  IF mem[$C83] AND $80=$80
   THEN {the buffer extends beyond $7FFF}
    BEGIN
     Printbounds($4000,$7FFE); Printloop($7FFF);
     Printbounds($8000,bufferend)
    END
   ELSE
     Printbounds($4000,bufferend);

END.
```

NASCOMS ON THE RAMPAGE!

(or What to Do with the GM862 and a Nascom CPU)

by Warren Williams

INTRODUCTION

The Gemini GM862 is a 256K RAM board that supports Page-Mode addressing when used with Nascom 2 or Gemini GM811 CPU boards, memory mapped Extended Addressing when used with the Gemini GM813 CPU/RAM board, and Direct addressing within the 512K 80-BUS address range when used with the Gemini GM888 8088 CPU board. Although its advanced features are intended for use with current (and future) Gemini 8 bit and 16 bit products, owners of more venerable Nasbus/80-BUS machines can also put it to good use as a Virtual or RAM disk - given that they already have at least one Disk Drive in operation. The advent of 256K and 512K memory boards has led to used 64K GM802 boards being offered for sale at modest prices as professional users (and fanatical amateurs) upgrade their machines. This means that `normal´ Nascomaniacs can get to experiment with Paged RAM at relatively low cost. This article attempts to provide a few tips on installing the GM862 or GM802 as a RAMDISK on a Nascom 2 that are absent from, or perhaps beyond the scope of the original Instruction Manuals.

Having aquired a Gemini GM862 256K RAM board [Ed. - Mr. Williams won it in a contest, but perhaps is too modest to say so!] the task remained of implementing it on either of my trusty Nascom/Gemini computers. Each of these has a Nascom 2 CPU, GM812 IVC, GM809 FDC and (originally) a GM802 64K RAM board, plus interchangeable I/O, EPROM, and Programmer facilities. CP/M or Nas-Sys and various DOSs are available on both by means of switches. One machine is housed in a Kenilworth case while the other is built into a Vero card frame. Currently the 256K board is installed in the Kenilworth console, and both 64Ks are used in the Vero rack, providing respectively 180K and 60K of RAMDISK with 60K CP/M operating systems. However, to achieve this happy state a few problems had to be overcome, as shown below.

DOCUMENTATION and HARDWARE

First a quick look at the GM862 and its documentation. Mine was an early production sample, and the original documentation consisted of photocopied A4 sheets without a circuit diagram; however an additional A5 booklet and A3 size circuit diagrams were supplied within about two weeks of my posting the request slip thoughtfully provided by Gemini. The Manual omits detailed discussion of how the circuits work but concentrates strongly on the setting of the two 8 bit DIL switches and drops carefully concealed hints on other matters. The information is accurate and some thought has been put into presenting the various complex modes and options available. The trick is to pick out ALL the parts vital to your own requirements and avoid bending your brain on the rest. WARNING! There are (were?) vital omissions about Nascom usage.

As for the board itself; construction was well up to standard and all the RAM and a few other selected i.c.s are socketed. It has proved to work reliably on both a friend´s Gemini and my own two Nascoms. On the Gemini, the GM862 worked first time without a hitch in Extended Addressing mode when the appropriate onboard switches were set according to the detailed manual, and the software was configured to suit. Extended (memory mapped) addressing capability is not available on the Nascom 2 CPU board, and Page Mode operation

is the only authorised method of implementing GM862/802 RAM on this system. Explicit examples are shown in the GM862 Manual of how to select each 64K page via switch IC1, disable Extended addressing and enable Page mode via poles 4 & 5 of switch IC2, and disable the 4K/8K 'Common Area' via poles 2 & 3 so that an entire 64K memory bank is switched each time a different page is accessed. As existing software for Nascom/Gemini Page Mode operation already provides simultaneous Read/Write to the top of each memory bank or page, a hardwired Common Area is NOT required to run CP/M 2 etc. on the GM862. Future developments like CP/M PLUS may need this facility however, and dedicated experimenters could find it interesting.

A HARDWARE FIX

When the GM862's DIL switches were carefully set up as per the manual to provide 4 pages of 64K, the Nascom refused to work under CP/M and other DOSs, or even good old Nas-Sys. It would run all systems as usual with the old 802 board without page mode selected, but would not boot CP/M when this board was set to read and write as page zero. Under Nas-Sys it seemed that none of the new board's RAM was being addressed, while the old 64K board could be addressed when setup as any of the four pages. I suspected that the GM862 was not responding to the 80BUS /RAMDIS signal but without a circuit diagram was disinclined to poke about with over £300 worth of brand new hardware. Nevertheless I could have kicked myself when our esteemed Editor told me brightly some days later that a pull-up resistor was needed on (you've guessed it) the /RAMDIS line when using the 256K board with a Nascom. All Gemini CPU boards have pull-ups, on some (open collector) bus lines, which the original Nas-Bus specification expects to find on the auxilliary board/s. Note also that /NASIO, /NASMEM and DBDR signals are not provided on the GM862. Gemini have provided the mounting holes and pads for the necessary 4k7 (4.7k ohm) resistor quite close to the /RAMDIS edge connector (pin No.9) and the +5V termination pad of electrolytic capacitor C6, but for some strange reason have not laid tracks between the resistor pads and these points. Only short links are needed however, and the layout allows the resistor to be inserted and its leads bent and soldered neatly to the capacitor pad and edge pin track.

MODIFYING SIMON

With the resistor fitted, each 64K page of the 256K RAM could be accessed via Nas-Sys and, with page 0 selected, would support programs and DOSs running under Nas-Sys. CP/M still refused to boot-up however, and the clue to the solution of that problem is to be found in the manual under Note 1 of Section 4 - SOFTWARE IMPLICATIONS. This states that the GM862 Page-Mode latch is cleared on RESET, disabling its memory until any given page is selected by writing to Port 0FFh. The standard Gemini CP/M auto-boot EPROMS and versions 2.0 or later of RP/M for the GM811 and GM813 CPU boards, include initialisation code to select Page 0 on start-up or RESET. Earlier software and firmware for the Nascom has no such code and so it must be modified. The routine is simple and fortunately there is ample space within SIMON to accomodate it. My method was to change the first jump table instruction at the begining of SIMON to go to the Page 0 selection routine, and when this is completed a jump is made back to the original target address. The A register and Flags are saved and recovered to allow the routine to be portable i.e. to be used in other locations or boot systems. The assembled patch listing for SIMON 1 located at F000h is shown below and can be easily be adapted as required:-

```
        START   EQU     F015H           ;Start of code after
                                        ;the jump table in SIMON 1

F000    C313F3          JP PAGEON       ;Changed jump to select Page 1

F313    F5      PAGEON: PUSH AF         ;Save A and Flags
F314    3E11            LD A,11H        ;Load System Page No.(0) in A
F316    D3FF            OUT (FFH),A     ;Select Page 0
F318    F1              POP AF          ;Recover A and Flags
F319    C315F0          JP START        ;Go boot CP/M
                        END
```

CONFIGURING SOFTWARE

There are other ways to do it, but the above code works on both of my Nascoms, and CP/M, CCPZ etc, boot up and run with either GM862 or GM802 RAM boards installed. Having got thus far, there was still the job of configuring CP/M to actually drive the other memory page/s as RAMDISK. If Gemini's CONFIG.COM is available for the version of CP/M used, then it's a doddle. One simply selects Menu option 8 (7 on some versions) and if you are using the GM862, have SIMON in EPROM at F000h, and run 60K CP/M or whatever, enter 3 for the number of extra pages and 60 for the Kilobyte size of each page. By selecting the appropriate number of pages, CONFIG can accommodate up to three additional 64K GM802 boards in the same manner, but do NOT specify more boards or pages of RAM than are installed because the system will crash and/or fail to boot. Test the system by typing M: to select the RAMDISK, when the M> or M0> prompt should appear. A DIR command should respond with a NO FILE message. Return to `normal' via an A: command and type STAT M: when the response should indicate between 172K and 178K `bytes free' if three 60K pages are in use. If only one 60K page is in use the response will be a (disappointing) 52K to 58K depending upon the size of the BIOS and the allocated RAMDISK Directory. Try PIPing and STATing programs around between Disk and RAM to ensure that all is well. SWEEP may also be used, but the versions written in high level code hang-up if zero sized files are transmitted and exhibit a few other minor bugs. However NSWEEP 2.07, a later Z80 code version, has behaved faultlessly with RAMDISKs. The proven configuration on the Disk system tracks may now be SYSGENed onto other disks or the configuration process can be repeated as required.

SYS ASSEMBLY

To activate Page mode RAMDISK operation under later versions of SYS requires that module SYSB1 be re-assembled using M80 with the appropriate software switches set as follows:-

```
vdisk   equ     true            ;Virtual disk i.e. RAMDISK support
vboot   equ true(or false)      ;Boot from virtual disk (best set to
                                ;false at first, but recommended)
vdrive  equ   12(or15)          ;Drive letter (12=M,15=P) as preferred
vflip   equ     false           ;Else on own your head be it!

vgem    equ     true            ;Set map32, map4 and g833 false

vbmem   equ     60              ;60K when SIMON is in EPROM at F000H
vnum    equ     2(1,or3)        ;Number of boards/pages see * below
                                ;may be 3 only if SYS is modified
vnorm   equ     1(or2)          ;System page, normally 0 (1=Pg0)
```

* For some reason, which may be historical, SYS supports only 2 Virtual memory or RAMDISK pages, and these are Pages 2 and 3 rather than 1 and 2. If GM802 boards are used their read and write address switches MUST be set accordingly. Because GM862 boards are always supplied fully populated, the standard switch settings as shown in the Manual for a Nascom with four memory pages (one system page and three virtual pages) may always be used. However unless SYS is modified, only two pages of RAMDISK will be available, reducing its effective size to 118K or less.

SYS MODIFICATIONS

As such restrictions were intolerable, SYS (Version 18.1) was modified so that it could be assembled to run with 1, 2, or 3 Pages of RAMDISK and also allow these pages to be contiguous. That is, starting with the system memory on page 0 ,the first disk page will be 1, the next 2, and the last 3. Besides enabling all three pages of the GM862 to be accessed, this hardware configuration is compatible with that employed with Gemini CONFIG.COM so that either operating system can be used without resetting switches or unplugging boards. The modifications, made to Modules SYSB3 and SYSB8 of SYS 18.1, are shown below. Although, sadly, no longer marketed, there must be quite a few SYSes of differing ages and pedigree out there in the vast void of space and time, and hopefully this information will also be of use with some at least of these:-

N.B. To aid identification of the line to be modified, the nearest previous label name is stated, followed by the (inclusive) number of lines to count from the label. Blank lines are NOT included in the count. Generally only one or two parameters need to be changed, making it easy to identify the correct line. Additions are inserted BEFORE the original line indicated.

SYSB3 Modifications - if vgem

```
change  vinit:+11       to      ld a,vnorm*16+02h       ;Read from page 1
change  vinit:+18       to      ld a,0e0h+vnorm         ;Write pages 1, 2, 3
change  be2: +7         to      ld a,20h+vnorm          ;Write to page 1
```

SYSB8 Modifications - if vgem

```
change  dpbv:+5         to      defb 0e0h,0             ;ALO 3 directory blks.
                                                        ;change is optional
change  vb: +4          to      ld a,vnorm*16+02h       ;Read from page 1
add at  vr: +8                  ld a,vnorm*16+02h       ;Read page 1
                                jr c,vr2                ;Next page ?
add at  vw: +8                  ld a,20h+vnorm          ;Write page 1
                                jr c,vw2                ;Next page ?
```

HEAT POWER and LIGHT

That should have been the end of the story, but during the first few weeks of operation both my systems suffered odd failures at times. The original 3A power supplies were overloaded by the additional memory, even though the supplies had been `boosted' to deliver more power and had yielded stirling service in the past. Two 6A switchmode units had to be bought (sob sob) and were carefully wired in, but there were still a few odd `happenings' which were finally exorcized by adding more shielding around the disk drives in the Vero rack and fitting an 80mm cooling fan in the console machine. Since then all has been sweetness and light.

USES and ABUSES

The 172 - 178K RAMDISK capacity now available from the GM862 is just about large enough for `serious' applications requiring lots of data transfer, where its fast and silent operation is a definite asset. Also, using the RAMDISK as the working drive, having first loaded the appropriate software, helps prevent real disks becoming cluttered with all sorts of rubbish in less important applications. The single GM802 RAMDISK with its limited 52 - 58K capacity may not be of so much `real' use, but is quite a good educational toy, enabling techniques to be developed and casual remarks about having 128K of memory aboard to be dropped to flatten SCROTUM and BEEBLBOX braggarts. I intend, sometime, to arrange for SIMON to be switched out of the memory map on completion of the CP/M boot so as to enable entire 64K, rather than 60K, pages to be selected. However memory space will still be occupied by the RAMDISK directory and `common area' needed by Page Mode so the actual gain in capacity will be modest.

THE FINAL WORD

Although the Gemini GM862 is relatively expensive compared say with some 96 t.p.i. disk drives, it is attractive to GALAXY and MULTIBOARD users, who can fully exploit its more advanced features. Nascom users looking to upgrade in the future, but who need RAMDISK now, should also consider this board. Used GM802 boards, sometimes available at about half their original price, offer a way for the amateur to get into RAMDISK (and go on the Rampage).

A Review of the Microcode BP14C Backplane

by D. G. Richards

Way, way back in May of '84, my faithful old Nascom-1 crashed (again). At the time I was using a database program which was up-dating a very large data file; needless to say, all data was lost forever. Now it's things like this that get me real peeved, so after doing the rounds of all the local sources of electrical interference, e.g. the fridge, CB rigs, etc, I went next door and gave their cat a good kicking, just to be on the safe side (thanks for the warning about cats, Dr Dark!).

When all the above proved fruitless, I decided it was time to take a close look at my Nascom, you know the old saying, "When you've eliminated everything you can think of, whatever's left is probably to blame." The only thing left that I hadn't checked was my computer. Was my Nascom at fault (shock, horror, probe)?

The answer to the above was - NO. My Nascom checked out OK, as did the other boards in the system, except for the motherboard. Now, my motherboard is of the Vero-board type (when I started expanding my system back in '81, this board was considered hi-tech) and over the years I'd done some really horrible things to it, like adding 1,000 uF capacitors to the power rails, and soldering 16 SWG copper wire onto the power rails in an attempt to make my fast failing 3 Amp PSU last just a little while longer.

The result of all this soldering on my motherboard was that instead of it being flat, it was more banana shaped, and, infact, it was a real `mother'. The time had come to put my dear old motherboard out to grass (no, no, I said "out to grass" not "onto grass", twit).

While reading the latest 1984 issue of this rag, i.e. June 1978, I stumbled upon an advert for the Microcode BP14C 14-slot Backplane. Succumbing to the blurb in the advert, I ordered one, pronto. A few days later, back came the goods - in a BIG box. Actually, the only thing bigger than this board is its price tag (67 quid).

To get the board to fit into my case (hand-built out of ex-county-council plywood) I had to lose one of the board slots. I tell you folks, I cried. After paying out 67 quid for the thing, here I was lopping an inch off the end with a rusty hack-saw! Now, onto the board itself. The board is of a very high standard (it should be for the price), measures 15" by 8" and is 2.5 mm thick. The board is covered on both sides with green solder resist (I do hear that Chris Blackmore is having one custom-built with blue solder resist to match his MAP-80 RAM board which is, so I've been told, really amazingly blue.....).

The sheer thickness of the board may cause a few problems to the unwary as the pins on the standard 77-way 80-BUS edge connectors just barely emerge from t'other side of the board for soldering. This fact hit me for six as the pins of the edge conectors on my old Vero motherboad were cut short so's they wouldn't touch anything nasty, like the mains input plug. So, after forking out all that bread buying the new motherboard I had to spend another 25 notes on five new edge connectors. After splashing out close on £100 on a new backplane with five edge connectors, guess what? My Nascom still does the odd, un-programmed action. Ah well, that's progress, I suppose!

And now a word for all those die-hards, who, like me, intend to use the Microcode backplane with a Nascom-1.

As most owners of expanded N-1's know, the Nascom buffer board has a ribbon cable sticking out of its component side which connects said board to the 43-way edge connector on the N-1 board. The layout of the buffer board is such that the best way of mounting it in the system is with its component side facing the Nascom's component side (motherboard driven from the left-hand end).

One of the main features of the Microcode backplane is that it terminates all active 80-BUS signals into balanced RC filter networks. These filter networks have to placed as far as possible from the CPU board in order to function properly, i.e. the buffer board plugs into the backplane at the end opposite the end containing the filter networks.

Now, as with most things to do with computers, it's at this point that all the fun begins. The tracks of the Microcode backplane are laid out in such a way that when the buffer board is plugged into it, the ribbon cable on the buffer board faces AWAY from the Nascom board. To overcome this problem, I re-soldered the ribbon cable onto the solder side of the board so that it again faced the Nascom board. Owing to this piece of board surgery, the ribbon cable also needs to be twisted through 180 degrees so that the 43-way edge connector mates with the Nascom board.

Although the Microcode backplane is of a high standard and works like a dream, I don't think it's worth the 67 quid I paid for it. You pays yer money and yer takes yer choice .....

Henry's CCPZ, BDOSZ, & Utility Disks

by W. H. Turner

Recently, feeling rich, I sent off to Henry's for BDOSZ, CCPZ and their Incredible CP/M Utilities Disk. Having used them now for a month or so I thought a few comments on how I found them might be interesting.

I expected BDOSZ to have lots of free space to add extra goodies, but it didn't. I feel this shows how compact 8080 code can be. Now that BDOSZ is installed in my system, I am only aware of the change when I run out of disk space or get a select error. There's not much more to say about it except that it's nice to be able to look through the code when you want to find something out.

CCPZ, on the other hand, has had a fundamental effect on my way of working. For a start, all my commonly used programs (e.g. PIP, SUBMIT, ERQ etc.) now reside on drive A and I do all my work on drive B with a disk for each activity (Pascal, assembler, correspondence etc.). I use Gemini's "Auto Execute" facility to log in drive B on reset and although most programs are to be found on drive A, it only takes a second for CCPZ to determine that the file is not on drive B. Files are transferred from one working disk to another via drive A. I have patched SUBMIT and EXSUB to put the $$$.SUB file on drive A where CCPZ expects to find it and Wordstar to find its overlay files there too.

Initially there was a problem with CCPZ as after the first warm boot, all I got was the message:

Wrong system size/No system on this disk

I have Gemini Bios vers 2.2 2-DM and I traced the problem to the code at BIOS+E3H where it compares the byte at CCP+4 to E0H which it is in the original CCP. I changed this test to comparing CCP+3 to C3H (jp).

I can't leave the subject of CCPZ without saying that I find all the new facilities incredibly useful. Its a bit like the TV Times, I didn't realise you could get so much in it!

Finally to Henry's Incredible Utilities. These are a mixed bag, some being extremely useful. Although some of the programs have been mentioned before in the magazine, I have never seen a review of the entire contents before. The programs come in source form or as COM files (some as both) and there is so much on the disk that all the sources are in compressed form but by following the directions all can be retrieved.

The disk contains the following programs:

a) Cataloguing suite
b) COMP (file compare program)
c) CPU (Z80 processor test)
d) Program dating suite (Its all done using bit 7 of the letters in the file name)
e) CRC (File CRC generation)
f) DU (Disk patch utility)
g) ERQ (Selective erase)
h) FILEMAP (Displays file maps)

i) FIND (Search file for hex or ASCII strings. There is an error in the documentation here as it requires '??' for wild hex bytes.)
j)File compress/uncompress suite
k) Memory testing utilities
l) PIP (with all the bugs fixed)
m) FINDBAD (Collects bad sectors into single file)
n) SD (Sorted Directory listings with file sizes)
o) SPEED (Check for speed of PERTEC disk drives)
p) STATUS (Displays RAM size, TPA size, BIOS start etc.)
q) SUB (or EXSUB - replacement for SUBMIT)
r) TRANSLAT (Converts 8080 to Z80 mnemonics)
s) SWEEP (For moving files from disk to disk)
t) UN (Removes protection from protected MBASIC files)
u) UNERASE (Unerase files)
v) UNSPOOL (Provides background file printing)
w) HELP (Online help facility)
x) WHATSNEW (Displays changes to disk directory)

My only critisism of these centres on UNSPOOL, for which only source is supplied, as it would not compile with M80 until tweaked a little. This is one program that should have been supplied as a COM file. Having said that, the source of UNSPOOL supplies the solution of how to write programs that can relocate themselves (and also reveals the purpose of the 'r' on every second line of SYS listings).

Well that was a brief review! Not so much a review as a list of what's on the disk. Considering the number of useful programs packed onto the disk it must be value for money as are both CCPZ and BDOSZ.

This magazine also represents excellent value for money (a Superbrain friend of mine eyed it enviously). Readers, support your magazine with more letters and articles; Editor, what has happened to all the issues this year?

Private Advertisements

Hantarex RGB standard res. monitor suitable for Nascom AVC in very good condition and little used. Nascom AVC Model B also little used. Offers for either to C Tame, Nuneaton (0203) 373260.

Nascom 2 in Kenilworth case with fan. 32K RAM A board. NasSys 1, NasPen, ZEAP, NasGra. £100. ASR33 teletype also available on stand in acoustic case. £30. Telephone Ampthill (0525) 404572 (Beds.).

Nascom 2 with CP/M and Nas-Dos, MAP80 RAM card (full 256K), Nascom AVC with CP/M and Nas-Dos software, Nascom FDC, one SSDD disk drive in twin drive box and PSU, Gemini 5A PSU and 8 slot backplane in rack frame. Gemini EPROM Board with all the usual firmware, Bits and PC's EPROM Programmer - lots of spare 2708's and 2716's, PAL Encoder Card (needs attention). Nascom 1 with 32K Ram A Card and Basic Rom, Cottis-Blandford interface, 3A PSU (needs attention). Lots of disk and tape software and all the manuals. Offers for whole or parts to: Dr David Plews Tel. Steeton (0535) 51511.

UPGRADING A NASCOM 1 KEYBOARD (or Spaghetti Junction revisited!) P. D. COKER

There must be a fair number of Nascom 1 owners who have wondered how to connect up a series of Licon keys in the 10 spare holes in the mounting frame so that they actually do something useful. Compared with the Nascom 2 keyboard, the original Nascom 1 keyboard lacks the following keys:

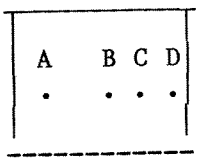
Graphics, Cursor Home, Cursor Up, Down, Right and Left, Control, right and left square brackets and a Shift key next to the 'Z' key.

Provision of the facilities offered by these would make many aspects of the Nascom 1 a great deal more user-friendly, particularly when screen-editing facilities are required or supported as in Nas-Sys or Naspen, for example.

A short article by Derek Brough in INMC80 issue 2 gave details of the connections for an additional Shift key but there were still 9 holes to fill in on the keyboard. I acquired some keys and tops from B & L Micros (Lucas/Nascom were a bit sniffy about orders less than £25); new Licon keys are a bit pricey (about £1.50, tops about 60p but Nascom dealers may have some ex-equipment, or you may be able to acquire a defunct Nascom keyboard and carefully (with a solder sucker and patience), extract what you need. You will also need 1 each of 22R, 1K and 2K2 ohm 0.25 watt 5 % resistors and some fine, PVC insulated wire (NOT enamelled). Note that this modification will nullify any guarantee on the keyboard and DON'T attempt it unless you have a steady hand. Licon keys must be used - the usual keyboard 'N.O.' switch won't work in this circuit. After the conversion the keyboard will be equivalent in its functions to a Nascom 2. A short article by David Pears dealing with this keyboard appeared in INMC80 issue 5 and shows the majority of the 'new' keys connected to bit D6 of the data bus and their relationship to the driver lines from the 74145 (IC 5 on both types of board).

There were at least 2 issues of keyboard - issue A boards have tracks running very close to some of the new key pins which will have to be cut and re-routed as necessary but there are convenient gaps in all the right places on the issue B boards. The issue letter is in the top left hand corner of the underside of the board next to the Reset key. Having turned the board over to find out its issue letter (and gender), arrange it so that the space bar is nearest you and place a ruler along the pins of the top (numeric) keys. Draw a line about 2" long at the left-hand end with a fine point indelible marker - with luck this line will enable you to locate the pin positions for the square bracket keys! Draw a similar line 1" long on the right hand end of the next (QWERTY) row of keys. The third row needs a 1" line on the left hand and right hand ends of this (ASDF) row. Another 1" line on the right hand end of the ZXCV row follows and finally a line on either side of the Space bar pins, exactly 0.75" below the ZXCV row, extending 4.5" either side of the key pins.

If you look at the base of a Licon key, you will find that the terminal pins are assymetrically arranged, as in the diagram:



Pins A and B are the input and output of a sensing coil and C and D are the input and output of a driver coil which is pulsed by an output from a 74145 BCD - decimal decoder/driver. In the unexpanded keyboard, the keys are arranged in a matrix of 8 driver lines (Dr0 - Dr7) and 6 sensing lines (S0 - S5) and one of the 48 points is not used. In the expanded version, an additional sense line is employed (S6) to give 56 points in the matrix, with the new Shift key added to give a total of 57. The sense outputs are processed to give data bits 0 - 5 on the CPU bus (or bits 0 - 6 in the expanded version). A browse through a circuit diagram of the keyboard (CK9009, April 1978) showed that half of IC1 (CA3086 transistor array) and half of IC3 (7400 quad 2-input NAND gate) are available. The buffering for the sense lines is provided by single transistors within each of the two 3086 arrays (IC1 and 9), and up to two new sense lines could be connected with the unused part of IC1. The circuit diagram also shows each buffered sense line and associated pull-up resistor connected to individual flip-flops each formed from 2 of the NAND gates in a 7400 (ICs 2,7 and 8). It is a simple matter to implement another flip-flop for bit 6 from the two spare gates available in IC3 and connect this to pin 7 of the keyboard socket. Fitting in the three resistors for the collector pull-up, emitter bias and base of the IC transistor buffer calls for some care.

The Nascom 1 keyboard circuit diagram shows a particular layout of characters against sense/driver intersections in the key matrix but the actual order is slightly different (Murphy's Law). For the issue B board, the following order was found: (issue A boards may be different)

	Drivers							
	0	1	2	3	4	5	6	7
Sense								
0	-	5	6	7	8	9	0	4
3	@	T	Y	U	I	O	P	R
1	BS	H	J	K	L	;	:	G
2	RET	B	N	M	,	.	/	V
5	SHFT	X	Z	S	A	Q	1	SPC
4	not used	F	D	E	W	3	2	C

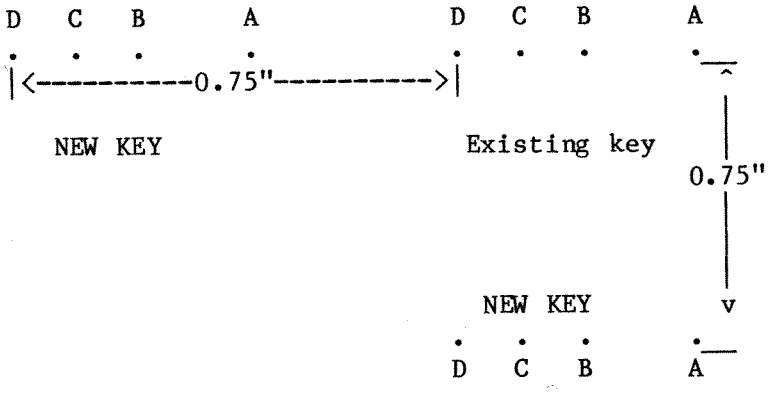
The sense outputs are connected to IC9 and 1 as follows:

Sense	IC	Pin	IC pin for output to data bus
0	9	9	11
3	9	6	8
1	9	4	5
2	9	12	14
5	1	12	14
4	1	4	5
New sense line 6	1	9	11

The Driver lines are supplied from IC5 pins 1 - 7 and 9 (corresponding to Dr0 - Dr7). The new sense line could have been connected to IC1 pin 6 but the circuit layout means that the connection to pin 9 was easier to arrange.

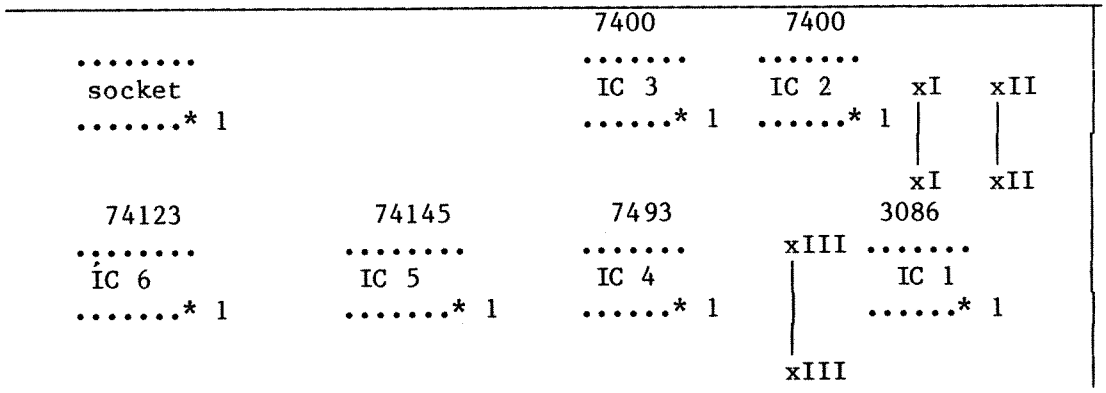
Board surgery can now commence! The holes for the new pins must be drilled accurately with a 1 mm bit and for most of the holes required, simply match up

pin 'D' of an existing key with the new key's 'D' pin and ensure that the distance from the existing pin D to the new key's pin D is exactly 0.75"; mark this accurately on the line which you previously drew. Holding a new key, mark the positions of the other 3 pins carefully. Centre punch each point lightly and drill all 4 holes.



Key drilling guide

Insert the new key through the frame and check that the pins go through the holes. If all is well, carry on, otherwise get your eyes checked! The cursor control keys are normally fitted on the same level as the Space bar and the easiest way of locating the pin positions is to use a setsquare and draw lines down from the corresponding keys in the row immediately above. When all 40 holes have been drilled, breathe a sigh of relief and drill 6 more! These are the mounting holes for the resistors and are positioned as in the sketch.



I.C. and new resistor locations on keyboard

* 1	pin 1 of each i.c.
x - - - - x	1 mm holes
I - - - - I	2K2 resistor
II - - - II	22R resistor
III - - III	1K resistor

In order to obtain the correct bit pattern on the databus the new keys must be inserted in the driver and sense circuitry in appropriate places; after some trial and error, a suitable sequence was worked out which involved some track cutting. The allocation of keys on the keyboard, viewed from the underside is as follows:

(Unshifted key symbols shown)

] [- 0 9 8 7 6 5 4 3 2 1

BS @ P O I U Y T R E W Q Graph

CH NL : ; L K J H G F D S E Control

(old) Shift / . , M N B V C X Z Shift (new)

Right Down S P A C E Up Left

Mark these on the board and DOUBLE check....

The next stage is to cut some of the sense and driver lines; a sharp scalpel or model-maker's knife should be used - CHECK BEFORE CUTTING -take out a piece of track about 1 - 2 mm wide - carefully. Each key has 4 pins and with the space bar facing you, they are in the order D, C, B, gap, A. Cut the track leaving the following pins:

KEY	PIN
Z	A
W	B
Q	C
P	C
Z	C
X	C
K	C
L	C
R	D
-	D
NL	D

Having cut the tracks, check again (and use the Elastoplast if necessary!). If all is well, insert the new keys and ensure that 4 pins from each protrude through the board and that the key body clicks into the frame. Now for the spaghetti....

Any fine insulated wire can be used; I used 'wire-wrap' wire which happened to be handy. While no need has been found to used twisted pairs of wires, it is important to keep the wiring neat and close to the board. Solder each end carefully and quickly to avoid damaging the keys. Wire exactly as shown below. If you reverse the connections to either of the coils in the key, the sense line pulse produced when the key is operated will be of the wrong polarity and will be ignored.

Join pin	A of	] key	to pin	B of	[key
	A	[		A	G
	A	CH		B	]
	A	Graph		B	Left
	A	Left		B	Up
	A	Up		B	Down
	A	Down		B	Right
	A	Right		B	CH
	A	Z		B	NEW Shift
	A	NEW Shift		B	S
	B	W		A	Control
	B	Control		A	E
	C	X		D	Up
	C	Up		D	F
	C	Z		D	Left
	C	Left		D	D
	C	K		D	Down
	C	Down		D	M
	C	L		D	Right
	C	Right		D	,
	C	Q		D	Graph
	C	Graph		D	3
	C	P		D	[
	C	[		D	:
	D	R		C	]
	D	]		C	4
	D	NL		C	CH
	D	CH		C	BS
	D	-		C	Control
	D	Control		C	NEW Shift
	D	NEW Shift		IC 5	pin 1

Check your wiring carefully for shorts and accuracy and, when satisfied, stick it down to the circuit board with spots of adhesive.

Refer to the IC layout; insert the resistors as follows. In holes I, a 2K2 ohm, in holes II a 22R resistor and in holes III a 1K resistor. Solder one end of the 2K2 and 22R resistors to their respective neighbours; the other end of the 2K2 resistor to pin 11 of IC 1, the other end of the 22R resistor to pin 10 of IC 1. Then solder one end of the 1K resistor to pin 9 of IC 1 and the other end to pin B of the Graph key; use sleeving on the exposed wires. Check for possible solder bridges and breathe a sigh of relief.

The final part of the conversion consists of wiring up the new flip-flop. Locate IC 3; link pins 10 and 11, then 8 and 12 and finally 6 and 13. Solder a wire from IC 3 pin 9 to IC 1 pin 11 and another from IC 3 pin 11 to pin 7 of the keyboard socket. Check again for solder bridges. Fit the caps on the new keys, connect to the Nascom 1, keep fingers crossed and switch on. If all is well, the original keys will work and the new keys will carry out their functions as described. If not, switch off and recheck the wiring and track cuts - or get someone else to do so.

This is a fairly intricate conversion and it is best to do it in two stages to avoid mistakes through fatigue. The results are worthwhile and the two keyboards that have been converted are working very satisfactorily on Nascom 1's for several months.

Aunt Agatha's Agony Column

by D. W. Parkinson

RAM-A

A letter from Mr Piper of Sheffield re. problems with his Nascom system: "Can you explain why selecting start addresses on LSW1 (other than NAS-SYS) no longer works after having done 2716 mods on the RAM-A card. `J' command and others except `Y' work Ok. Setting LSW1 to B000 start address fills the screen with garbage. I can select A000 where I have Ext. Basic, but this seems to put an inhibit on the PIO and I can't use the printer. With the switches set to NAS-SYS everything (except `Y') works Ok."

He doesn't give any detail of the modifications, but looking back through my pile of back issues I came across an article from the INMC days [1]. This is basically a list of cut-and-strap operations on the RAM-A card, and as my only copy of the RAM-A circuit diagram dates from the days after Nascom got its reducing photo-copier, I've no idea what the modifications are. (Most IC references and pin numbers are unreadable on the miniscule diagram, and not all ICs have their pin connections numbered.) However let's start with the obvious possibilities and then start guessing.

The Nascom 2 `Reset-Jump' circuit works by forcing the high four address lines (A12-A15) of the Bus to the value set on switches 1-4 of LSW1. This over-ride of the normal addressing persists for only the first three bytes fetched from memory, thereafter the four high address lines of the Z80 are passed to the Bus. Thus any EPROM that is going to be entered this way must start with a JP instruction to set the Z80's program counter correctly so that execution continues in the EPROM once the forcing of the high address lines is removed. (i.e. if the EPROM is to be located at B000, and entered by a `jump-on-reset', then the first instruction must be JP B003, or at least JP <somewhere-in-the-EPROM> in order to prevent the Z80 fetching its next instruction from address 0003.)

There is no way that modifications to the RAM card can affect the operation of the `Reset-Jump' circuit, and as the `Y' command (JP B000) does not work for Mr Piper either, it is obvious that the problem must lie in the hardware modifications. Looking at the RAM-A circuit diagram, one thing is immediately apparent: The address lines to the EPROMs are not buffered from the Bus. This means that any error here may well affect the operation of the RAM, or any other board connected into the Bus. (This includes the byte-wide sockets on the Nascom 2.) In replacing 2708s by 2716s, pin 19, which used to be connected to the +12v supply, is now used for A10. I was suprised that I could find no mention in [1] of removing any decoupling capacitors from this line. (I assume the +12v supply line is decoupled around the EPROMs on RAM-A!) If these are left behind they will have a rather detrimental effect on A10 to put it mildly. (Similarly if you move on to use 2732s, ensure that there are no decoupling capacitors left on pin 21 (ex -5v supply) which becomes A11.)

If there are no capacitors present, run a few tests to try to isolate the fault. Do memory tests check out the 32k RAM perfectly? Does BASIC run perfectly? If the answer is yes, then you can rule out a catastrophic wiring error on the address lines. The next step is to insert a 2716 with known contents into one of the sockets. (e.g. a copy of NAS-SYS.) Can you find it in the memory map? If you can, when you `T'abulate are the contents correct? (Use a program to compare it against the original NAS-SYS at 0000.) If you are partially successful here, then you may be able to work out where the error

exists by what it is in the correct place, and what isn't. If you can't even find the EPROM, then the fault may lie in the /CS decoding, or the enable to the output buffer. It is at times like these that a 'scope (or logic probe) is useful, especially when used in conjunction with a complex test program like:

```
0D00  2100B0        LD HL,0B000h      ; Load address of ROM
0D03  7E      LOOP: LD A,(HL)         ; Access the ROM
0D04  18FD          JR LOOP           ; Continue
```

A tight loop like this allows you to synchronise a 'scope easily, and to follow various signals through the circuit.

If everthing checks out to this point, but problems still exist in executing programs in the EPROMs, then it looks like its a case of switching on wait-states. (Or else revising the modifications to generate an earlier /CS signal.)

CLIMAX PROBLEMS (No it's **not** that type of Agony column)

It has come to my notice that somebody has problems in trying to use the Climax card in a Gemini Network system. After initial suspicions of the software, it was finally realised that some Network systems had been shipped fitted with a particular model of power-supply which did not provide a -5v supply - a fact that didn't matter with the supplied boards (GM813 CPU + RAM, IVC, etc). The Climax, however, uses 4116 RAMS (which require a -5V supply), and so when this oversight was rectified.... This struck a chord with me, because I can remember doing something similar several years ago. I hadn't bothered to wire the -12v supply through the backplane because I didn't use it, and then some months later came to connect a serial printer to the RS232 interface....

[1] Zienkiewicz A.O. "Modification to `TYPE A' RAM board", INMC80-5 Oct-Dec 1981

More Private Advertisements

Spiral Systems PCG for Nascom 2. Ready to plug into main board. Gives hi-res display 256 x 210. Complete with documentation and software on disk or tape. Has given good service for 1 year, but now replaced by AVC. £30 or offers. Contact C.R. CASE, tel Rugby (0788) 832560.

Nascom Single Disk Drive + Controller + NAS-DOS + Disks, Nascom AVC, Enhanced BASIC, GM817 75W switch-mode Power Supply. Offers for any of these items to Andy Briggs on (0203) 404006.

Vero Case to house N2, RAM-B and PSU. Also NAS-SYS 1 and NASPEN PROMs and original documentation. Offers to P Dishart, 01-904 3212.

Epson Printer FX80 as new £280. 2-off 32K CMOS battery-backed RAM boards for £100 each. Phone L Harold, 0702 555619 (Southend) evenings.

Polydos File Update

by M. J. R. Gibbs

This is a program for directly updating files on disk for a Gemini GM809/GM815 system with Polydos 2.0.

This allows the user to update copies of a file stored on several disks without having to enter all the normal commands which can cause problems especialy if you are like me and sometimes forget to use the 'NEW' command (I have spent many hours trying to sort out the chaos that this causes). The required program/data is loaded into RAM between £1000 and £C000 either by using the Polydos 'Read' command or by Assembling the program directly into RAM.

This program is loaded into RAM at £0C80 and executed at £0C80, the user is asked for the file name, file extension and the RAM location for the new code. The Program asks you to insert the disks and press the 'Enter' key, the disk directory is read and the filename checked and, should it exist, the file is replaced with the new version. The old file is completely overwritten and this means that the number of sectors used by the old file are replaced by the program stored in RAM. The user should be careful in ensuring that the new version is not going to require more sectors on disk than the original otherwise the whole of the new program will not be saved. I normally save programs that are common to several disks with a few sectors more than required to allow for expansion, this does not normally cause a problem as I have found that the maximum number of fifty files means that the disk is not often full before the directory is used up. This program can update a disk with a full directory because the directory is left unaltered.

Should the program not exist on the disk inserted a suitable message is printed out telling the user and the disk is left unaltered.

The Program informs the user of the start disk sector and the number of sectors updated and the RAM location used for the update. Should there be any disk errors then a message is printed out and the program repeats the inital menu asking the user to insert a new disk. The one thing that I have found is that it is easy to forget to remove any write protect tabs and this does not cause a problem as you simply take the disk out remove the tab and reinsert the disk (do not forget to replace the tab afterwards).

A word of warning when testing this program especially when entering it from the dump listing, which is always prone to errors, you are strongly advised to test it on a copy of an existing disk because it can overwrite irreplaceable portions of disk if anything goes wrong.

I suggest that you proceed as follows:-

1) Enter the program and save it on disk using Polydos. Use the 'Read' routine to load a file from the disk to say £1000 in RAM.
2) Examine and make a note of the number of sectors that this file uses.
3) Use SZAP to examine the next file on the disk and make a note of the first few bytes.
4) Change the first few bytes of the file loaded into RAM with the NAS-SYS 'M' (modify command).
5) Run this program to update the disk and check the user information for the correct sector location, number of sectors and RAM location.
6) Use SZAP to verify that the original file has been updated correctly. Also that the file following is not corrupted.

Below is a full Assembly listing, a sorted symbol table and a dump of the program using a modified version of the disk dump published in Vol.1 iss. 2 of 80-BUS NEWS (the numbers on the left hand side are the RAM locations).

```
0F35                    ;*************************
0F35                    ;*  DISK UPDATE ROUTINE  *
0F35                    ;* --------------------- *
0F35                    ;*                       *
0F35                    ;* M.J.R GIBBS 29-12-82  *
0F35                    ;*************************
0018          NASSYS EQU   £18
0028          PRS    EQU   £28
0030          ROUT   EQU   £30
0030                                ;NASSYS COMMANDS
007B          BLINK  EQU   £7B
0068          B2HEX  EQU   £68
0065          CRT    EQU   £65
0063          INLIN  EQU   £63
0064          NUM    EQU   £64
005B          RETNAS EQU   £5B
0066          TBCD3  EQU   £66
005D          TDEL   EQU   £5D
005D                                ;DISK COMMANDS
0080          ZDSIZE EQU   £80      ;DISK SIZE
0081          ZDRD   EQU   £81      ;READ
0082          ZDWR   EQU   £82      ;WRITE
0083          ZRDIR  EQU   £83      ;READ DIRECTORY
0084          ZWDIR  EQU   £84      ;WRITE DIRECTORY
0085          ZCFS   EQU   £85      ;CONVET FILE SPECIFIER
0087          ZENTER EQU   £87      ;UPDATE DIRECTORY
0088          ZCOV   EQU   £88      ;CALL OVERLAY
0089          ZCOVR  EQU   £89      ;CALL OVERLAY RESTORE
008A          ZCKER  EQU   £8A      ;CHECK FOR ERRORS
008B          ZCBRK  EQU   £8B      ;CHECK FOR BREAK
008C          ZCFMA  EQU   £8C      ;ABORT COMMAND MODE
0086          ZLOOK  EQU   £86      ;LOOKUP FILE DIRECTORY
0063          ZINLIN EQU   £63      ;MONITOR INPUT LINE
0063                                ;DISK LOCATIONS
C055          S1FCB  EQU   £C055
C069          S2FCB  EQU   £C069
C001          DDRV   EQU   £C001    ;DIRECTORY DRIVE
C414          NXTSEC EQU   £C414    ;NEXT FREE SECTOR
C414                                ;DUMMY FCB
C055                 ORG   S1FCB
C055                 LOAD  0
C055          FNAM   DS    8        ;FILE   NAME
C05D          FEXT   DS    2        ;FILE   EXTENSION
C05F          FSFL   DS    1        ;SYSTEM FLAG
C060          FUFL   DS    1        ;USER   FLAG
C061          FSEC   DS    2        ;SECTOR ADDRESS
C063          FNSC   DS    2        ;NUMBER SECTORS
C065          FLDA   DS    2        ;LOAD   ADDRESS
C067          FEXA   DS    2        ;EXEC   ADDRESS
C067                                ;SCREEN ADDRESSES
C067                                ;
080A          LINE1  EQU   £080A
0C29          CURSOR EQU   £0C29
```



```
0C21          NUMV   EQU   £0C21
000C          CLEAR  EQU   £0C
000D          CR     EQU   £0D
001B          ESC    EQU   £1B
001B                                ;
0C80                 ORG   £0C80
0C80                 LOAD  £8000
0C80                                ;
0C80                                ;--- DISK UPDATE ---
0C80                                ;
0C80 CD230E   UPDATE CALL  HEAD     ;WRITE HEADINGS
0C83 EF              RST   PRS
0C84 4469736B
0C88 20557064
0C8C 61746520
0C90 46696C65
0C94 204E616D
0C98 65202E2E
0C9C 2E3A2D20
0CA0 00              DB    'Disk Update File Name ...:- ',0
0CA1 DF              RST   NASSYS
0CA2 63              DB    ZINLIN    ;GET REPLY
0CA3 211C00          LD    HL,28
0CA6 19              ADD   HL,DE     ;HL = A(START OF REPLY)
0CA7 1155C0          LD    DE,FNAM   ;DE = A(S1FCB)
0CAA 010800          LD    BC,8
0CAD EDB0            LDIR            ;LOAD NAME
0CAF EF              RST   PRS
0CB0 0D              DB    CR
0CB1 46696C65
0CB5 20457874
0CB9 656E7369
0CBD 6F6E202E
0CC1 2E2E2E2E
0CC5 2E2E2E2E
0CC9 2E3A2D20
0CCD 00              DB    'File Extension ..........:- ',0
0CCE DF              RST   NASSYS
0CCF 63              DB    INLIN
0CD0 211C00          LD    HL,28
0CD3 19              ADD   HL,DE     ;HL = A(EXT)
0CD4 115DC0          LD    DE,FEXT   ;DE = A(FEXT)
0CD7 010200          LD    BC,2
0CDA EDB0            LDIR            ;LOAD FEXT
0CDC EF              RST   PRS
0CDD 0D              DB    CR
0CDE 52414D20
0CE2 53746172
0CE6 74204164
0CEA 64726573
0CEE 73202E2E
0CF2 2E2E2E2E
0CF6 2E3A2D20
```

```
0CFA 00              DB    'RAM Start Address .....:- ',0
0CFB 2A290C          LD    HL,(CURSOR) ;HL = CURSOR LOCN
0CFE 22290C   ERR10  LD    (CURSOR),HL ;RESET CURSOR
0D01 E5              PUSH  HL          ;KEEP CURSOR LOCN
0D02 DF              RST   NASSYS
0D03 63              DB    INLIN       ;GET  REPLY
0D04 D1              POP   DE          ;DECODE REPLY
0D05 E5              PUSH  HL          ;NUM  MODS HL
0D06 DF              RST   NASSYS
0D07 64              DB    NUM         ;SORT OUT HEX NUMBER
0D08 E1              POP   HL          ;RESTORE HL
0D09 38F3            JR    C,ERR10     ;ERROR??
0D0B 2A210C          LD    HL,(NUMV)   ;HL = NUMBER
0D0E 22350F          LD    (RAMPOS),HL ;KEEP REPLY
0D0E                                   ;
0D0E                                   ;MAIN LOOP FOR LOADING FILES
0D0E                                   ;
0D11 CD6B0E   LOOP   CALL  HEAD2
0D14 DF              RST   NASSYS
0D15 7B              DB    BLINK
0D16 FE1B            CP    ESC
0D18 CA1C0E          JP    Z,DSKEND    ;STOP
0D1B 3EFF            LD    A,£FF
0D1D 3201C0          LD    (DDRV),A    ;FORCE DIRECTORY READ
0D20 0E00            LD    C,0         ;DIRECTORY DRIVE 0
0D22 DF              RST   NASSYS
0D23 83              DB    ZRDIR       ;READ DIRECTORY
0D24 20EB            JR    NZ,LOOP     ;ERROR MAYBE NO DISK
0D26 2155C0          LD    HL,S1FCB    ;HL = A(FCB)
0D29 0630            LD    B,£30       ;LOAD & LOCKED FILES
0D2B DF              RST   NASSYS
0D2C 86              DB    ZLOOK
0D2D 2829            JR    Z,LOADIT
0D2F EF              RST   PRS
0D30 0D0D            DB    CR,CR
0D32 20202020
0D36 2020203C
0D3A 3D3D3D3D
0D3E 2046494C
0D42 45204E4F
0D46 5420464F
0D4A 554E4420
0D4E 3D3D3D3D
0D52 3E00            DB    '       <==== FILE NOT FOUND ====>',0
0D54 DF              RST   NASSYS      ;WAIT TO SHOW RESULT
0D55 5D              DB    TDEL
0D56 18B9            JR    LOOP        ;
0D56                                   ;
0D56                                   ;ALL  OK LOAD DISK
0D56                                   ;SHOW ATTRIBUTES FOR CHECK
0D56                                   ;
0D58 CD230E   LOADIT CALL  HEAD        ;LOAD DISK
0D5B 2A61C0          LD    HL,(FSEC)   ;HL = SECTOR LOCATION
```



```
0D5E EF              RST   PRS
0D5F 20202020
0D63 20202044
0D67 69736B20
0D6B 53656374
0D6F 6F72202E
0D73 2E2E2E2E
0D77 2E2E2E3A
0D7B 3D2000          DB    '       Disk Sector .....:= ',0
0D7E DF              RST   NASSYS
0D7F 66              DB    TBCD3
0D80 2A350F          LD    HL,(RAMPOS) ;HL = RAM POSN-
0D83 EF              RST   PRS
0D84 0D0D            DB    CR,CR
0D86 20202020
0D8A 20202046
0D8E 726F6D20
0D92 52616D20
0D96 2E2E2E2E
0D9A 2E2E2E2E
0D9E 2E2E2E3A
0DA2 3D2000          DB    '       From RAM .....:= ',0
0DA5 DF              RST   NASSYS
0DA6 66              DB    TBCD3
0DA7 3A63C0          LD    A,(FNSC)    ;A  = NUMBER OF SECTORS
0DAA 47              LD    B,A         ;B  = NUMBER OF SECTORS
0DAB EF              RST   PRS
0DAC 0D0D            DB    CR,CR
0DAE 20202020
0DB2 2020204E
0DB6 756D6265
0DBA 72206F66
0DBE 20536563
0DC2 746F7273
0DC6 202E2E3A
0DCA 3D202020
0DCE 00              DB    '       Number of Sectors ..:=   ',0
0DCF 78              LD    A,B         ;A  = NUMBER OF SECTORS
0DD0 DF              RST   NASSYS
0DD1 68              DB    B2HEX
0DD2 C5              PUSH  BC          ;WAIT TO SHOW
0DD3 DF              RST   NASSYS
0DD4 5D              DB    TDEL
0DD5 C1              POP   BC          ;RECOVER BC
0DD6 0E00            LD    C,0         ;C  = DISK DRIVE
0DD8 ED5B61C0        LD    DE,(FSEC)   ;DE = A(SECTOR)
0DDC 2A350F          LD    HL,(RAMPOS) ;HL = A(RAM)
0DDF DF              RST   NASSYS
0DE0 82              DB    ZDWR        ;WRITE IT OUT
0DE1 2003            JR    NZ,ERRORD
0DE3 C3110D          JP    LOOP        ;NEXT
0DE6 F5       ERRORD PUSH  AF          ;KEEP ERROR NUMBER
0DE7 EF              RST   PRS
```

```
0DE8 0D0D4572
0DEC 726F7220
0DF0 3D3D3D3E
0DF4 2000           DB    CR,CR,"Error ===> ",00
0DF6 F1             POP   AF
0DF7 DF             RST   NASSYS
0DF8 88             DB    ZCOV          ;LOAD ERROR OVERLAY
0DF9 456D7367       DB    "Emsg"        ;PRINT MESSAGE
0DFD EF             RST   PRS
0DFE 0D0D           DB    CR,CR
0E00 20202020
0E04 20202050
0E08 72657373
0E0C 2022456E
0E10 74657222
0E14 202000         DB    '       Press "Enter"  ',0
0E17 DF             RST   NASSYS
0E18 7B             DB    BLINK
0E19 C3110D         JP    LOOP
0E1C 3EFF     DSKEND LD   A,£FF         ;FORCE DIR READ
0E1E 3201C0         LD    (DDRV),A
0E21 DF             RST   NASSYS        ;END
0E22 5B             DB    RETNAS        ;
0E23 EF       HEAD  RST   PRS           ;--- SUBROUTINE HEAD ---
0E24 0C             DB    CLEAR
0E25 20202020
0E29 20202020
0E2D 20202020
0E31 20444953
0E35 4B205550
0E39 44415445
0E3D 20555449
0E41 4C495459       DB    '             DISK UPDATE UTILITY'
0E45 0D             DB    CR
0E46 20202020
0E4A 20202020
0E4E 20202020
0E52 202D2D2D
0E56 2D2D2D2D
0E5A 2D2D2D2D
0E5E 2D2D2D2D
0E62 2D2D2D2D       DB    '               ------------------'
0E66 0D0D0D00       DB    CR,CR,CR,0
0E6A C9             RET                 ;
0E6B CD230E   HEAD2 CALL  HEAD          ;--- SUBROUTINE HEAD2 ---
0E6E EF             RST   PRS
0E6F 20202020
0E73 20202046
0E77 696C6520
0E7B 4E616D65
0E7F 202E2E2E
0E83 2E2E2E2E
0E87 2E2E2E3A
```



```
0E8B 2D2000         DB    '       File Name ..........:- ',0
0E8E 2155C0         LD    HL,S1FCB
0E91 0608           LD    B,8
0E93 7E       HEAD10 LD   A,(HL)        ;PRINT FILE NAME
0E94 F7             RST   ROUT
0E95 23             INC   HL            ;INDEX ALONG
0E96 10FB           DJNZ  HEAD10
0E98 EF             RST   PRS
0E99 2E00           DB    '.',0
0E9B 0602           LD    B,2
0E9D 7E       HEAD20 LD   A,(HL)        ;PRINT FILE EXT
0E9E F7             RST   ROUT
0E9F 23             INC   HL
0EA0 10FB           DJNZ  HEAD20
0EA2 EF             RST   PRS
0EA3 0D0D           DB    CR,CR
0EA5 20202020
0EA9 2020204C
0EAD 6F616465
0EB1 64204672
0EB5 6F6D2052
0EB9 414D202E
0EBD 2E2E2E3A
0EC1 2D2000         DB    '       Loaded From RAM ....:- ',0
0EC4 2A350F         LD    HL,(RAMPOS)
0EC7 DF             RST   NASSYS
0EC8 66             DB    TBCD3         ;PRINT IT
0EC9 EF             RST   PRS
0ECA 0D0D           DB    CR,CR
0ECC 20202020
0ED0 20202049
0ED4 6E736572
0ED8 74204469
0EDC 736B2069
0EE0 6E746F20
0EE4 44726976
0EE8 6520302E       DB    '       Insert Disk into Drive 0.'
0EEC 0D             DB    CR
0EED 20202020
0EF1 20202050
0EF5 72657373
0EF9 2022456E
0EFD 74657222
0E01 20776865
0E05 6E207265
0E09 6164792E       DB    '       Press "Enter" when ready.'
0F0D 0D             DB    CR
0F0E 20202020
0F12 20202050
0F16 72657373
0F1A 20224573
0F1E 63222020
0F22 20746F20
```

```
Z2 ASSEMBLY LISTING ..... PAGE  7
-----------------------------------

 0F26 5465726D
 0F2A 696E6174
 0F2E 652E3A2D
 0F32 20              DB    '       Press "Esc"    to Terminate.:- "
 0F33 00              DB    0
 0F34 C9              RET
 0F35 0100      RAMPOS DW    1
```

```
Z2 ASSEMBLY LISTING ..... PAGE  8
-----------------------------------

 B2HEX   0068 BLINK   007B CLEAR   000C CR      000D
 CRT     0065 CURSOR  0C29 DDRV    C001 DSKEND  0E1C
 ERR10   0CFE ERRORD  0DE6 ESC     001B FEXA    C067
 FEXT    C05D FLDA    C065 FNAM    C055 FNSC    C063
 FSEC    C061 FSFL    C05F FUFL    C060 HEAD    0E23
 HEAD10  0E93 HEAD2   0E6B HEAD20  0E9D INLIN   0063
 LINE1   080A LOADIT  0D58 LOOP    0D11 NASSYS  0018
 NUM     0064 NUMV    0C21 NXTSEC  C414 PRS     0028
 RAMPOS  0F35 RETNAS  005B ROUT    0030 S1FCB   C055
 S2FCB   C069 TBCD3   0066 TDEL    005D UPDATE  0C80
 ZCBRK   008B ZCFMA   008C ZCFS    0085 ZCKER   008A
 ZCOV    0088 ZCOVR   0089 ZDRD    0081 ZDSIZE  0080
 ZDWR    0082 ZENTER  0087 ZINLIN  0063 ZLOOK   0086
 ZRDIR   0083 ZWDIR   0084
```

```
DUMP V1.0       UPDATE V1.0                                    PAGE 01

0C80  CD 23 0E EF 44 69 73 6B  20 55 70 64 61 74 65 20  .£..Disk Update
0C90  46 69 6C 65 20 4E 61 6D  65 20 2E 2E 2E 3A 2D 20  File Name ...:-
0CA0  00 DF 63 21 1C 00 19 11  55 C0 01 08 00 ED B0 EF  ..c!....U.......
0CB0  0D 46 69 6C 65 20 45 78  74 65 6E 73 69 6F 6E 20  .File Extension
0CC0  2E 2E 2E 2E 2E 2E 2E 2E  2E 2E 3A 2D 20 00 DF 63  ..........:- ..c
0CD0  21 1C 00 19 11 5D C0 01  02 00 ED B0 EF 0D 52 41  !....]........RA
0CE0  4D 20 53 74 61 72 74 20  41 64 64 72 65 73 73 20  M Start Address
0CF0  2E 2E 2E 2E 2E 2E 2E 3A  2D 20 00 2A 29 0C 22 29  .......:- .*).")
0D00  0C E5 DF 63 D1 E5 DF 64  E1 38 F3 2A 21 0C 22 35  ...c...d.8.*!."5
0D10  0F CD 6B 0E DF 7B FE 1B  CA 1C 0E 3E FF 32 01 C0  ..k..{.....>.2..
0D20  0E 00 DF 83 20 EB 21 55  C0 06 30 DF 86 28 29 EF  .... .!U..0..().
0D30  0D 0D 20 20 20 20 20 20  20 3C 3D 3D 3D 3D 20 46  ..       <==== F
0D40  49 4C 45 20 4E 4F 54 20  46 4F 55 4E 44 20 3D 3D  ILE NOT FOUND ==
0D50  3D 3D 3E 00 DF 5D 18 B9  CD 23 0E 2A 61 C0 EF 20  ==>..]...£.*a..
0D60  20 20 20 20 20 20 44 69  73 6B 20 53 65 63 74 6F        Disk Secto
0D70  72 20 2E 2E 2E 2E 2E 2E  2E 2E 3A 3D 20 00 DF 66  r .........:= ..f
0D80  2A 35 0F EF 0D 0D 20 20  20 20 20 20 20 46 72 6F  *5....        Fro
0D90  6D 20 52 61 6D 20 2E 2E  2E 2E 2E 2E 2E 2E 2E 2E  m Ram ..........
0DA0  2E 3A 3D 20 00 DF 66 3A  63 C0 47 EF 0D 0D 20 20  .:= ..f:c.G...
0DB0  20 20 20 20 20 4E 75 6D  62 65 72 20 6F 66 20 53       Number of S
0DC0  65 63 74 6F 72 73 20 2E  2E 3A 3D 20 20 20 00 78  ectors ..:=    .x
0DD0  DF 68 C5 DF 5D C1 0E 00  ED 5B 61 C0 2A 35 0F DF  .h..]....[a.*5..
0DE0  82 20 03 C3 11 0D F5 EF  0D 0D 45 72 72 6F 72 20  . ........Error
0DF0  3D 3D 3D 3E 20 00 F1 DF  88 45 6D 73 67 EF 0D 0D  ===> ....Emsg...
0E00  20 20 20 20 20 20 20 50  72 65 73 73 20 22 45 6E         Press "En
0E10  74 65 72 22 20 20 00 DF  7B C3 11 0D 3E FF 32 01  ter"  ..{...>.2.
0E20  C0 DF 5B EF 0C 20 20 20  20 20 20 20 20 20 20 20  ..[..
0E30  20 20 44 49 53 4B 20 55  50 44 41 54 45 20 55 54    DISK UPDATE UT
0E40  49 4C 49 54 59 0D 20 20  20 20 20 20 20 20 20 20  ILITY.
0E50  20 20 20 2D 2D 2D 2D 2D  2D 2D 2D 2D 2D 2D 2D 2D     ----------
0E60  2D 2D 2D 2D 2D 2D 0D 0D  0D 00 C9 CD 23 0E EF 20  ------.....£..
0E70  20 20 20 20 20 20 46 69  6C 65 20 4E 61 6D 65 20        File Name
0E80  2E 2E 2E 2E 2E 2E 2E 2E  2E 2E 3A 2D 20 00 21 55  ..........:- .!U
0E90  C0 06 08 7E F7 23 10 FB  EF 2E 00 06 02 7E F7 23  ...~.£.......~.£
0EA0  10 FB EF 0D 0D 20 20 20  20 20 20 20 4C 6F 61 64  .....        Load
0EB0  65 64 20 46 72 6F 6D 20  52 41 4D 20 2E 2E 2E 2E  ed From RAM ....
0EC0  3A 2D 20 00 2A 35 0F DF  66 EF 0D 0D 20 20 20 20  :- .*5..f...
0ED0  20 20 20 49 6E 73 65 72  74 20 44 69 73 6B 20 69     Insert Disk i
0EE0  6E 74 6F 20 44 72 69 76  65 20 30 2E 0D 20 20 20  nto Drive 0..
0EF0  20 20 20 20 50 72 65 73  73 20 22 45 6E 74 65 72      Press "Enter
0F00  22 20 77 68 65 6E 20 72  65 61 64 79 2E 0D 20 20  " when ready..
0F10  20 20 20 20 20 50 72 65  73 73 20 22 45 73 63 22       Press "Esc"
0F20  20 20 20 74 6F 20 54 65  72 6D 69 6E 61 74 65 2E     to Terminate.
0F30  3A 2D 20 00 C9 01 00 00  00 00 00 00 00 00 00 00  :- ............
0F40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0F50  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0F60  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0F70  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

NASCOM CP/M AND ENHANCED BASIC AND OTHER RUBBISH by DR. DAVID PLEWS

I've just got the April issue of 80-BUS News (in November ! - honest I'm not complaining) and feel prompted to write after reading Robin Scadden's letter echoing my disquiet with Lucas Logic and Roger Dowling's article on Nascom Enhanced Basic for Nas-Dos.

It starts back in the New Year after reading the December issue of the Nascom Newsletter and the `advert' about Lucas Logic's CP/M Rev. 2.3. This included details of the 80 column screen available on the AVC and virtual disk on the MAP80 RAM card. I decided to take the plunge and `upgraded' from Nas-Dos to CP/M and purchased Nascom Enhanced (XBASIC) for CP/M. So starts the epic.

I checked with Lucas Logic that all new CP/M purchases were of Rev. 2.3, and they agreed, so I ordered both through the Leeds Computer Centre. You can guess what happened. I got Rev 2.1. Leeds Computer Centre were very good and phoned Lucas Logic who said I would have to send my master disk back. This was done and returned with MDISK.COM, but no documentation despite a note with the disk requesting it. Of course the damn thing didn't work. After a lot of costly phone calls by the Leeds Computer Centre, all to no avail, I wrote to Lucas Logic and eventually received the documentation. Totally useless. MDISK would still not work. So, I sent Lucas Logic a letter asking for help. No reply. Like Robin Scadden I can write scathing letters when necessary, so I sent another letter with a print out of the first 1K of garbage of my MDISK. I soon (yes SOON!) received a nice reply with a disk with a copy of a working MDISK and away I went. It really is very good. The ram card is treated just like a floppy disk from the users point of view and is VERY fast. You get 192K of file space with the fully populated card. One problem though. It doesn't work with my XBASIC !!! My first letter has not been replied to. So it's another unpleasant letter to Lucas Logic. I'll let you know what happens. But it's very bad form when you HAVE to get nasty to get anywhere with people like this. It makes me wonder if they're NHS administrators who've moved to the Private Sector !

Anyway, now I've got that off my chest, what about the rest of Nascom CP/M ? Well it's just like any other. AVCTXT.COM gives AVC users the 80 column screen which is really handy, if not essential for wordprocessing. However, it's a bit slow when scrolling, but very readable because it IS a high resolution graphics card.

There are 47 special keyboard functions which are probably similar to the IVC/SVC plus the ability to use the numeric keys as function keys via the GRAPH key. These are of course user-definable and there's a neat little example program to show you how to do it. However, Lucas Logic let themselves down again. The manual has a table of flags, parameters, etc in it, but when I asked how the cursor position is stored as the manual doesn't make this clear, I get no reply. Can anybody else help ?

Like any CP/M you buy it's got all the dross from Digital Research, but what surprised me most is it's speed. Actually it's lack of it. It's so slow compared to Nas-Dos, but of course not as versatile (?). How to get the most out it for the least investment of money ? - see below.

Now onto XBASIC. For sixty odd quid you get a disk with the ordinary interpreter plus the GXBASIC, the graphics version for the AVC. With AVCTXT sitting at the top of ram to give the 80 column screen (55K System size), the former gives you 35122 bytes of ram, the latter only 27105 bytes ! I won't retramp the areas covered by Roger Dowling, just get straight into the disk handling stuff. Well, it's really a bit of a mess, but somehow Lucas Logic came out with some twists to almost make up for it. I think they've gone down the road of - wait for it - BBC BASIC (sorry, quick, drink a glass of water, or something a bit stonger). I've used MBASIC on a Superbrain and it was quite nice. XBASIC lost me three days figuring out how to use the random access files without screwing up. The relevant commands are -

DRIVE <letter> - selects which drive to be logged in, `A' is drive 0.

CREATE <F>,<SV>,<I> - creates a file with name `F', `SV' sets up buffer space and system variables, `I' determines how long the random record length is. The range of `I' is 0-65535. If `I' is not specified or 0, then the file is deemed for sequential access only. Any existing file with the same name is first deleted and an empty one then created.

OPEN <F>,<SV>,<I> - this is the same as for CREATE. If the file doesn't exist then you get a `No File error'.

CLOSE <SV1..n> - closes the named files. CLOSE alone will close ALL open files.

APPEND <F>,<SV> - this is useful for sticking data onto the end of sequential files without having to read to the end first, so saving a lot of time. It doesn't work with random files as you can specify which record you want to go to any way.

Now comes the daft bit, or B**b method. To get data in and out of files you have to direct the computer with the following -

PRINT £ <SV>,<I>;<variable(s)> - the variable(s) are output to the opened file with system variable name `SV'. ANY output from now on until otherwise redefined will go to that file. So if you forget the drive purrs away quite oblivious to the fact that it's just ruined an evenings work.

INPUT £ <SV>,<I>;<variable(s)> - this is the opposite of PRINT £ (why they couldn't have used DISCIN and DISCOUT or similar is beyond my logical brain). Like PRINT £ if you forget to redefine the input you get one hell of a mess. To redefine you can CLOSE <SV>, or do another PRINT £ <SV>,<I> (or INPUT £ <SV>,<I>) or PRINT £0 (or INPUT £0) so that the file(s) remain open. To backtrack a bit, the `SV' has to be a string variable and Lucas Logic claim it to be a wonderful idea as you can have lots of files open at the same time and because it's a string variable, when not in use the memory is a available for use by something else. Sounds good doesn't it ? So, what took me three days to work out ? Well, for one thing I had to learn to redefine the output and input. Secondly, when using random files the record length is not what you tell the computer. The manual doesn't make it clear that the end of record marker (two bytes) is INCLUDED in the record length. So if you specify a record length of 50 bytes and output 50 bytes, two bytes are stuck on the end. When you then input the file you run out of string space as the whole file is dragged

off the disk at once !! I discovered this after three nights, when in an inebriated state I used DDT on a file that had just dumped itself everywhere but I wanted it to be. (Actually it's been scientifically proven that with a blood alcohol of approximately 12% you're most creative !!)

Why Lucas Logic have chosen this tedious file handling systemy is difficult to determine. I was hoping that XBASIC would do all the file handling for ME. However, this method has a slight advantage. You can concoct a file of fixed length from mainy variables. Prepare it for input with INPUT £ <SV>,<I> alone, and then nibble away at the records using INCH$(N). This speeds things up a bit and you can bung all the data to the variables immediately afterwards.

What about GXBASIC ? Well the commands are the same as the Nas-Dos based G32 and G48 running with Rom Basic except the `SET' prefix is dropped. However, it draws at a much faster rate, and dumping pictures off the disk has been turbocharged ! There is a command to dump graphics from the screen to a printer, but I can't get that to work either. My printer is a Gemini-10X so it's probably not compatible. It doesn't bother me at the moment .... but, perhaps another letter to Lucas Logic ?

So is XBASIC worth sixty quid ? Well had I but known I wouldn't have bought it. If only someone had told me about the CP/M Users Group (UK). This is an `amateur' organisation on similar lines to 80-BUS. Membership is £7.50 p.a. with a really good quarterly journal. But, the real benefit is the software library. Over 250 VOLUMES (YES !!!!) of software in the public domain i.e. NO copyright. Choose a volume, send a disk with £2.00 for copying and postage and it's returned tout de suite. Theorectically, you don't need to buy anything from the `big-boys' again. If you want more information write to -

CP/M USERS GROUP(UK)
72 Mill Road, Hawley, Dartford, Kent, DA2 7RZ

Why haven't our great leaders told us this before ???

So what do I use XBASIC for ? Well, it has a really good on screen editor in the old Nascom tradition, so I use it to write all my letters on ! (If I bought a decent text editor I'd have to throw it away and my fiancee would be very annoyed at the cost.) Actually, this article was prepared on it. The AUTO command keeps churning out the line numbers as needed, then I edited it to something vaguely acceptable to our Editor (oh Lord and Master hear my prayer). Next, saved it to disk as an ASC file and then used EDIT (a much improved ED.COM from the CP/M Users Group) to remove all the line numbers. Not very difficult, tedious, but a damn site easier than using EDIT denoveau (from scratch !).

With regards to Lucas Logic, I personally think that many computer companies will be going to the wall next year, and that Lucas Logic will be one of them, but there won't be anybody to buy them this time.

How to access them there Bulletin Boards

by R. D. E. Brown

(Or a better solution to Dr. Dark's Ring of Rust.)

1) What You Need

i A computer with an RS232 (V24) interface.
(Nascom 2 Owners NB: this generally means including the control signals but do not despair, read on.)
ii A Modem or acoustic coupler.
iii A terminal program (with file up/down load facilities).
iv Various bits of wire, plugs, your infinite patience, etc.
v A telephone line. (Preferably someone else's to keep your own bill down.)

2) Where You Get The Above From

i&v Most readers will already have one.
ii Surplus stores; I obtained an ex-GPO Modem 2b from Display Electronics for £30. The modem only does CCITT V21 300 bd but it is a well-built device in a very smart box (about the size of a Kenilworth case), complete with its own PSU.
iii CP/M users have it easy here:
Henry's have both the original MODEM 7 program and an improved UK version called UKM715, so pick a time when they aren't very busy, wander in with a disk, get down on your knees and grovel, and you might get a copy.
Tape Nascomers, PolyDossers, etc:
Not having tried this myself (you understand), start with a simple re-write of one of the terminal programs already published, then progress to a terminal file saving program by dumping all characters typed into memory before saving. (Similar to the Prestel program.)
iv The wire and plugs bit should be easy, as for the infinite patience, well, er ...

3) How You Put It All Together

Assuming you haven't fallen asleep yet, here's how you get the thing working.

Connecting the modem to the computer (serial interfaces, here we go again, yawn ...)

When the RS232 standard was conceived (?) it was assumed that communication would proceed thus.

Computer --> Modem --> (Telephone) --> Modem --> Computer

Computers are always thought of as
Data Terminal Equipment (DTE) or just plain Terminals

Modems are always thought of as
Data Communications Equipment (DCE) or Hosts

as they protect BT from its users by hosting the communication.

All computers, therefore, should be wired up as terminals and have a MALE plug. All modems are hosts and have a FEMALE socket. Thus, for Gemini serial connector wire a male DB25 plug onto it with transmit data out on pin 2 and receive data in on pin 3, bring all other control signals out to the plug as

specified in the Gemini CPU card manual. Connecting the computer to the modem is then dead simple; all you then need is a cable with a male plug at one end and a female socket at the other with each pin connected to its corresponding number, ie, 2 - 2; 3 - 3; 4 - 4; 5 - 5; 6 - 6; 7 - 7; 8 - 8; 20 - 20.

Now comes the dodgy bit:

4) Connecting To BT

Connecting to BT is something of a grey area. Presumably if you buy a brand spanking new modem, ie, spend more that £100 and it is BT approved with a BT jack plug, you can plug it straight in and off you go. Surplus modems, even ex-Telecom, are, however, a different ball game since:

i When becoming surplus they probably lose immediate BT approval.

ii They generally do not have a correct jack plug on them when you buy them. As far as I understand the current regulations, you are supposed to call BT up to put a correct plug on your modem to work with your existing phone.

Technically, if you do not comply with the above you can be prosecuted by BT for using illegal equipment especially if when you wire it up, you make a mistake and damage BT equipment. However, I am sure many readers have at least wired their own extension and adding a modem is similar.

iii The 2b modem has a two-wire connection to BT, this is to the red and white wires in current BT coding ie, pins 2 & 5 of the new wiper style socket. If you have a Senator style multi-extension system this also works.

Wire your phone in parallel to the modem, the best way to do this would be to wire an extension box with 2 sockets one for the phone, the other for the modem. If you feel at all uneasy about this either get a BT engineer who you know to do it, or get BT to do it officially.

iv For Nascom 2 owners with no RS232 control signals try and find out how many control signals the modem needs, some may only require DTR which could easily be faked.

5) More On UKM715

UKM715 as supplied by Henry's is about 64K of source well commented with a single patch area at the start. There are source copies either for Macro-80 or MAC, the Macro-80 version contains a Nascom 2 set of port numbers so somebody has got it working. There is also an installation program UKINSTL for those of you who get a .COM file which saves you assemblling it. For those interested in telexing, a Telex utility is available called UKTLX.

6) Testing Your Handiwork

Having built all this marvellous equipment, what will it do? Well, every month PCW prints a list of public access networks with tone standards and operating times. The problem with quite a few of these is that they operate only in the evenings when phone calls are a bit cheaper, most of them only

have 1 line so when ringing around, you will find they are virtually always engaged. So when testing, try the 24 hr ones during expensive call times and you should get through.

When testing, set your software into terminal mode and do not plug in your modem till you hear the high pitch modem answer tone from the other end, then connect your modem and type a couple of spaces or carriage returns to wake the other end up. Then follow the instructions as they appear if you get any problems, then putting your phone down and unplugging your modem will log you off. Good luck!

7) Where Is All This Leading To?

Having thought long and hard about this it appears to me that 80-BUS subscribers could communicate far more easily and more frequently through a bulletin board. Public domain software (80-BUS oriented), letters and information could easily be moved about without all the hassle of posting disks, tapes, etc. The 80-BUS Board should, however, be solely for 80-BUS News subscribers with a name (subscriber no) and password protection scheme. Such a Bulletin board could replace much of 80-BUS News, although it should not entirely kill off the newsletter. It may even provide more articles, who knows!

8) The $64K Question

Who is going to (ought to) set it up???

i Not me (cor, what a cop out!)

ii Gemini themselves??! After all, 80-BUS is their product and they are committed to supporting ex-Nascom hardware and they could afford to donate (hint, hint) much of the necessary hardware. They could also publish any product news and fixes on the Bulletin board.

iii A leading MicroValue Dealer?? or all of them! A MicroValue Dealer could also provide many bits of product information as well as even enhancing his own sales by allowing 24 hour credit card orders, assuming that the necessary security is available.

iv 80-BUS News! Obviously, one would have thought that this is the best solution, ie, the Board existing and maintained by 80-BUS News, however such a system needs a fairly large outlay on hardware which 80-BUS subscriptions could not cover. Thus, the best solution would be for 80-BUS News to run the Board in conjuction with hardware and technical assistance of either Gemini or the MicroValue group or both, in order to create a mutually beneficial service for Manufacturer, Dealers and most important of all, Users.

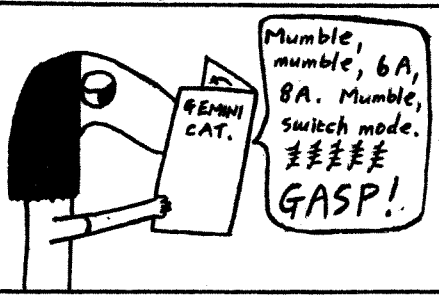
I hope that the above notes are thought-provoking rather than provocative.

Laurence (super-programmer, unequalled hardware hack, part-time president of the galaxy and all-round big head) has just added a Nascom I/O board (g'teed genuine) to his system, which is run off a 3A PSU.

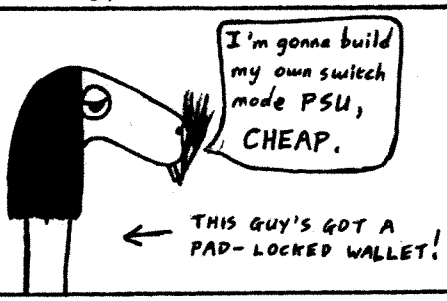
And....

Laurence has hit the 3Amp barrier, and his N-1 now has megalomania - it needs more power!

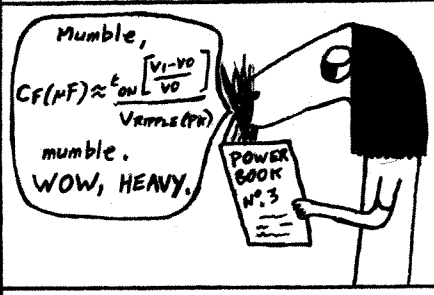
Laurence dives into the nearest Gemini catalogue.



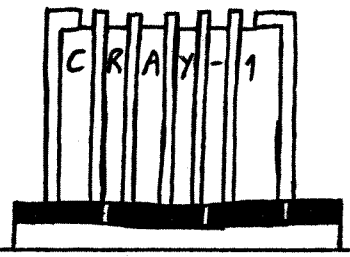
Convinced of a Nascom/Gemini price conspiracy, Laurence has an idea.



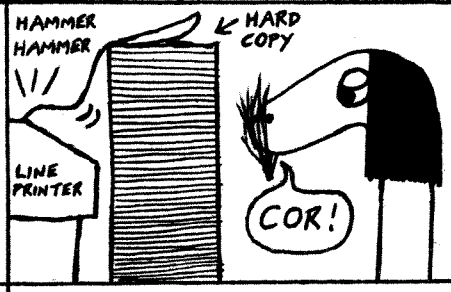
Out come the reference books, and the designing begins.



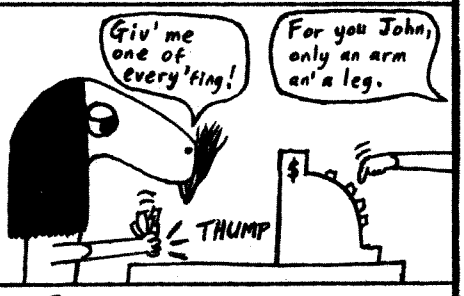
Laurence warms up his 'calculator' to help with the maths.



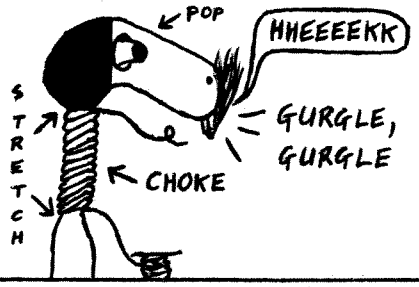
With the printout in-hand, the construction begins....



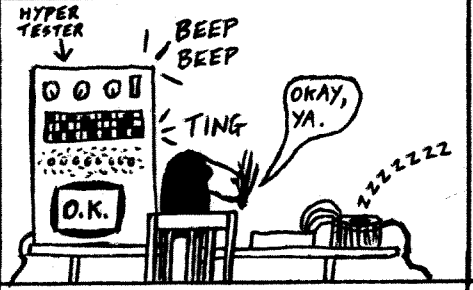
.... But first, the parts have to be bought.



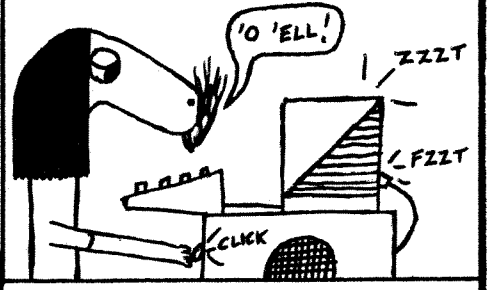
Winding high current chokes is easy, once you've got the knack.



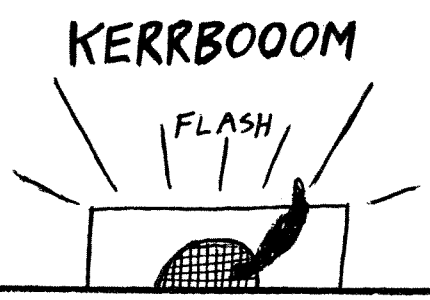
With his brainchild transformed into physical hardware, Laurence tests it...

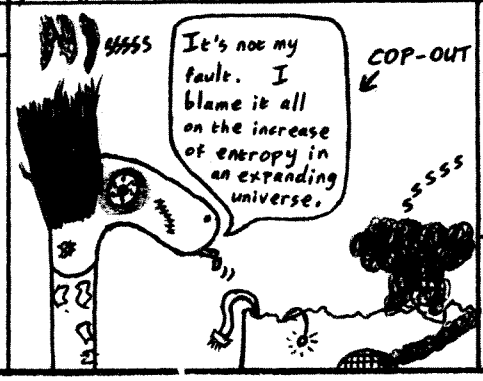


... Then installs it in his long suffering Nascom. But....



.... Sod's law strikes again.





Although this story did end in success in real life, a gratuitously violent ending sells more magazines.

[Ed.- That's not true, and if you say any diff'rent I'll smash 'yer face in.]

By D.G. Richards.
TONYREFAIL. MID. GLAM. 1983