

INMC NEWS

80

Issue: 5

OCTOBER - DECEMBER 1981

c/o Oakfield Corner, Sycamore Road, Amersham, Bucks. HP6 6SU

'Same Size' Issue !

CONTENTS

Page 1	This is it.
Page 2	Chairman's Bit.
Page 7	Letters to the Editor.
Page 18	Directory Listing Program for CP/M Systems.
Page 19	GENERALLY HARDWARE
	REVIEW of ROM Graphics for Nascom 1.
Page 20	DIY ROM Graphics for N1.
Page 22	REVIEW of Programmable Sound Generator.
Page 25	DIY Sound Board for Nasbus.
Page 30	About the Nascom 2 Keyboard.
Page 31	Adding 2716s to the RAM 'A' Board.
Page 32	Adding 2716s to the Nascom 2.
Page 34	REVIEW of Double Density Disk System.
Page 36	Subscribe to INMC80 here.
Page 37	About Disk Systems & About CP/M.
Page 40	REVIEW of Speech Synthesis Board.
Page 41	Miscellaneous Notes from the Editor.
	REVIEW of EPROM Programmer.
Page 42	GENERALLY SOFTWARE
	Teach Yourself Z80 - Part 5.
Page 50	REVIEW of Pascal.
Page 53	A Description of 'SYS'.
Page 55	PRINT Routine for Teleprinters.
Page 58	REVIEW of Software Book.
Page 59	Software for the Intelligent Video Card.
Page 62	Loading Several MC Programs at Once.
Page 63	Giving Your Programs Some Style.
Page 66	GENERALLY GENERAL
	Some Miscellaneous Notes.
Page 71	Lawrence and the 'New' Printer.
Pages 52/54/67	Private Ads.
Pages 68/69/70	Not So Private Ads.

£1.50

PLEASE NOTE. INMC80's Amersham address is used by INMC80 purely as a postbox. It is not possible for personal or telephone callers to obtain any INMC80 services.

PLEASE ALSO NOTE. INMC80 is run by a voluntary committee (of two!) on a part-time free-of-charge basis, and it is NOT possible for us to become involved in technical correspondence. Please contact Nascom or your Nascom distributor for this.

The DH Show

The Chairman's Bit

Back again, and may I apologise in advance for the length that this piece will run to. There are a number of things to report, so sit back and please try not to go to sleep through the boring bits.

When we added, somewhat facetiously, the words 'And don't expect another for a considerable time', below the 'Biggest Issue Yet' caption on the front page of the last issue we thought very little about it. Since then events have taught us to be more careful about such utterances. I have had a number of phone calls and people remarking, that we didn't really mean it, did we? And shades of gloom at the prospect of having to wait until Christmas NEXT year for the next issue seemed to abound. Well, sorry, all we meant was that we are variously off on holiday and things, and we knew we wouldn't get down to preparing the next issue before September. It's nice to know we are that appreciated, and very flattering too.

When I wrote the last Chairman's Bit I had just come back from the inaugural dealer meeting of Nascom under it's new masters, Lucas Logic. Now three months have passed, and wearing my dealer's hat for a moment, I can report that Nascom is slowly recovering ground lost over the past year. Low volume products that haven't seen the light of day in twelve months (buffer boards for instance) have re-appeared on dealers shelves along with I/O boards and the like. Nascoms (both 1 and 2) are readily available and the range of Nascom products available now is very much the same as when Nascom went into receivership. A noble effort on the part of Lucas to re-introduce so much so quickly. Unfortunately, things haven't picked up as quickly as was hoped, but I think the general economic climate may be blamed for that rather than any lack of effort on the part of Lucas or the dealers.

Anyway, Lucas have had another dealer meeting, to tell us what they are up to, and what is coming soon. ("Meeting last time, meeting this time, and I bet he doesn't only sell Nascoms either.", I hear you think. This makes my life sound like one continuous round of being wined and dined by various computer companies, sadly it's not true, and to cap it all, these meetings always seem to be on what would otherwise be my day off. People wishing to treat me to dinner are always welcome, but remember, Thursdays are out. Computer companies please note.) Firstly the new products due out about the time you see this issue. There's a case at long last, and it's not 'Son of System 80' either. It's designed to take the Nascom with two other boards and the Nascom power supply. It's tidy, coloured a sort of mushroom colour with a slightly textured finish, and looks a cross between an Apple and a Sorcerer case. Its made out of high density polyurethane foam (which sounds like the brittle stuff they make plastic ceiling tiles out of, but believe me, it's tough and resilient). From what I remember of this type of plastic, it has one snag. For the first week or two after moulding it has the most God awful pong as the resin catalyst evaporates. Something like overpowering cat's p**. This shouldn't worry you the end user, as the cases are bound to be a few weeks old by the time they see the shops. But I pity the guy who has to move them around in Lucas's stores. The case is provided with a metal backplate which mounts the power supply heatsink externally to reduce the dissipation within the case. The backplate also has enough prepunched holes to keep most people happy. The case costs about 36.00 and the internal frame kit to mount three cards costs about 23.00.

Another long awaited addition is the disk system. This consists of three components. First, the disk controller card which is a Nascom 8" x 8" bus card and uses the Western Digital WD1793. The card is a redesign of the original Nascom card, this time with the write precompensation circuitry implemented. Then there is the drive box, in a case matching the computer case, and designed with the drives side by side with an integral power supply to mount on top of the Nascom case. Lastly, there will be two operating systems, CP/M 2.2 for business or development applications and NAS-DOS (which if I'm not mistaken bares a remarkable resemblance to the B & L M DCS-DOS) for the home user wishing to maintain full compatibility with his existing software. The disk system is a double density system with a nominal capacity of about 350K per drive. The drives used (not named at the Nascom meeting) would appear to be TEAC TD50-C which are single sided 77 track drives and are mechanically incompatible with the existing Henelec/Gemini 805 or the Gemini G809/G815 disk systems, both of which have already gained wide acceptance with Nascom owners. It would be possible for users of the Nascom disk system to read one side of the disks from the competition but the necessary software to do this would be bordering on the horrific and I doubt that anyone will try. Another thing about the disk system is that the choice of drive and hence disk format limits the amount of 'off the shelf' software available to the CP/M user at the moment, as this particular format is at present little used. Nascom did say at the meeting that they will be offering a copying service to translate software from other systems to the Nascom format. I hope they have thought out the implications of that, when you consider the number of alternative disk formats already available, not to mention the infringement of copyright and license agreements signed by the original purchasers regarding the copying of the software by third parties. The disk controller card and case with one drive is about 470.00, with two drives 685.00, the CP/M software about 100.00 and NAS-DOS about 45.00.

Other items were announced for delivery late this year and early next. There is a Pascal promised (but no price and specification) and there is an extended Basic package intended as a 'bolt on' goodie to the existing Basic. This I understand will contain many of the features of the 'BBC Basic' for the forthcoming TV series. Mention was also made of domestic and business applications software, but no details. Most tantalising is the announcement of Nas-Sys 4 for some future date. No details, but it probably will contain driving routines for the colour card (see below). As this monitor will be inevitably larger than Nas-Sys 1 or 3, it prompts the question as to where they are going to put the rest of it to maintain compatibility, as the end of the existing Nas-Sysii are immediately followed by the video RAM and then the monitor workspace, neither of which can be touched to maintain compatibility. Perhaps it will be split, with jumps to the second half at, say, B000H. For Spring delivery there will be a high resolution colour video card, with facilities for some hairy graphics based on the Lucas Computer Aided Design package I mentioned last time. Price about 160.00 depending upon options. Cheap until you consider that a high resolution colour monitor will almost certainly be required to get the full benefit from the card, and they range from 300.00 to well over a 2,000.00. A colour TV could possibly (??) be used if you added a modulator to the AVC, but there is usually only one of those per household, and the wife watches that whilst you sit computing.

Here endeth the Nascom commercial.

Of almost equal importance was the announcement of the 'Nascom Fayre'. There was discussion as whether to spell 'Fair' 'Fayre' or not, I don't remember what was decided. Anyway, Nascom have decided to hold their own exhibition at the Bingley Hall, Birmingham, at the beginning of February. An ambitious project but looks most exciting. The idea is for the dealers to bring along all their various widgets to display, Nascom will be there in

force to display all the existing and new goodies, and (hopefully) deal with the myriad technical problems that we feel sure people will bring along. There was also talk of a seminar, although some projections of numbers will have to be gone into before this aspect is proceeded with. I guess Nascom adverts will start to carry invitations or something. Be that as it may, as Nascom aren't 'talking telephone numbers' for the cost of hiring a stand, we've already decided that we are likely to dip into the INMC80 funds and have a stand for ourselves. So with luck you can come along and meet us, and if there's a bar, treat us to a noggin or two.

Have you noticed we have competition for the affections of Nascom owners? Program Power have started a 'Nascom News' type venture. The press release in Computer Weekly rather implied that this was sponsored by Nascom, and I have heard a number of comments to that effect. Well it ain't. Private enterprise strikes again. Having read the first issue, and noting we have been credited where due, I can say that it is very interesting. The current issue runs to 32 pages and has more 'hardware' content than we normally publish. It'll be interesting to see how well it does, although it would be churlish to wish them anything but the best of luck.

One more quick commercial before getting on to the bus debate. Tangerine have just announced a modification to their Tantel Prestel adaptor. So apart from being able to talk to main frame Prestel computers via the phone, the Tantel can also talk to your Nascom via its tape I/O. There are some minor mods required on the Nascom, but these are simple. This means your Nascom can receive data from Prestel, or your Nascom can generate colour graphics for display via the Tantel adaptor. Could be useful. After giving Tangerine's 6 amp power supply a plug in the last issue, and wondering if they'd got the price wrong, it has shot up in price by about 25.00 (I wonder if they read INMC80 News!!). Still at 80.00 including VAT it's still difficult to beat.

Now on to other matters, the 'Bus Debate'. May I thank you for the response, I can't say that we've been suddenly inundated with letters, a steady flow is nearer the mark. There haven't been a vast number, but about 3% of the membership have replied, and from the spectrum of views, I think these can be considered representative. If you haven't yet given us your views and wish to, please write, we can only try to do as the membership wants. One thing I did ask was to just indicate your views on the matter in hand. This was a mistake on my part, and most of those who answered must have seen it for what it was, as most of the replies have been very detailed, closely argued, on average two pages each and on an interesting variety of different kinds of paper. It has been quite fun reading them and they confirm my suspicion that Nascom owners are all highly literate and erudite people and definitely a 'cut above the rest' of the home computing fraternity. My thanks, for apart from the Nascom only - Bus debate, we now have a far better idea of what people expect from this newsletter than if the replies had only been mono-syllabic. To the replies themselves, these cover a broad spectrum of varying shades, but by far the most boil down to 'leave the format, balance and content exactly as it is, but please remember we all have Nascoms at the heart of the system, so keep it relevant to using the Nascom'. Of course, both extremes were represented, from the signed post card with 'NASCOM ONLY' written on it in 1" letters, through a similar card that said 'NASBUS RULES, OK', to the Gemini compliments slip received with 'Gemini only' written on. The latter we disregarded as possibly having a vested interest!! We will be printing a representative selection of the letters in the letters column, but there were many detail points raised on a variety of issues, some of which I intend to take up now.

One letter we received said, "Why don't you do a review of the Watkins Tool Kit, I think it's marvellous.". That letter answered itself; Sir, if you took the trouble to write to us, and already have the item in question, and think it's marvellous, why didn't you write a review and put it in the envelope with your letter!! We can only review items sent to us for that purpose, or items we have ourselves. Unless, that is, a member who has a particular item cares to write a review. So please write, we can soon knock what you say into shape if you have doubts as to the publishable quality. On the score of knocking stuff into shape, I saw an interesting article submitted this month which included an appalling hand typed unreadable listing. Now we have neither the time nor inclination to rewrite source listings, so, whereas we can soon sort out text, sorting source code and then transcribing by hand is next to impossible. If it's a listing, please send it in machine readable form on tape (or disk), that way any mistakes are yours, and not compounded by ours. Another letter asks about preferred sizes of drawings for inclusion and also how text should be prepared. Well drawings (unless we redraw them) should be on A4 paper (we don't go in for photo reduction unless we have to), text may be type written, hand written (we'll transcribe it) or, we just love Naspen tapes. The writer also suggested publishing a small standardized word processor so people could submit tapes, we'll think about that one.

Another letter says that as we are always short of funds (true), why the recent gaily coloured covers and does printing newsletters less frequently save that much money? Well the gaily coloured covers add something like 10.00 to the total print bill, and we think this makes the mag more saleable in retail shops. As to the other point, the honest answer is yes! Although (on my part at least) this has been unconscious. Posting four newsletters a year is only two thirds the price of posting six. At 19p a time, you work it out.

A couple of letters raise the point of the high cost of adding RAM and other peripherals to the system. One letter went to the bother of costing a RAM card. In general I agree that costs are high. But taking an overall view, Nascom is cheaper than most (take the cost of adding 16K of extra RAM to a Video Genie for instance or the typical cost of S-100 cards). There are a number of considerations to be borne in mind. The costing mentioned above arrived at 90.00 for a 16K RAM, but totally ignored the amortization of the development costs (designing, prototyping, PCB layout, sample boards, documentation generation, etc.) for instance. Add that in, and the current price of 100.00 is not too bad.

A couple of letters mention the waving of the INMC80 banner, and lack of INMC80 publicity. True. Any volunteers for the job of publicity officer.

Rory O'Farrell (one of our more regular contributors) asks why we don't do as the Pascal User's Group, and publish all letters related to editorials, articles etc, with names and addresses so that members can read (and answer) all points and problems raised amongst themselves. Nice one Rory, only two snags. First if we published all relevant letters we would end up with a two hundred page newsletter, and secondly, we are very cautious about publishing address lists of members because anyone can then get hold of it, knowing that here is a list of a couple of thousand dedicated computer nuts. I get enough junk mail as it is, and just in case one of the senders of junk mail is reading this; ALL junk mail sent to my home address goes straight in the bin unread.

We've had the offer of free accomodation in a hotel in Nottingham for an INMC80 convention, last year we had a similar offer from a hotel in Wales. Thank you kindly gentlemen. But I think you underestimate the response (particularly if it's free), and this would place an extremely unfair demand

upon your resources. If the Nascom Fayre is a success, then an INMC convention could become a possibility. But please keep it on a commercial basis (non-profitmaking if you like). We don't like to feel we are sponging on the generosity of a couple of members.

A number of letters ask for details of Nascom clubs, sorry folks, we don't know who they are, as they seem as reticent about publicity as we are accused of being. Let us know who and where you are and we'll try and collate them into a list.

I've obviously read all the letters relating to the bus debate, and many members thank me personally 'for all my hard work'. Well this doesn't send me off on an ego-trip, as I know full well that there is one other whose name rarely appears in print and who does not often write bits for the mag, yet, whose contribution is equal to many times my effort. That is Paul Greenhalgh our Editor/Secretary. All the initial sorting of mail, editing, distribution, filtering of rubbish, subscriptions, paste up, getting it printed, you name it, have all been his doing. All I do is spend a few evenings a month at the computer typing, and taking the occasional decision. Paul does the rest. Whereas, I devote, say 20 - 30 hours a month, Paul's task is often daily, and not surprisingly, he's getting a little frayed around the edges. Reluctantly, he feels he's done his share, and has tendered his resignation. I can not say that Paul will be impossible to replace. For this mag to continue, he has to be replaced. But I suspect that no one person will be able cover the workload that Paul has so readily accepted for nearly three years. I have already approached a few people to see who is willing, and I suspect the committee will end up with a triumvirate of secretaries instead of one. This will be markedly less efficient, but I can see we have little choice. If you have a lot of time to spare (for no money) and are dedicated to computing, live north Londonish, are efficient, and can be left alone to work without supervision (and get it right), then please apply. In the mean time, thank you Paul, on behalf of us all, I don't know what I'm going to do without you.

Well that's the lot for this time. Any more comments on the bus debate are welcome, although the course is becoming clear. By the time you receive this the Committee will have been shuffled, and the next issue will be in the hands of those new members who will have been co-opted to replace Paul. Again, our thanks to Paul for all his hard work.

Sincerely
D. R. Hunt.

Chairman's bit post script.

As you may have gathered, the above was written sometime towards the end of September (it's so long ago I forget exactly when), with every intention of creating an October newsletter. Since then other things have intervened and we haven't had time to put it to bed. The delay over the last fortnight has been my fault, finding time to type up the debate letters. Well I've sat down tonight to finish it off, and with luck it'll be off to the printers just the right side of December. My apologies. Whilst I'm at it, I wish you all a very happy Christmas.

With apologies,

D. R. Hunt.

LETTERS

RESTORING BASIC

Have you ever typed CLOAD when you really meant CSAVE? Come on now admit it. Well all is not lost. The only thing that has happened is that two pointers have been changed and they are easy to change back. This program restores the pointers and warm starts basic for you.

The easiest way to use it is to have a generated tape. All that is needed then is to reset and play the tape. The program can go anywhere in memory, so long as it does not overwrite the basic, and it will also recover from NEW and a basic cold start

Program Retriever

=====

```
21 FE 10 7E FE 00 23 20 FA 22 FA 10 5E 23
56 EB 7E 23 B6 2B 20 F6 22 D6 10 C3 FD FF
```

B.M. Farrelly, Belfast

SPACE INVADERS

I have just received INMC80 News and would like to congratulate you on its quality.

With reference to W.Squires letter in the INMC80-3 edition I have enclosed a relocatable routine which will allow you to select the keys used to control the base movement. Once the routine has been typed in the Space Invaders program should be executed from the start of the routine.

```
0F70 00 31 FE 0F EF 0C 00 D7 - 3A EF 6C 65 66 74 20 00
0F80 CF F7 F5 D7 4A 22 86 17 - 32 8C 17 F1 32 5C 10 32
0F90 40 10 D7 1F EF 72 69 67 - 68 74 20 00 CF F7 F5 D7
0FA0 2E 22 91 17 32 94 17 F1 - 32 58 10 32 42 10 DF 5D
0FB0 C3 00 10 EF 0D 0D 45 6E - 74 65 72 20 6B 65 79 20
0FC0 74 6F 20 6D 6F 76 65 20 - 62 61 73 65 20 00 C9 21
0FD0 09 0C 06 08 7E B7 C0 2B - 10 FA 00 00 00 00 00 00
(NAS-SYS only)
```

To slow down the speed of the game he could insert the following simple delay routine at 1776H, to change the delay length he should alter the value at 1777.

```
1776 06 30 FF 10 FD 00 00 .....
1780 00 00.
```

This replaces the routine which checks which monitor is being used, I have assumed that he is using NAS-SYS. These may not be the best solutions to the problems, but they do work.

Finally, I have the Xtal 2.2 BASIC and would like to add the SET, RESET, and TEST commands to it, could you please publish an article on how this could be done. (Any volunteers - Ed.) Also is it possible to add extra commands to NASPEN? I have in mind commands such as 'find a word and replace it with another' or 'delete word'.

Mark Taylor of Hackney, London

Nascom Microcomputers
Lucas Logic Limited
Welton Road
Wedgnock Industrial Estate
Warwick
CV34 5PZ

Tel: 0926 497733
5 October 1981

INMC 80 News
c/o Oakfield Corner: Sycamore Road
Amersham
Bucks
HP6 6S4

Dear Sirs

I am writing to you in response to the letter from Mr Scadden published in Issue 4 of INMC 80 News. We knew that Nascom owners are enthusiasts, but building three in a week is certainly rather unusual. Keep up the good work!

Regarding the points made by Mr Scadden we would like to make our position clear as the new owners of Nascom.

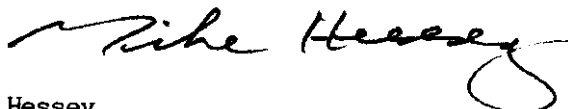
1. All Nascom manuals used to be produced by photocopying, and as a result the occurrence of angled pages was not unknown. Since we have taken over we have been modifying and restructuring parts of the manual, and have in the process converted to a printing process for the entire manual. This will eliminate the problem described, but in the meantime anyone with faulty pages supplied by ourselves will be sent new copies of the page concerned on request.

2. Any components in Nascom kits which were faulty on delivery will be replaced free of charge by the dealer from whom the kit was purchased. Faulty boards have been known to occur occasionally in the past, but we hope that we have now managed to eliminate this problem. Any such faults should be taken up with the dealer from whom the equipment was purchased, who can contact us on the subject if necessary.

As both I and the Editor have indicated problems should be dealt with in the first instance by the dealer who supplied the equipment, since it is much easier to resolve such matters in person rather than via the telephone. However, if you feel you are not receiving fair treatment you can of course contact us subsequently.

Finally I can assure Mr Scadden and other readers that there are some interesting new products in the pipeline - depending on the date of issue of the next copy of INMC 80 News they may already be available. Like Mr Scadden I look forward to receiving my INMC 80 newsletters.

Yours faithfully



Mike Hessey
Technical Manager

TRANSPARENT VIDEO

I read Mr Willmott's letter in INMC80 issue 3 (page 42) with interest and I thought you might be interested in my own ideas on the subject.

The reason Mr Willmott's attempt to use WAIT failed is that he has misunderstood the effect of WAIT. It is described for various cycles (Memory read, Memory write, etc) on pages 15-18 of the Z80 Technical Manual. Whenever /WAIT is asserted the cycle is stretched by inserting wait states until /WAIT goes high again. During this time all the bus signals are maintained. Thus while waiting for a blanking period the VRAM signal will be active, preventing the VDU circuitry from accessing the RAM.

For this technique to work the VRAM select line must be interrupted sufficiently early in its path to prevent the VDU circuitry from being locked out. However this could itself raise problems. The VRAM select would probably have to be restored before releasing the wait, to give the RAM time to pick up the CPU access. Also if dynamic memory is in use, long periods spent waiting for RAM access could prevent refresh from cycling fast enough.

I have been considering a different, though related, solution to the same problem. The blanking signal can be connected to an input port bit allowing the CPU to monitor its state (the PIO can be used for this, so that no extra hardware is required). Then, before accessing the VDU memory, this input is tested, and the CPU waits in a short loop until the next blanking period starts. I have not so far experimented with this enough to be sure it works, but the technique is promising.

/BLANKING is a composite of two blanking signals: blanking between scan lines, and blanking between frames. The former lasts for quite a short period - I think less than the time required to access one byte. Therefore I have been using only the interframe blanking period, identified by the VBLANK signal which is brought out to TP22. By my calculations the blanking period lasts around 5.38 msecs., which would allow quite major alterations to the VDU RAM before the next frame starts. In addition after VBLANK ends there is still a short blanking period at the start of the first displayed line, which allows for some overrun.

One way of handling moving graphics displays would be to keep a copy of the display in non-displayed RAM. This copy can then be updated as required, and copied to the VDU RAM during one or two blanking periods. (At 4MHz without wait states, LDIR could just about move 1K bytes in a single blanking period.)

I hope this helps Mr. Willmott and others. I would be interested in hearing about other members' efforts in this area.

Stephen Prince, Winchester

PASCAL CHANGES

Further to my review of the Hisoft Pascal, I have now been informed by Hisoft that all copies of the NAS-PAS 3 supplied after 1st May 1981 include the following additions to the predefined functions:

EXP(x), LN(x), COS(x), SIN(x), ARCTAN(x), TAN(x)

where x is a REAL or INTEGER argument (in radians for trig functions), returning a REAL result.

I omitted to mention that the benchmark timings as suggested by PCW show a very creditable performance from the code produced by this compiler. The manual comes with suggestions on how to improve the runtime speed even more, but quite frankly I don't think this will be necessary.

Nas Pas 3 purchased before 1.5.81 will be updated with trig. routines for 3.50.

Rory O'Farrell, Co. Wicklow

ZEAP AND PRINTERS

I was encouraged to read Len Ford's contribution (INMC80 Issue 3) since it suggested I was not after all the only NASCOMer still relying on the T4 monitor. Financial constraints also guided me towards the Creed 7B teleprinter to obtain that essential hardcopy output. I am writing in case there are other users who had not yet succeeded in directing assembler listings from ZEAP to the printer.

The version of ZEAP I use is an early one, 1.1 in fact, on tape, which loads at 0F00. My printer driver software is a modified version of CREOUT published in PCW April 79. The printer is connected via a spare bit in port 0 on the Nascom 1. CREOUT takes an ASCII character in register A, translates it to Baudot code and outputs the appropriate bit pattern to Port 0.

ZEAP 1.1 changes the CRT reflection at 0C4B from 013B in the T4 monitor to 18E1. This piece of coding calls the monitor CRT to print the contents of A after first checking A >= 1D.

Not all writing to the screen is done through calls to this reflection, however, e.g. 1803 CALL Z,013B. And not all information is written to the screen through 013B e.g. the line numbers, which are stored in hex are printed using a CALL to monitor 0232 in location 1458. My requirement was to direct assembled listings from ZEAP to the CREED teleprinter. To do this I use a print routine which takes complete lines from the bottom of the screen rather than individual characters. This takes advantage of the fact that all instructions to scroll up the display are handled through 18E5 CALL 013B. We can examine A at this point and if it is 1F (CRLF) the display is about to be scrolled up. In that case the bottom line on the screen is copied to the printer before the scroll is performed.

I enclose the section of my printer driver program which deals with this procedure.

Having loaded ZEAP replace the contents of 18E6 and 18E7 by the address of the new routine ZPRINT. ZPRINT activates the printer by the jump to LPV at line 1400 which prints the character in A. ZPRINT also tests for and ignores trailing blanks and paginates the output.

40CB	C3500C	1400	NOECHO	JP	LPV
40CE	F5	1410	ZPRINT	PUSH	AF
40CF	FE1F	1420		CP	#1F
40D1	CAD940	1430		JP	Z PRNTLN
40D4	F1	1440	NPRINT	POP	AF
40D5	CD3B01	1450		CALL	CRT
40D8	C9	1460		RET	
40D9	C5	1470	PRNTLN	PUSH	BC
40DA	D5	1480		PUSH	DE
40DB	E5	1490		PUSH	HL
40DC	3A0040	1500		LD	A (LCOUNT)
40DF	3C	1510		INC	A
40E0	FE40	1520		CP	PAGLEN
40E2	200D	1530		JR	NZ SAMEPG-#
40E4	210A00	1540		LD	HL LINGAP
40E7	46	1550		LD	B (HL)
40E8	3E1F	1560	BLLINE	LD	A #1F
40EA	CDCB40	1570		CALL	NOECHO
40ED	10F9	1580		DJNZ	BLLINE-#
40EF	3E00	1590		LD	A 0
40F1	320040	1600	SAMEPG	LD	(LCOUNT) A
40F4	11B90B	1610		LD	DE #BB9
40F7	0630	1620		LD	B #30
40F9	1A	1630	TESTBL	LD	A (DE)
40FA	FE20	1640		CP	#20
40FC	2004	1650		JR	NZ ENDFND-#

40FE	05	1660	DEC	B
40FF	1B	1670	DEC	DE
4100	18F7	1680	JR	TESTBL-#
4102	118A0B	1690	ENDFND LD	DE #B8A
4105	1A	1700	COPY LD	A (DE)
4106	CDCB40	1710	CALL	NOECHO
4109	13	1720	INC	DE
410A	05	1730	DEC	B
410B	C20541	1740	JP	NZ COPY
410E	3E1F	1750	LD	A #1F
4110	GDCB40	1760	CALL	NOECHO
4113	E1	1770	POP	HL
4114	D1	1780	POP	DE
4115	C1	1790	POP	BC
4116	C3D440	1800	JP	NPRINT

Many thanks for an excellent mag., but where oh where have Lawrence and friends got to? Not defected I hope.

C.G. Richards, Wimborne, Dorset.

NASBUG TO NAS-SYS

I have bought several programs from the INMC80 library for my minimum Nascom I with Nas-Sys, but not all the Nasbug programs are available for Nas-Sys. So I decided to modify those that weren't available. The first was Life by J Haigh and this is how it goes:

1. Load the program from OC80 instead of OC50. It still fits easily into the minimum system.

2. Apply the following patches (shown as address, old contents, new contents)

OC81 1E OC	OC87 7E AF	OC99 CD CF	OC9A 3E 00
OC9B 00 00	OCD6 A3 D3	OCD9 A4 D4	OCE7 B2 E2
OD28 AB DB	OD36 53 83	OD53 53 83	OD82 9E CE
OD85 97 C7	OD8D A3 D3	ODB4 93 D1	ODC8 30 60
ODCE 1E 4E	ODD1 2B 5B	ODD4 6F 9F	ODD8 B2 E2
OE17 CD DF	OE18 69 61	OE19 00 00	OE1F CA CA
OE20 F6 26	OE21 0C OD	OE22 CD CF	OE23 3E 00
OE24 00 00	OE26 8C BC	OE2E C3 C3	OE2F F6 26
OE30 0C OD	OE36 50 80	OE3A CA CA	OE3B F6 26
OE3C 0C OD	OE40 53 83	OE43 7A AA	OE4B C3 C3
OE4C F6 26	OE4D 0C OD	OE4F 35 65	OE5C 3F 6F
OE61 49 79	OE64 0E DF		

3) Now save it! The result is well worth the effort - a very interesting program.

Two of the patches were applied because of (non-serious) errors in the listing.

- 1) Centre of screen (ff)
- 2) Population count (m)

Spurred on by my success, I started on Robots, but my cassette recorder failed me and work stopped. I hope to report success eventually.

W.H. Turner, New Malden

IT'S DIFFERENT

I am writing to point out an error in the Chairman's Bit in INMC80 issue 4. The third paragraph of page 3 mentions a new computer and suggests that the software for it will be virtually identical to the Nascom software. I assume that you are referring to the Gemini system, which is described on page 57 of that issue. Since I wrote NASBUG T4 and NAS-SYS 1 and 3, and have just completed development of the RP/M software which is the operating system for the Gemini machine, I should be in a good position to know that you have made an incorrect assumption.

In fact, RP/M bears no resemblance to the Nascom software since it is the result of a new and original design. Differences include the commands (all different), the tape data format (totally different) and the operating system call support (completely different)! In fact, RP/M pretends to programs that it is the CP/M disk operating system, and it can run software such as the Microsoft 24K Disk Basic. Incidentally, that takes 4 minutes to load from cassette, but you only need to do it once each session - anyway you need some reason to upgrade to disks eventually!

I must agree that it is possible that software written by independent suppliers, such as ZEAP, NASDIS, DEBUG etc. might be converted if they feel like it. This will depend on many factors, and does not alter the fact that the systems are very different.

Richard Beal, Kingston.

(We have heard that some software is being rewritten for RP/M, notably NASPEN, which in its 4K GEMPEN version is virtually identical to the existing DISKPEN, itself an enhancement of NASPEN rewritten to run under CP/M on the Gemini Henelec G805 disk system. NASPEN, GEMPEN and DISKPEN are available from the Microvalue ('Gang of Six') dealers. Ed.)

THE BUS DEBATE

=====

Here is a small selection of the letters we have received on the 'Bus Debate', in most cases these have been edited as most covered other points not relevant to the debate. We hope to catch up on the more general letters in the next issue, and perhaps publish more from the debate then.

Congleton
Cheshire

Dear Sirs,

Though I can understand why John Deane should want INMC80 to stick with Nascom alone, my support must go to the bus.

But in any case, keep going.

Yours faithfully
P. D. Taylor.

Finchley,
London N.3.

Dear Dave,

You asked for comments from readers on future policy towards Nascom products and 80-BUS products.

I wish Lucas Logic every success in getting it all going again and feel that the INMC80 should give priority of space to news and information on their products, particularly the technical updates which we hope will be forthcoming from their design team.

Nevertheless, when the future of Nascom seemed so uncertain and many of us might have ended up with some un-repairable and certainly un-expandable white elephants on our hands, the product was kept alive by the small group of dealers and manufacturers who gave service and assistance as well as developing compatible products. INMC80 must continue to support these as well. I don't think Mr. Deane has anything to fear from this, in fact I doubt if his company buys many components which are not "second sourced" to use the jargon; and as long as you don't start publishing information for users of American white-boxes and Japanese copies thereof, I see no need to rename the magazine "Practical Computing" or somesuch.

Yours sincerely,
Robin Luxford

The following is an extract from a long and detailed letter from David Bryden of Scarborough.

I am concerned that the club and the manufacturers might make the mistake, as I see it, of making and supporting only complete systems. By all means do that, but please don't forget that a lot of us started off as primarily constructors. Having said that, it would be of tremendous interest to me if the system could develop to a stage where I could have a homebrew system compatible with a commercial system at work.

I feel that in the current situation you have to support the bus rather than the manufacturers, because in the period of limbo so much time and effort has been put in by the dealers developing products. If we had had to wait for the outcome of receivership then I think many of us would have changed to other systems. Further, I feel that whatever Lucas say, if the subsidiary, Lucas Logic loses money, then we could be back in the familiar situation.

I see the club's role as that of a co-ordinator and projector of hobbyists such as myself, whose interests won't always coincide with those of the manufacturers, who sees himself selling large volumes of unsocketed boards and packaged software. Please don't let us get into the Pet/Tandy position.

Etc.

Beddau
Mid Glamorgan

Now that the Nascom ship has been salvaged and appears to have a promising captain at the helm, perhaps some of the deserters will return to the fold.

Subscription to INMC80 seems pretty good and I suspect that most subscribers own or plan to own a Nascom. On these grounds I think INMC80 should remain Nascom oriented since there is only one other Nascom oriented publication available. Naturally Nasbus is important to Nascom users, but how many will be interested in the purchase of another Nasbus computer? When the other computer is more established, it may be worth broadening the INMC80 coverage to cater for the common units which may be used with Nascom. Till then the magazine should be consolidating itself and not losing touch with owners of Nascom 1 and 2 machines.

Etc.

Yours faithfully,
Dave Lorde

Greenock
Renfrewshire

Dear Sir,

I Wish to advise you that I am in favour of our mag supporting 'Nascom only'. However, General articles on, say, the Z80 and how to interface IBM

printers, etc, should also be included. Any product approved by Nascom for attachment to their computers should also fall within the 'Nascom only' definition.

Yours faithfully,
David Edgar

Bognor Regis
West Sussex

Dear Editor,

I agree wholeheartedly with your suggestion that the 'news' become Nasbus oriented and not solely concerned with Nascom products. In this way one can get a properly balanced view of all that is available as a rival to the S100, however small.

(There follows some technical details related to Nasbus specs).

Thirdly, but not least, I note that there are a number of 8 x 8 cards which are becoming available for the Nasbus. The operative words in this are surely 'becoming available', not actually 'available now'. One hopes that we do not go through the usual rigmarole of products 'being developed' which somehow seems to take two years to fruition.

Etc.

Yours sincerely,
John Stuckey

(Ed. I am sure that all the boards that we referred to in the last issues are 'available now' - sorry if that is a shock to the system!)

Baldock,
Herts.

Dear INMC80,

My vote goes to Nasbus compatible products. Without them the present dedicated dealers would have hardly survived, so I don't go along with the Lucas request for Nascom only.

Many thanks for another superb mag etc.

Yours gratefully,
John Raines

University of Sheffield
Sheffield.

Dear Mr. Hunt,

I think the INMC80 should support all Nasbus compatible products, not just those from Nascom. Whilst the newsletter would clearly benefit from the blessing of Lucas Logic Ltd., its prime responsibility should be to its readership, who I think would be better served by taking a 'wider' approach. Enough said.

(He then says some very nice things about us, and please send us a rough draft of your 'Laboratory Control System', it sounds fascinating).

Yours sincerely,
Dr. C. R. Brown

Leicester.

Dear Chairman,

I feel that the mag should retain its existing identity and content, continuing to cover ALL Nasbus compatible items, since these are of prime interest to the membership and must be encouraged by Lucas since they enhance and encourage ownership and sales of Nascom products.

Now as regards alternative main boards and systems, I know that interest and hope during the past year, of any future, has depended a great deal on the INMC committee, but also on those who now wish to follow parallel commercial paths. The membership owe them some support. Would it be possible to open a 'Compatible System' section at the rear of the mag to allow these to be commented on and advertized. The size of this section could increase (or decrease) dependent on the future related membership, subscription fee, increased committee size to deal with the increased work load, etc. No doubt Lucas would object to competitive adverts, but competition is good for inspiration and could only benefit future customers, which, in part will become part of the future membership. --- Unless, that is, Lucas intend to offer us all discounts, or something for exclusive rights!! ---

Yours sincerely
Eric Downs.

Acomb,
York.

Dear Ed,

Having galvanized myself into action, I have a couple of comments to make.

Nasbus (80-BUS ? etc)	Yes
Nascom only	No

If the opinion need justification, then I feel that most Nascom owners want the most from their machines and hence need to know about products from as many sources as possible. Lucas may be wonderful (correction: ARE wonderful, they bought Nascom) and should obviously be given a chance, but then so should Winchester Technology, Gemini, Arfon, I/O Devices, etc.
Etc.

Yours sincerely,
Paul Wilson

Horndean,
Portsmouth.

Dear Sirs,

Briefly:

- 1) INMC80 is and should remain primarily a Nascom users magazine/club.
- 2) It follows that all peripheral products using Nasbus/80-BUS or the Nascom serial or parallel interfaces and all software compatible with Nascom, whoever makes or markets them should fall within the scope of INMC80.
- 3) Other computers following very closely on Nascom software and hardware conventions may be included if the arguments proposed are relevant to Nascom users. Other computers using the same bus but not the same hardware/software conventions should be excluded.

Yours faithfully,
A. J. Fry

Kirkby-in-Ashfield
Nottingham.

Dear Sir,

By all means support the Nasbus, but keep the INMC80 Nascom. I thoroughly agree with John N. S. Deane when he says that supporting rival products would mean becoming like all the other computing mags.

Finally, it seems strange that INMC80 is considering dropping Nascom (well almost) when Program Power are launching a new Nascom only magazine for a mere 95p a copy.

Staunch Nascom Fan,
G. C. Claypole

Dundee

Dear Editor INMC80,

THE BUS OF COURSE. Mr. J. N. S. Deane of Lucas Logic appeals for a chance. By all means Lucas should be given a fair chance but he should not be surprised that Nascom owners are more than a little uneasy about resting their eggs in one basket.

Most are capable of appreciating a good product, why else did we plump for a Nascom. The market is here with we the owners or future owners, and the customer is king. He can be sure we will continue to exercise care in selecting products. I wish him well, as like most I would like to see Nascom flourish. If however he is overwhelmed by smaller competition he can rest assured it is because he has not appreciated our needs, or that our money can buy better elsewhere.

Many thanks to you all in producing an excellent news letter, more power to your Naspen.

R. Bain

Harrow,
Middlesex.

Dear INMC80,

Yes, follow the bus unless it's essential to 'keep in' with Lucas Logic.

Yours sincerely,
John Waddell

Thornton Heath,
Surrey.

Dear Editor,

With regard to the direction the club should take in future, there has in the past been no question of restricting it purely to Nascom items and excluding other companies' products. The real question then, it do we choose to ignore one particular item from Gemini's compatible product range whilst acknowledging the rest?

If we do so, where do we draw the line? Perhaps by accepting only products that Nascom think fit to approve. The club would then regress to little more than a mouthpiece for Nascom. Pretending that the Gemini board does not exist would also inevitably lead to coy references to 'the other system', and confusion and duplication of effort by owners of the different systems.

On the other hand, much mutual benefit and experience could be gained from users of another intrinsically very similar machine. After all, Gemini users will be trying to do broadly the same things with the same add-ons, and (with disks at least. Ed.) with very similar software as Nascom users.

In my view therefore, we have to go for the 'Nasbus compatible' approach, and brush aside calls to ignore anything non-Nascom.

P. J. Mathews.

Loughborough
Leics.

Dear Sir,

I feel most strongly on this matter and would certainly discontinue my membership if the Nascom content were diluted. Most people are apathetic until 'it' affects them and I suspect protests would not occur until after any decision had been taken. Please give Lucas a chance, they obviously realise that Nascom 3 is needed and perhaps a reworked Nascom 2.

Nascom's success to date has been that you can personalize the computer, understand, alter and modify as the mood/requirement dictates. One can see how all this has contributed to the success of Nascom by the numbers of add-ons that have been produced. Not everybody needs an all dancing all singing micro.

Yours sincerely,
A. Braithwaite

The following letter has been received from John Marshall, the founder and ex-Managing Director of Nascom, and now MD of Gemini Microcomputers Ltd.

Dear Editor,

In the 18 months that have elapsed since May 27th 1980, when I was obliged to call in the Receiver to Nascom Microcomputers Ltd., I felt that it was inappropriate for me to make any observations concerning the history of Nascom and subsequent events. Now that the company has been sold to Lucas, and taking into account that a reasonable 'honeymoon' period has lapsed, I feel that the time is perhaps appropriate for me to bring into the open certain facts which appear to have been obscured by Nascom's period of difficulty.

First and foremost I would like your readers to understand that the original Nasbus concept, as perceived by myself and my colleagues, was that given the large volume of sales achieved with both Nascom 1 and 2, it was conceivable that Nasbus could be established as the standard Z80 BUS outside of the US. Unfortunately the Receiver and I did not share the same attitude towards this important issue and eventually we reached a situation whereby it was impossible to use the word 'Nasbus' without involving ourselves in lengthy and expensive litigation. I therefore proposed that the last (and not very clearly defined) issue of Nasbus should be carefully re-examined, tidied up and republished under the name of 80-BUS. This work was carried out earlier this year.

After the Receiver took over control of Nascom, it became obvious that the sales and marketing effort of the company was to dwindle to virtually zero until such time as the new owner arrived. In the dark months of the autumn and winter of 1980 both I and my fellow distributors concluded that the prospects were steadily becoming less positive for the future of Nascom. In the December of 1980 I took the decision to proceed with the design of a two card system which could, if necessary, although not at the same price, be used as a substitute for Nascom 2 if the Receiver was unsuccessful in his attempts for selling the company as a going concern. As luck would have it the launch of Multiboard and the rescue of Nascom occurred almost simultaneously and whilst initially I must admit I was concerned that this could diminish the sales of Multiboard I subsequently concluded that for the first time the prospective customer would be faced with two real alternatives as far as his prime supplier was concerned.

I have endeavoured to bring to the market products which have long been wanted but which for various reasons had previously proved almost impossible to produce. I do not believe that in any of my actions I can ever have been accused of undermining Nascom. The Gemini 805 disk system answered the extensive call from customers for a low cost disk system. The EPROM Board, EPROM Programmer, subsequent disk card design, the 64K RAM board, Supermum, Hypermum, 80 x 25 display on the IVC, all were responses to demands from the marketplace for Nascom compatible products.

We must now address ourselves to the future. I believe that 1982 is the year in which we can establish the 80-BUS as the de facto standard (please excuse the cliché) Z80 BUS. The Z80 is now the largest selling processor and is, I believe, the base around which a formidable range of product can be built. There are already approx. 20 80-BUS compatible boards available from 7 different manufacturers and the numbers are growing daily. Gemini and the other companies involved in the production of 80-BUS compatible boards will hopefully in 1982 at last manage to produce a system which can surpass the flexibility and range currently offered by S100 based products. In addition to this I believe we have two major advantages:

1. TOTAL COMPATIBILITY. Unlike S100, which seems to conform to a variety of standards, 80-BUS (as published in INMC 80-4) has been clearly defined.

2. COST ADVANTAGE. The cost of producing 80-BUS boards v. S100 boards gives the user a significant saving if adopting the "logical route" i.e. 80-BUS.

The Editor kindly gave me the opportunity to preview the responses to 'The Great BUS Debate' which are included in this issue. I have concluded that many feel that Gemini and Nascom are converging on a collision course, and whilst there are some areas of direct competition, the recent radical moves on our part as far as software is concerned should ensure that a conflict of this nature will not take place. Gemini have adopted a CP/M look-alike monitor called RP/M (produced by Richard Beal, the author of Nasbug T4 and Nas-Sys 1&3), which means that customers starting with a ROM based system can move on to CP/M without making any changes whatsoever to their software, simply transferring the programs from tape or EPROM to disk with the routines provided. Nascom, however, for historic reasons continue to primarily follow the path of Nas-Sys.

The customer therefore has two clear choices - a single board Nascom approach, or the Gemini MultiBoard approach. In order to allow the user to expand his system and exploit its potential to the fullest, the adoption of one BUS would mean that peripheral manufacturers would have two basic systems to supply with the same product, thereby reducing costs dramatically. Surely this must be to the benefit of all of Nascom and Gemini customers?

Yours sincerely,

John A. Marshall,

Managing Director, Gemini Microcomputers Ltd.

CAT

Directory listing for CP/M systems

Steve Hanselman

Here I sit, 11 o'clock at night typing away at the keyboard of my trusty Nascom, after a hard days slacking. Anyway, that's enough of my tales of woe, here's what I'm writing about.

It's a bit for CP/M disk systems. Those of you who have a disk system running CP/M 1.4, (and lets face it, judging by the number of people seen carrying away expensive boxes from the various dealers, there must be plenty around), will probably have noticed the fact that when a directory exceeds 13 entries it scrolls off the screen of the Nascom. Well here's a small program that can be entered under ZSID or DDT which will give a three column directory display. It is executed in the following fashion:

A>CAT	Which will list all the directory entries on drive A.
A>CAT d:	Which will list all the directory entries on drive d:
A>CAT filename.typ	Which will list all the directory entries with the unambiguous filename.typ on drive A.
A>CAT d:filename.typ	Which will list all the directory entries with the unambiguous filename.typ on drive d:.

Tab version 1.0

HEX output file

HEX dump of : CAT .COM

[illegible]

N1 Graphics

ROM Graphics unit for Nascom 1 (a.k.a. Econographics)

=====

a review by Rory O'Farrell.

This hard to find item was trapped for me by Bits & PCs, to whom much thanks. Other dealers were saying that the Econographics kit was unobtainable (even the INMC80 suggested this in the INMC80-2 newsletter). It costs 30.00 plus VAT, and consists of a P.C. board about 6" x 3.5" which is singlesided, and soldermasked. On this board are (or will be when you put it together) 3x24 pin DIL sockets, and sockets for 4 smaller i.c.s. You are also supplied with a short length of 24 way cable and two 24 pin DIL plugs. This umbilical is wired up to connect from one socket on the Econographics unit to the socket on the N1 which held the character generator.

The instructions for assembly are fairly clear, but suppose that you have already assembled an N1 so that you know about the handling of MOS character generators. You first solder the 37 (THIRTY SEVEN !!!) links on the board, the capacitors, and the sockets. Soldering the capacitors is where the trouble starts. The instructions say that you should line up the + mark on the two Tantalum caps with the + mark silkscreened on the board. There is no such mark! Then when tracing the tracks to find out the polarity, so you can insert the tants properly (Tants don't like reverse voltages - normal rating is less than or equal to 0.3 volt in reverse), while tracing the tracks to find the polarity, you notice that pin 1 of the input socket, which carries -3v is connected to pin 16 or 14 of the small i.c.s. Worried by this, you then check a bit further, and find that pin 2 of the input socket is connected to pin 3 of IC1, which expects + 12 volts. Now panicking, you go away and have a cup of coffee. On your return, you inspect the board. Definitely, the tracks are mixed up somewhat. You also notice that IC4 does not appear to have any connection to pin 16. However, the problems are easily solved.

Holding the board, looking at the track side, with the 24 pin sockets towards you, IC4 is at top left. Pin 16 is the top righthand pin. It needs to be connected to the broad horizontal track just about it. The track should have the solder mask scraped off it with an XACTO knife or similar. The input socket is the bottom right of the board, and pins 1,2,3 are the rightmost on the bottom row. With a vero cutter, or a hand held drill bit, or the knife, cut the connection to pad 1 and 2 from the direction of the top edge of the board. This will leave connections to those pads coming from the direction of the bottom edge of the board. These are correct. Using an insulated wire link, link pad 2 (the second one in) to the topmost of the two broad tracks beside where you are working. This is now correct. With a wire link, link pad 3 to the lower of the two broad tracks. Check that you have no solder bridges, and that you have in fact cut the two broad tracks connecting from the top edge direction to pads 1 and 2 .

You can now complete the board according to the instructions. Wiring up the 24 way umbilical is a bit tricky, but there is just enough wire for another go if you are careful.

With the board assembled, you now need to turn you N1 board over to make a few links on the back. Before doing this, it is advisable to unplug it, and to remove IC15, & IC16 which will both be required on the Econographics, and also IC20. The few links are easily made, but if you are in any doubt, get a second opinion that they are correct, and that there are no solder splashes or bridges. (Character generators come expensive - like 10.00 each!) Replace IC20 on the N1 and insert all other i.c.s into Econographics.

With all this done, you are now ready to offer up the Graphic board. You are supplied with a short strip of plastic U and two screws, with which you fix the U to the board. Under the U you can place two sticky pads, which are also supplied, and stick the graphics board into position on the N1. It is advisable to try this a few times to be sure you know where everything is to go - the pads don't shift easily! At the same time as offering the unit up for final fixing, you also have to connect the 24 way umbilical with the graphics board lying beside the N1. Check that everything is working properly. You should be able to get the extended graphics set by holding down the graphics key and typing (assuming you have an N2 keyboard, or have modded the N1 one - otherwise use the Nas-Sys K options - Ed.). Then, all being well, fold the graphics board into position having removed the paper off the tabs.

The Graphics Rom supplied is the NASGRA, as used on the N2. It is 2716 compatible (single 5v rail), so the possibility of custom character sets is now with you. The whole unit, once you have found the track errors, works very well, albeit with a slight degradation of sharpness in both the standard character set and in the graphics set. This would look like a timing fault on the unit, but is not acceptable. Any suggestions on how to clear it up would be most welcome. It is to be regretted that the boards have the above described track errors on them, and no mention of it in the accompanying documentation, as it is easily cured - either on the etched board by the constructor as described, or on the original negatives/positives for print down (a knife and some opaque is very easy to use).

In case the track fault is peculiar to one series of boards, those I have seen with it were marked (on the component side) PF-020. One last thing, the tants go with their marked sides facing away from their legends.

Extra Econo-Graphics for Nascom 1.

by S.J. Oakes.

For those of you who feel that Nascom's Econo-Graphics kit at 30.00 is not cheap enough, try this instead. It was built for a total of 17.96, including all sockets, cable etc.

The circuit shown will fit comfortably on a 65mm by 80mm piece of Vero board. Solder the I.C. sockets first, breaking the Vero tracks where necessary to separate pins on the same track. Then wire point to point with thin solid core cable following the circuit diagram. Use the tantalum bead capacitor to decouple the 5V power supply to the Character Generator and the Graphics ROM and the ceramic disk to decouple the supply to IC15 and the buffer.

Use ribbon cable to attach the 24 pin and 16 pin DIL plugs to the Vero board. About 180mm should be enough. Pins 1 and 2 on the 74LS00 should be taken to IC17 on the Nascom board - pin 19. If the pin is bent out sideways, then a single soldercon socket and a small piece of heatshrink sleeving will make a socket to attach the wire. On the underneath of the board IC28 pin 18 should be joined to IC17 pin 18 to latch Bit 7 of the Video memory (used as Graphics select).

At this stage I took the opportunities to make the modification to IC18 detailed in previous issues (drive the clock divider chain and IC 17 from Pin 12 of IC 18 rather than Pin 5.)

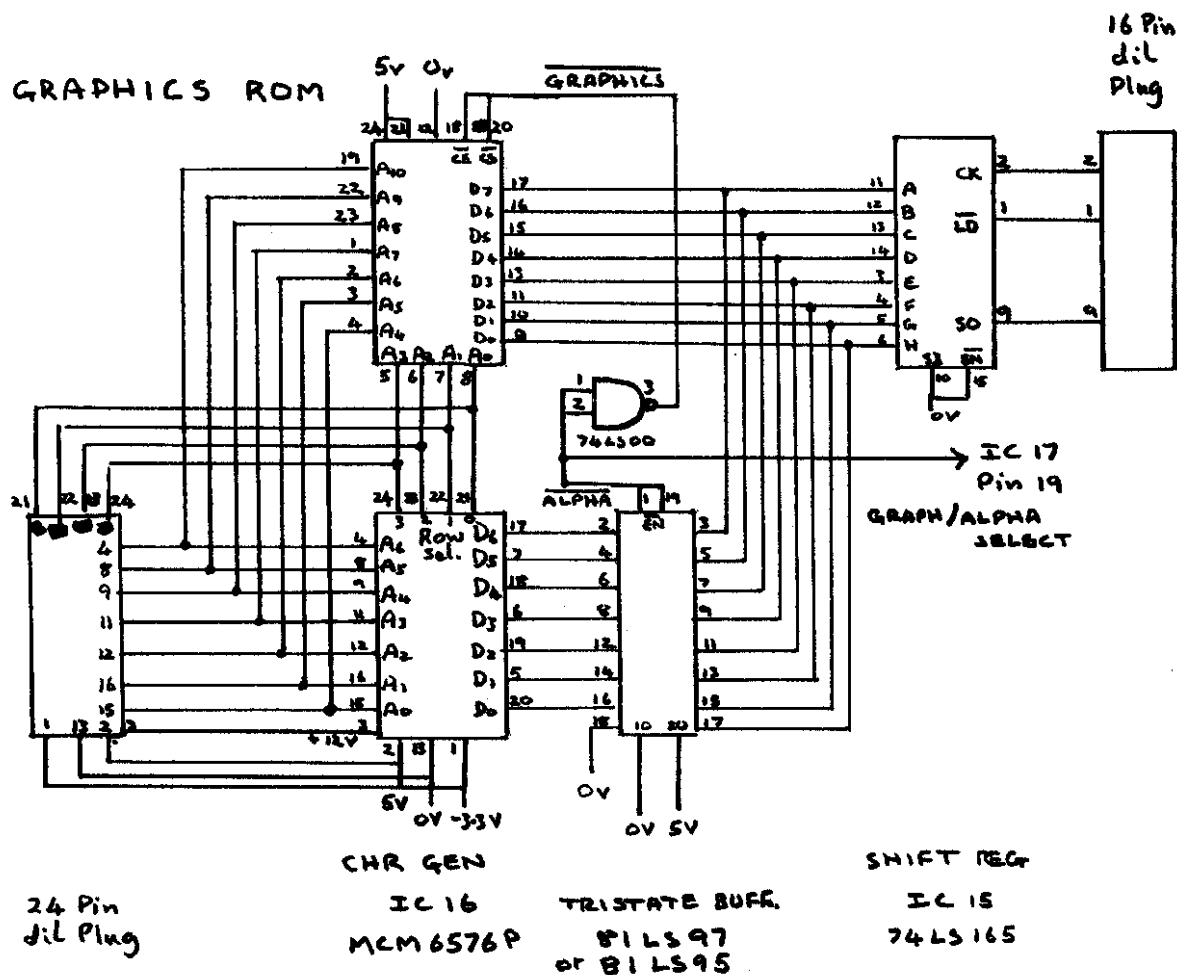
The Graphics board is now ready for testing. Remove the Character Generator IC 16 and the shift register IC 15 from the Nascom board and plug the 24 pin plug in place of IC 16 and plug the 16 pin plug in place of IC 15. Insert IC 15 and the 74LS00 and the 81LS95 into their places on the Graphics board. Switch on and check that nothing drastic happens.

Switch off. Insert the Character Generator into its place on the Graphics board. Switch on. Your Nascom should now work normally but if you try to put Graphics characters on the screen you will get white rectangles. If you try to display a single graphics character or alternate spaces and rectangles then it may not display properly. This is normal.

Switch off and insert your expensive Graphics ROM. This time when power is applied it should work perfectly.

Parts needed:

- | | |
|---------------------------------|----------------------------------|
| 1 81LS97 or 81LS95 | 1 Graphics ROM |
| 1 74LS00 | 1 Ribbon cable |
| 2 24 Pin DIL sockets | 1 14 Pin DIL socket |
| 1 20 Pin " " | 1 16 Pin " " |
| 1 16 Pin DIL Plug | 1 24 Pin DIL Plug |
| 1 1uF Tant. bead Capacitor 10 v | 1 0.01 uF Ceramic Disc Capacitor |



GRAPHICS BOARD FOR NASCOM I

Sound Board

The PHG Electronics Programmable Sound Generator

A review by D. R. Hunt.

=====

A couple of weeks ago a new, some what noisy, goodie appeared on the home computer front. It is the PHG Electronics Programmable Sound Generator (PSG for short). The PSG is supplied built, using a good quality single sided pcb about 4" square, and fitted with a 3.5" loudspeaker. The pcb is drilled for mounting pillars to support speaker and the card itself, but no pillars are supplied. The card was terminated with a 'gold flashed' single sided edge connector, the connector being supplied, but the lead to the Nascom PIO is not. There aren't many components on the board, the chip that does all the work is the General Instrument AY-3-8910 three channel synthesizer, and there are three other support chips, including a small power amp chip to drive the speaker. A few electrolytic capacitors, and a couple of resistors complete the components. The PSG comes with two manuals, a well produced and printed instruction manual of about 12 pages, and a 64 page booklet produced by GI giving very detailed operating instructions for the AY-3-8910 chip. No circuit diagrams were supplied. A demonstration tape recorded in Nascom 2 format at 1200 BAUD was supplied, this assumes that Nascom Basic and 16K of RAM are available.

Connecting the Nascom PIO lead to the connector is straight forward and well illustrated in the PSG manual, and short lengths of heat shrinkable sleeving are supplied to insulate and strengthen the connections. The PSG requires a single rail +5 volt supply, and this is drawn from the Nascom via the PIO cable. Adjacent to the connector is a 4 pin stripline connector, and as supplied, a plug is fitted connecting the four pins together. The manual describes this as the output and phones connector, and appears to parallel all three channels and connect them to the onboard audio amplifier. Although this is described in the manual with an illustration showing links, the layout is slightly confusing as the links are where the output coupling capacitors are fitted, and the description does not tie in too well with the way connection is intended. It is a pity no circuit diagram was supplied, as it would have shown quite clearly what was intended. To be fair, this, and the lack of circuit diagram are the only criticism that can be leveled at the PSG.

On plugging the unit into the PIO, and loading the tape, the PSG performed well. The demo consisted of five effects. The first, John Brown's Body, gave a pretty fair imitation of a harmonium, and was followed by some birds twittering. The third was very short, and would be a 'must' in any Space Invaders program, consisting of a missile flying across the screen, and hitting it's target with a commendable explosion. This effect could drive anyone not playing Space Invaders nuts within minutes. This demonstration was convincing because it showed how, given instruction, the PSG will continue to generate sound until updated by the program, allowing the computer to manipulate the graphics to suit. The graphics were fast, very impressive, particularly as the graphics were manipulated by Basic. The fourth demo was a bit of Bach's 4th Brandenburg Concerto. The program was written in three parts, each separate channel handling one part. Following the instructions for fitting stereo phones, the effect was very impressive. The tone of the music was definitely 'electronic', but Walter Carlos wouldn't have worried. It showed the programmer wasn't necessarily a musician, as there were at least three bum notes in the Brandenburg (easily corrected in the program). The last demo was Frere Jaque programmed as a three part 'round'.

The demos were good and gave an idea how the PSG performed. Now how about trying it for yourself? The manual gave examples of adapting the demo program to allow you to compose you own music and this was easy. Given a modest amount of patience it was easy to program recognisable tunes, and by presetting the noise register, other simple effects as well. Anything more elaborate required careful reading of the AY-3-8910 booklet, along with a reasonable

knowledge of machine code. (Basic could have been used, but would slow the response of complicated effects.) Within a couple of hours, we had a very passable 'sea lapping on shore' sound with nice changes in level for big and small waves, with a few random seagulls thrown in for good measure. The versatility of the AY-3-8910 is unending, and given a bit of external tone forming, limitless (well almost).

The AY-3-8910 contains three independant 'tone' channels, each covering a range from the sub-sonic to the supersonic. Each 'tone' channel is preset by a twelve bit word contained in 2 registers. There is a fourth channel, a 'noise' generator, which will produce random width pulses with a spectral distribution controlled by a five bit register. It can produce 'noise' from a quite throaty 'shuush' to a high pitched 'sssssss' (if you get my meaning). One register controls the functions of the channel mixer. Enabling each of the three 'tone' channels independantly, and allowing the 'noise' channel to be mixed with any or all of the three 'tone' channels. Three registers control the amplitude of the three output channels, these require four bits for each. A fifth bit on each amplitude register switches that channel through the 'envelope shaper', ignoring the preset amplitude in the lower four bits. The envelope period is controlled by a 16 bit control word in two registers, and may be preset from milliseconds to seconds, and affects those channels directed to it. The envelope shaper could produce quick annoying clicks with each change of envelope level when run close to it's maximum period. These could probably be filtered out externally. The last register controls the envelope shape and cycle. Allowing positive or negative attack, continuous output, alternate or one shot outputs, and lots of other permutations. There are two other registers which work on conjunction with an external EPROM/ROM pack for remote and/or preset control. Although the PSG board has these connected to a 16 pin socket, they play no part in the use of the PSG.

		Bit							
Register		B7	B6	B5	B4	B3	B2	B1	B0
R0	Channel A Tone Period			8-bit fine tune A					
R1						4-bit course tune A			
R2	Channel B Tone Period			8-bit fine tune B					
R3						4-bit course tune B			
R4	Channel C Tone Period			8-bit fine tune C					
R5						4-bit course tune C			
R6	Noise Period					5-bit noise period			
R7	Enable	IN/OUT		Noise			Tone		
		IOB	IOA	C	B	A	C	B	A
R10	Channel A amplitude				M	L3	L2	L1	L0
R11	Channel B amplitude				M	L3	L2	L1	L0
R12	Channel C amplitude				M	L3	L2	L1	L0
R13	Envelope period			8-bit fine tune envelope					
R14				8-bit course tune envelope					
R15	Envelope shape/cycle					CONT	ATT	ALT	HOLD
R16	I/O PORT A data			8-bit parallel I/O					
R17	I/O PORT B data			8-bit parallel I/O					

The PHG instruction book gave a machine code routine for updating the PSG registers, and although effective was a little tedious. For experimenting, we found the update routine below a little more handy. The Nascom screen set up with a Tab command to display the register table at the top. Beneath this was an Execute command to execute the command then return to NAS-SYS. Beneath this was an 'M' command. By entering the M command, the cursor could be skipped around the Tabbed register table on the screen, a few registers changed here and there, and then the cursor skipped down to the 'E' command and the routine re-executed to see the affect. That then brought the cursor back to the 'M' command ready for the next try.

Title PHG Electronics PSG register update

.Comment \

This subroutine is intended to replace the register update routine given as an example in the PHG Electronics PSG Manual. The calling program updates the register image starting at REGTAB, and then call this subroutine. This then updates all the AY-3-8910 registers at once. It is assumed that the ports have been previously initialized to the output mode.

Carl Lloyd-Parker 03-07-81

.Z80

```

0004          PBASE    EQU 04H          ; Port base address

0004          PCONT    EQU PBASE        ; PSG control port
0005          PDAT     EQU PBASE+1      ; PSG data/address port

0000'  21 001F'      UPDATE: LD HL,REGTAB ; Point to register table
0003'  01 1000      LD BC,1000H ; Prime register count in B
                                ; register address in C
0006'  79          LOOP:  LD A,C        ; Get register address
0007'  D3 05      OUT (PDAT),A ; Send register address to PSG
0009'  3E 03      LD A,03H          ; Strobe into address latches
000B'  D3 04      OUT (PCONT),A
000D'  AF          XOR A            ; Cause strobe pulse
000E'  D3 04      OUT (PCONT),A
0010'  7E          LD A,(HL)        ; Get register data
0011'  D3 05      OUT (PDAT),A ; Send data to PSG
0013'  3E 01      LD A,01H          ; Strobe into data latches
0015'  D3 04      OUT (PCONT),A
0017'  AF          XOR A            ; Cause strobe pulse
0018'  D3 04      OUT (PCONT),A
001A'  0C          INC C            ; Inc PSG register address
001B'  23          INC HL           ; Inc register data pointer
001C'  10 E8      DJNZ LOOP        ; Go round again for next
001E'  C9          RET             ; End of routine

001F'  01          REGTAB: DEFB 1 ; R0 Channel A fine tune
0020'  01          DEFB 1 ; R1 Channel A course tune
0021'  01          DEFB 1 ; R2 Channel B fine tune
0022'  01          DEFB 1 ; R3 Channel B course tune
0023'  01          DEFB 1 ; R4 Channel C fine tune
0024'  01          DEFB 1 ; R5 Channel C course tune
0025'  01          DEFB 1 ; R6 Noise channel period
0026'  01          DEFB 1 ; R7 Channel Enables
0027'  01          DEFB 1 ; R10 Channel A amplitude
0028'  01          DEFB 1 ; R11 Channel B amplitude
0029'  01          DEFB 1 ; R12 Channel C amplitude
002A'  01          DEFB 1 ; R13 Fine envelope period
002B'  01          DEFB 1 ; R14 Course envelope period
002C'  01          DEFB 1 ; R15 Envelope shape/cycle
                                END

```

Plus points. Supplied built, with almost everything required to make it go. Good clear manual, and GI's data booklet also supplied. Good demo tape.

Minus points. No circuit diagram, one connection diagram confusing.

Generally fair value for money (you could make one a lot cheaper, but that's only the bits, and doesn't include the sweat involved figuring it all out). A nice workman like job. Available at 39.50 + VAT from PHG Electronics, 27, Fenwick Drive, Rugby, Warks. And some Nascom dealers.

sound

THE AY-3-8910 PROGRAMMABLE SOUND GENERATOR MEETS THE NAS-BUS

Andy Tipler
Huntingdon

I am sure many readers are aware of the availability of the General Instrument's AY-3-8910 programmable sound generator ("super-easy to interface to the S100 and other buses...") from several suppliers currently advertising in the various computer magazines.

In my innocence, I bought one of these devices and, although it's not too difficult, I found that interfacing is not as straight-forward as it first appears. I thought that my experiences and subsequent design of a working NAS-BUS compatible board (that's the correct terminology isn't it?) may be of interest to other NASCOM 2 users.

In the first place any would-be user intending to interface and use the PSG chip with a NASCOM 2 would be well advised to study the NASCOM documentation (again) and to refer to several publications on using the PSG (see list of references at end of this article). In this respect I will not give detailed descriptions of the programming techniques involved, but rather concentrate on the hardware considerations specific to interfacing to the NASCOM.

What does the AY-3-8910 PSG chip do?

The PSG chip is a complex user-programmable digital to analogue converter specifically designed for audio sound output. The logic is TTL compatible and is thus able to be linked to the NAS-BUS. The chip contains 16 user-accessible registers, each containing 4 to 8 bits depending upon its application. Two of these registers are dedicated to supporting the two 8-bit parallel I/O ports provided - useful for joysticks, A to D converters etc (handshakes and interrupts are not catered for). The remaining registers dictate the type of sound generated. Three analogue outputs are provided, these can be adjusted independently for frequency and amplitude. A noise generator with variable pitch can be mixed with any of the above outputs. There is also an envelope generator, the profile of which can be programmed for frequency and shape. One of the apparent strong selling points of this device is that it will operate independently of the host CPU. This is certainly true for continuous sounds (boring) or sounds controlled by the envelope generator, but almost any "interesting" sound will require a lot of attention by the CPU. One way of improving this situation might be to use the Z80 CTC chip to interrupt drive the PSG at predefined time intervals leaving the CPU to get on with its business most of the time. However despite this, the AY-3-8910 does produce some amazing sounds and is very easy to operate.

Options available for interfacing the PSG to a NASCOM 2:

a) Configure it as part of the memory. The articles in PCW and PE use this approach to link the PSG to a rival (?) computer kit, using PEEK and POKE to address it. While this is fine if you have odd memory locations scattered around your memory map, it does no justice to a system such as the NASCOM 2 where every bit of memory may be put to more efficient use.

b) Attach it to the PIO port. This is very easy to do, however if you adopt this approach you will soon find that it does give a large software penalty over alternative methods: the PSG is controlled by pulses, the timings of which may be critical, so the PIO (which gives a latched output) must be repeatably set and reset, applying delays if necessary, in a specific order to control the PSG. This effectively more than doubles the amount of software required to drive it, making life rather tedious.

c) Configure the PSG directly as an I/O port. This is the approach I have adopted. It is not too difficult to build, inexpensive, plugs straight into the NAS-BUS (absolutely no wires), very easy to programme and up to 124 (Heaven forbid) PSGs can be used with no modifications to the NASCOM (except a link to the National Grid to drive them plus a rather large mother-board).

The working design:

The circuit works by controlling the status of the BC1 and BDIR inputs on the PSG according to the following truth table -

BDIR	BC1	Function
1	1	Latch register address
1	0	Load register with data
0	1	Read register contents

IC 1 decodes the address bus to configure the PSG as ports 8 and 9 (address lines 8 to 15 are not used for port addressing on the NASCOM 2). Rearrange these inverters if alternative addresses are required (but don't use ports 0 to 7 as these are already dedicated on the main board).

If the predefined address is received then the output of IC 3 will go low. Any I/O instruction will also push the IORQ line low (active). These two signals are inverted (ICs 2a and 2b) and fed into the inputs of ICs 4a and 4b. Address line A0 is then able, by using IC 2c, to select either IC 4a or 4b, giving a low output. As the circuit stands, addressing port 8 will give a low output for IC 4a, and port 9 a low output for IC 4b.

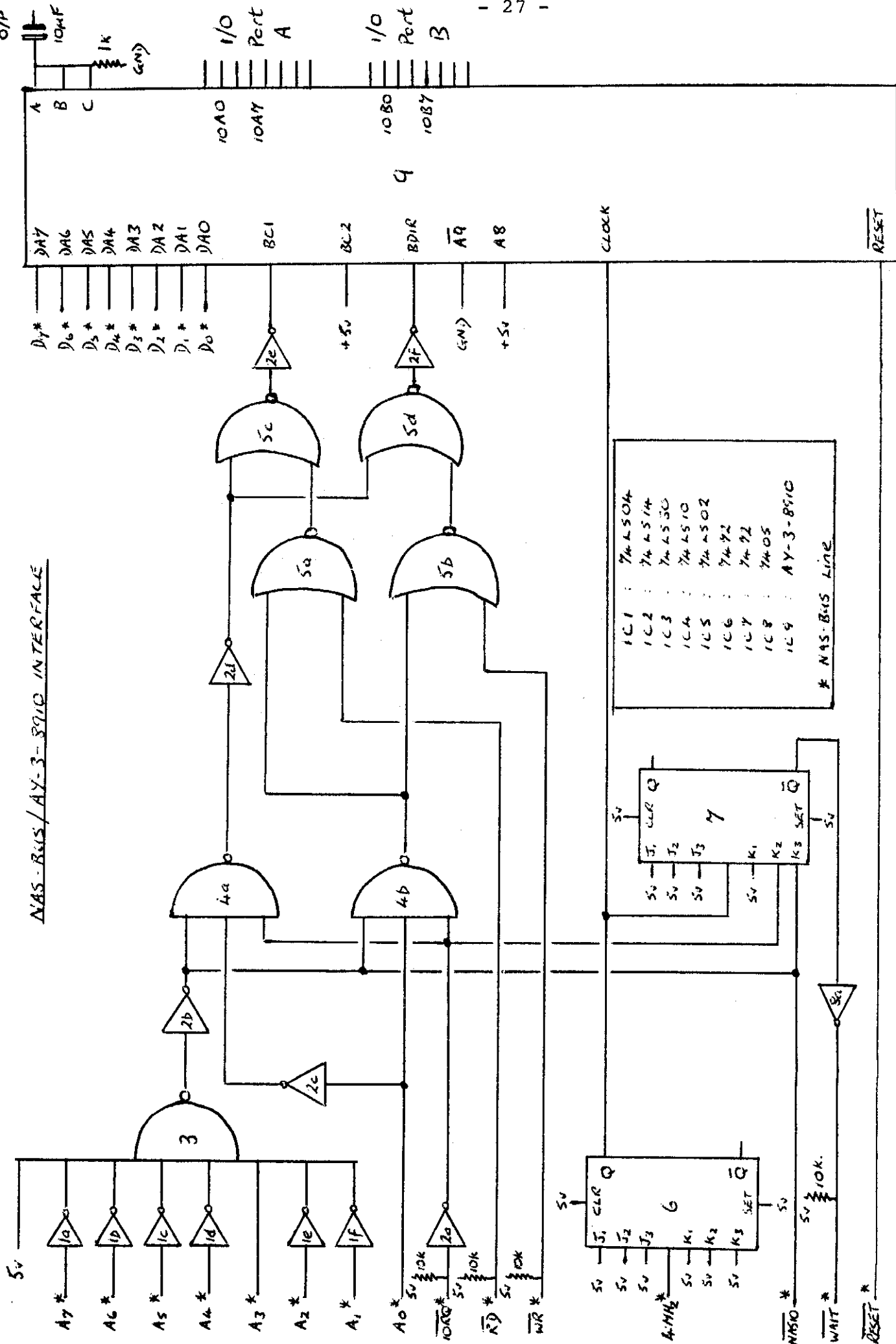
If port 8 is addressed, then the inverted (IC 2d) output from IC 4a raises the inputs of ICs 5c and 5d making their outputs low. These signals are inverted (ICs 2e and 2f) and fed into the BC1 and BDIR inputs of the PSG. As both will then be high, a register latch operation has been performed (see truth table at beginning), and the contents of the data bus will select the appropriate register. This operation does not take into account the CPU RD and WR lines. The programmer must ensure that "OUT" instructions only are used for port 8 (or else the CPU and the PSG will both read the data bus simultaneously with no-one supplying the data). If required, the WR line could be incorporated but it hardly seems worth the effort.

If port 9 is addressed the low output from IC 4b is NORed with the RD and WR lines in ICs 5a and 5b respectively. For a read ("IN" or "INP") instruction, the output of IC 5a and hence the BC1 input, will go high and the PSG will transfer the contents of the currently selected register onto the data bus. For a write ("OUT") instruction the same will apply to IC 5b and the PSG BDIR input, allowing the PSG to load the currently selected register from the data bus.

Lines BC2, A8 and A9 are additional PSG select signals but as their status in this circuit needn't alter, they are tied as shown.

For 4 MHz operation, the clock is divided (IC 6) to provide the 2 MHz maximum the PSG can handle and is further divided (IC 7) and controlled by the address enable output (IC 2d) and the IORQ line (IC 2a) to give a single positive-going pulse which is inverted (note open collector inverter) to provide a WAIT state for the CPU. I have used 7472 JK flip-flops here because I have a box full of them - a 7474 may be preferable, the unused NAND gate (IC 4c) could be used to link the address enable and the IORQ lines into the system to control the pulses. If you are using the CPU at 2 MHz then forget about ICs 6 and 7 and the WAIT states - just feed the system clock into the PSG.

NAS - BUS / AY-3-8910 INTERFACE



The address enable line from IC 2a is also used as the NASIO signal. Although the NASCOM 2 is fitted with a switch (LSW2/8) to allow either on-board or external ports to be used, as it stands you cannot use both. The NASIO signal provides a means of achieving this, but unfortunately only one signal can be fed to this line (unless you use expensive tri-state logic) as TTL outputs don't like mixing with each other. So if you intend to use more than one I/O board in the system you must be careful. One way of catering for all eventualities is to use the circuit in Figure 2. Fit this onto one (only) board which uses the bus, switch LSW2/8 to "EXT" and forget about NASIO. There are sufficient unused ICs in the circuit of Figure 1 to build this. If you are using the NASCOM I/O board, check how it handles NASIO.

Practical details:

Vero Electronics make an 8" x 8" copper-tracked perforated card with gold-plated "fingers" (i.e. NAS-BUS compatible) for about #5 (part number 09-0092K) and although not glass-fibre, it does present a cheap alternative to the prototyping boards currently available for DIY circuits, and is more than adequate for this application.

Remember that the NAS-BUS contains lines directly linked to the CPU and other important chips in the system, so be VERY careful of what happens to these lines when they enter your circuit. As a suggestion - letter each track with a felt tip pen as to its function, check and then check again before starting work. Cut any tracks carrying unused signals (NMI, INT, HALT, A8-A15 etc.).

Layout appears not to be critical - refer to a good TTL primer (eg. TTL Cookbook, Lancaster) if necessary. Check for shorts, tracks still needing to be cut etc. Test the board without the chips and check that all voltages are in the right places. Fit the TTL devices and adjust LSW2/8 to EXT and check that the keyboard still works, if possible check the correct operation of the 2 MHz clock (if you are using the divider circuit) and the BC1 and BDIR signals with a scope using the following programmes:

Routine to check PSG register latch:

```
LD A,#0F
LOOP OUT (8),A
JP LOOP
```

The BC1, BDIR, NASIO and all DATA lines should all give simultaneous positive pulses.

Routine to check data write:

```
LD A,#0F
LOOP OUT (9),A
JP LOOP
```

The BDIR, NASIO and all DATA lines should all give simultaneous positive pulses, BC1 is kept low.

Routine to check data read:

```
LOOP IN A,(9)
JP LOOP
```

BC1 and NASIO should give positive pulses, the data lines will reflect the contents of the addressed register and BDIR should remain low.

If WAIT states are used, these should be active (low) within the pulses described above.

If all is well, plug in the PSG chip (taking precautions against static charges), cross your fingers and switch on. Hopefully everything should now work.

Addressing the PSG:

This is very simple - select a register by writing to port 8 and then load it or read from it with port 9.

eg. to load register 7 (PSG output enable) with 244 (tone channels A and B, noise channel A enabled) simply write:

(BASIC) OUT8,7:OUT9,244

(ASSEMBLER) LD A,#07
OUT (8),A
LD A,#F4
OUT(9),A

And to read, say register 7:

(BASIC) OUT8,7:PRINT INP(9)

(ASSEMBLER) LD A,#07
OUT (8),A
IN A,(9)
:Contents now in A

Further ideas:

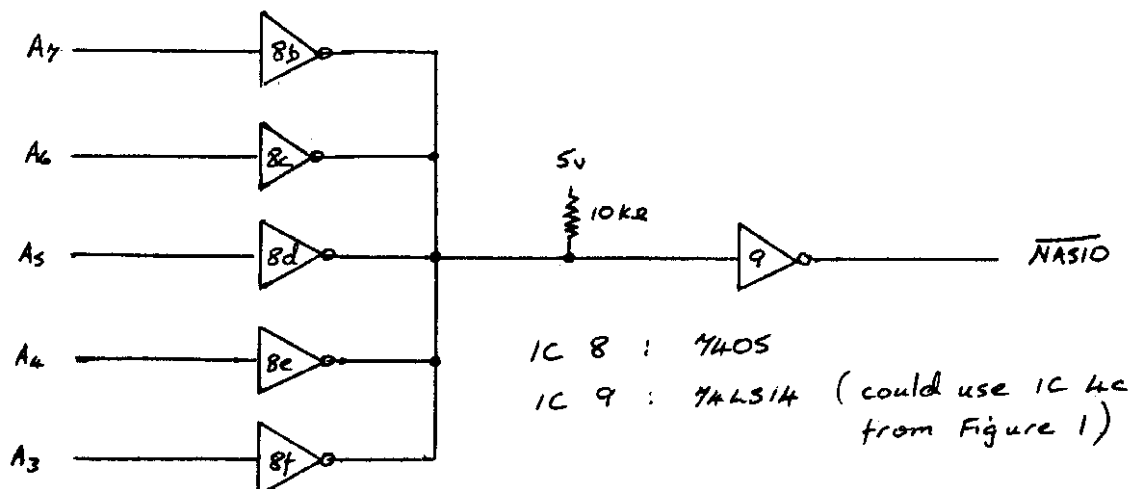
If you anticipate using several PSGs in your system, try using a 2708 EPROM to decode the address bus and control signals. The eight outputs could be paired to provide the BCI and BDIR signals for four PSGs. Optional simultaneous access to all of them would be possible (in the write mode anyway) and the hardware would be much easier and cheaper to build.

Suggested reading:

- The data manual for the PSG from the retailers
- BYTE, July 1979
- Personal Computer World, October 1980
- Practical Electronics, September 1980
- TTL Cookbook, D. Lancaster

FIGURE 2

NASIO CIRCUITRY FOR SEVERAL PORTS ATTACHED TO BUS



N2 Keyboard by David Pears

THE NASCOM 2 KEYBOARD

Nascom owners with keyboard problems (in my case a dry joint in one of the wire links) are confronted with a complete absence of information in the manual. Even a copy of the keyboard circuit diagram (with wrong pin number for PL3) from a friendly Nascom dealer only took me part of the way and the following notes may help others track down faults. The 56 keys (ignoring Reset and the duplication of SPACE) are at the intersections of 8 row conductors and 7 conductors. These are shown as column conductors and row conductors respectively on the keyboard circuit diagram, and also the Bits & P.C.s keypad circuit diagram but I am using Nas Sys terminology. I denote the rows R1 to R8 and the columns by their corresponding bits D0 to D6 in the data bus. The layout is then as in the following table which also gives pin numbers of keyboard IC5 for the row interrogating pulses and pin numbers of PL3 for the columns.

ROWS*	COLUMNS (WITH PL3 PINS IN BRACKETS)						
	D0(1)	D1(3)	D2(5)	D3(7)	D4(9)	D5(11)	D6(13)
R1(1)	BS	ENT	-	CTL	SHIFT	@	CH
R2(2)	H	B	5	F	X	T	↑
R3(3)	J	N	6	D	Z	Y	←
R4(4)	K	M	7	E	S	U	↓
R5(5)	L	,	8	W	A	I	→
R6(6)	;	.	9	3	Q	O	GR
R7(7)	:	/	0	2	l	P	[
R8(9)	G	V	4	C	SP	R]

* IC5 pins in brackets.

Basically the computer finds out which key is depressed by pulsing the row conductors in cyclic sequence and seeing which data line picks up pulses. For example "5" is detected as pulses on D2 at the time of pulsing R2. The start of the sequence is marked by making the pulses on R1 much wider than those on R2 to R8. IC6, IC3 and IC5 form a counter circuit driven by pulses on PL3/11 and 10 under software control, ie part of the Nas Sys keyboard routine, and the interrogating pulses appear on the pins of IC5 in the table above. The pulses picked up by the column conductors are buffered by IC9 and IC1 and latched by IC8, IC7, IC2 and half of IC4. The latches are cleared by a clear pulse from the counter circuit, (which also includes the other half of IC 4).

Nas Sys forms a word identifying row and column when it detects anything on any of D0 to D6 at the right time and uses a table to calculate the ASCII code for the corresponding character. The correspondence can be completely arbitrary, since a table is used,

which explains why SHIFT is only partly regular in its behaviour and why CTL is highly irregular - as a little time with an ASCII table and the Nascom keyboard will show. SHIFT, CTL and GR (graphics) can be separately identified as depressed in conjunction with any other key.

The input of data from the keyboard is entirely software dependent and the computer readily can (and does) get into a state in which no input from the keyboard is possible. Control can always be regained by Reset which acts in a totally different way. It puts an earth on the Z80 reset input, via PL3/12, and forces a reset to whatever address is set up on LSW1/1 to 4.

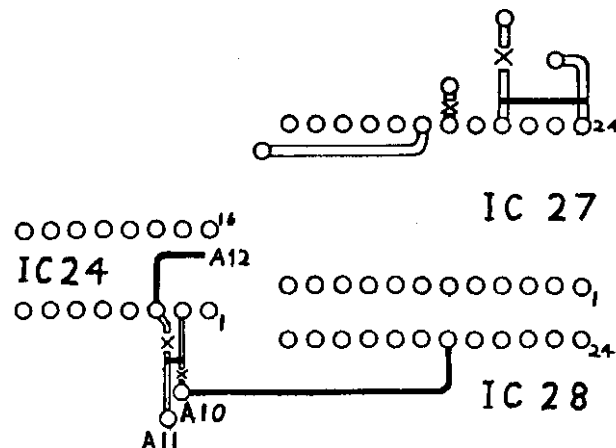
The above table will help to localize a keyboard fault (or a fault around IC25 on the mainboard) which, as in my case, affects the whole of one column or the whole of one row of the table.

NASCOM HARDWARE MODIFICATION

A. O. ZIENKIEWICZ

MODIFICATION TO 'TYPE A' RAM BOARD

This enables the use of 2716 type EPROMs in the four EPROM sockets allowing 8K of firmware to be held in memory.



Cut tracks near:-

IC 24 p2
IC 24 p3
IC 27 p21
IC 27 p19

Wire link:-

IC 24 p2 to A11
IC 24 p3 to A12 (At hole near Bus line 42)
IC 28 p19 to A10
IC 27 p21 to IC 27 p24
P5 to Two pads as for WIRED-OR RAM select.
(i.e. for C000-DFFF link P5 to 9+10)

Note that if the selection is to an odd '000 the sockets will not be in consecutive order.

To use 2732 type EPROMs for a total of 16K firmware, cut tracks as above, then cut tracks near p20 for each socket.

Wire link:-

IC 28 p19 to A10
IC 28 p21 to A11
IC 24 p16 to IC 24 p2+p3

The Chip-select (p20) of each socket is now wire linked directly to an appropriate 4K decode pad.

2K's on an N2

USING 2K EPROMS ON A NASCOM 2 E. P. T. Anderson

The NASCOM 2 is equipped with eight 24 pin sockets intended for 1K static RAMs or 1K EPROMS of the 2708 type. Although the initial literature - and advertising - stated that the use of these sockets was very flexible, it omitted to state how to use them for the now economical and hence popular 2516/2716 2K EPROMS. Furthermore the writer failed to understand the Block A/Block B etc. connections in spite of help from INMC magazine.

The N2MD ROM (IC47) give some convenient chip select pulses which are 4K wide so together with an A11 or A11 signal it is quite easy to define 2K boundaries. As well as suitably enabling the EPROMS it is also necessary to enable IC44 and this is done by the addition of simple diode gating.

STEP 1 (Solder side of board - see fig 4)

Wire all sockets to A10. This is done by strapping all pins 19 to A10 which is available on Pin 19 of the monitor ROM (IC 34)

STEP 2 (Solder side of board - See Fig 4)

For EPROMS which are to be operated on the lower 2K of a 4K block wire pin 18 to A11 which is available on Pin 18 of the BASIC ROM (IC 43)

STEP 3 (Solder side of board - See Figs 2 & 4)

For EPROMS which are to be operated on the upper 2K of a 4K block wire pin 18 to A11. A11 does not appear to be available so it is necessary to invert A 11. As the 1K chip select signals from IC46 are not now required, remove the 16 pin 74LS155 from this socket and replace with a 14 pin 74LS04 (or 7404) to the top of the socket (Pin 1 to Pin 1). Luckily, A11 is present on pin 3 which is the input of one of the 7404 inverters and hence A11 is then available on pin 4. Hence connect pin 4 to pin 19 of the upper EPROMS. Pin 7 of the 7404 must be strapped to the now vacant pin 8 of the socket for the Ov line.

From Fig. 2 it can be seen that the five spare inverters on the 7404 are not loading any circuits - the majority of the connections going to the link blocks adjacent to the EPROMS.

STEP 4 (Solder side of board - see fig 4)

Connect pin 21 of all EPROM sockets to +5 which is present on pin 24.

STEP 5 (Component side of board - see fig 1)

Wire the appropriate 4K block pulse available on pins 9 to 16 of LKSI to pin 20 of the appropriate EPROM via the link block pin.

Generally you will use:

D4	B000 - BFFF	Pin 12
D5	C000 - CFFF	Pin 11
D6	D000 - DFFF	Pin 10

Each 4K block used must also be gated into the IC44a/RDB circuitry and this can be conveniently effected by wiring small signal diodes from the block pulse (pins 10, 11 or 12) to pin 7 on LKS1 (XROM).

Switching EPROMS

Sometimes it may be required to switch different EPROMS into a given memory location and fig. 1 shows a method of doing this for, in my case, alternative monitors. The appropriate block pulse is switched to the appropriate pin 20 but it is necessary to disable the non used EPROM by allowing its Pin 20 to go high via a 1K resistor to +5v. The 1K for the original monitor is already on board (R53) but must be provided for any non-used EPROM. The gating diode for the switched monitor is also taken to LKS1-7.

Multi-Rail EPROMS

All the EPROMS used are single rail as these are readily available and hence the +12v and -5v connections are not used.

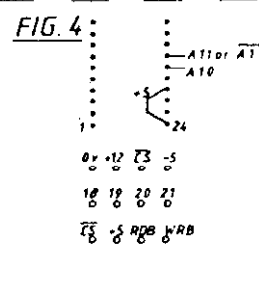
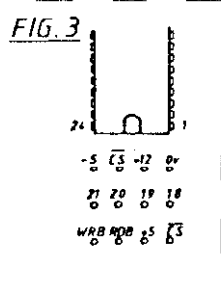
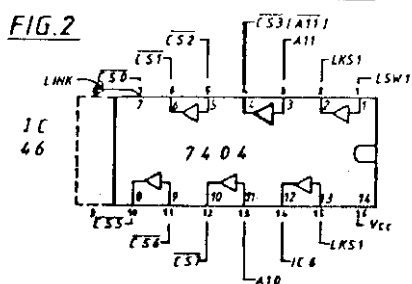
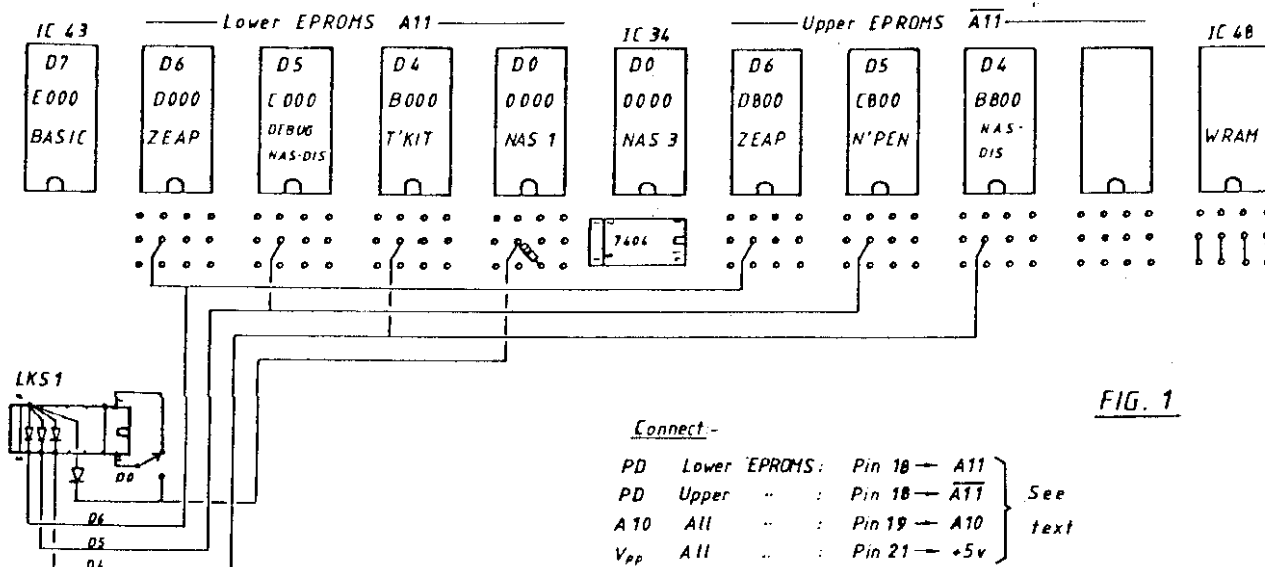
4K EPROMS

As I am still waiting for delivery of a 4K EPROM programmer, I have not tried this yet, but it looks to me as if a 4K EPROM can simply be plugged into a socket with A11 on Pin 18, in which case A11 will not be necessary.

General

Most of the wiring has been on the solder side of the board as I find wiring onto the link pins is not easy - and in any case it does not show on the bottom! Do not try using the spare IC7b and IC7h gates as A11 inverters as they are only buffer gates - not buffer inverters. It took me quite a time to realise this.

The diagram shows how I have populated my sockets but of course any socket can be used by suitable connections of A11 or $\overline{A11}$ and the relevant block pulse to pin 20.



NASCOM 2
MODS TO POPULATE
WITH 2K (or 4K ?)
EPROMS

6 PTH 10.2.8

Disk Review

REVIEW OF GEMINI DOUBLE DENSITY DISK SYSTEM

By Ian Henderson

I had been considering taking the road to disks for a considerable time but had been prevented from doing so by one very small fact - there were none available for the Nascom !! Nascom's own had been promised for some time, but receivership seemed to have well and truly put a stop to that. Then the Comp unit came out and I went to inspect it. There are few words to describe either the hardware or software, all of them unprintable.

Next came the Henelec (meaning 'of Henrys') FDC card, and the Gemini G805 disk system that used that card. I went and had a good look at one that a friend had bought and was duly impressed. Unfortunately, by that time, I had read too many adverts and had decided that I MUST have a double density system. There were noises emanating from the Nascom camp again and I decided to wait for 'the real one'. And wait I did !!

Then, a couple of months ago, INMC80 issue 4 arrived announcing the imminent arrival of the new Gemini double density card. "I'll believe that when I see it" I said. (Have you noticed the tendency for computers to make you start talking to yourself?) So I phoned my distributor and said "Let me know when you get the Gemini disks", "I've got them", "No, not the G805 - the new one", "That's what I mean", "&\$*+#\$... Don't sell them, I'm coming". And so off I went and bought myself a G809, a G815/2 and a G513. If that means nothing to you then read on (it doesn't mean an awful lot to me either so don't worry).

Gemini G809

The G809 is the disk controller card. It is a standard 8x8 80-BUS (ie Nasbus compatible) card and it comes built and tested. It handles 5.25" or 8" drives of single or double density, single or double sided, 48 or 96 track per inch types. (Note, the 50 way connector for 8" drives is not supplied.) Up to four drives can be controlled, and all drives are deselected (thus unloading the heads) after several seconds with no access. According to the manual it has 'variable write precompensation and phase locked loop data recovery circuitry', or more to the point - it works! As delivered it is set up for Pertec FD250 type double sided, double density, 48tpi drives (350K per drive!). Installation is simple - plug it into the Nasbus, switch the Nascom 2 I/O switch to 'External' and go and get a well earned cup of tea!

Gemini G815

The G815 is the disk drive box. This is the same box as that used for the G805 system. It contains either one or two (G815/1 or G815/2) Pertec FD250 drives, and the power supply necessary to power them. On the front an led indicates that the unit is on or off, and on the rear is the mains switch and fuse holder. The unit is fairly attractive and the lid is made of the same plastic coated steel used for the Kenilworth case. With the G815 alongside my Kenilworth the whole system (with my APF monitor and Epson printer) looks quite attractive, or to put it another way, the wife no longer objects!! Installation is again simple - plug the 34 way cable that comes out of the back of the drive unit into the disk card (making sure you get it the right way round), add a mains plug, and plug it in. Now go and get another cuppa!

Gemini G513

So now you have your controller card and drive unit, what next? Oh yes, software! The G513 is a "CP/M 2.2 Package for Nascom based systems." First of all the memory map of your Nascom needs altering to conform to the requirements of CP/M. For N2 owners a new MD PROM (N2MDCPM) is supplied, N1 owners just follow their own instructions. (By the way, for those not wanting to go to CP/M a mysterious item called Polydos 2 is supposed to be available soon.)

Next you install a single 2708 on the Nascom. Set the Reset Jump to F000H, switch on, insert a disk and hey presto, it works. Well mine worked straight away anyway. A friend of mine was not so lucky as he put the 2708 in the wrong way round and KILLED SIMON!! What's SIMON - the 2708 of course, Simple MONitor.

SIMON is actually very clever for 'his' size. He lives at F000H and 'boots' up the disk. If there is no disk present, or a disk without a CP/M system installed is inserted, then control is passed to SIMON and an error message is output. SIMON also contains a keyboard routine, screen routine and several Nas-Sys like commands - Copy, Execute, Fill, Modify, Output, Query, Tabulate and Boot to load the CP/M once you decide what you did wrong last time. SIMON allows you to work out (hopefully) where the fault is should something fail. The other clever thing about SIMON is that he works out whether or not you have got the Gemini 80x25 video card, and uses it if it is there!

So now we are into CP/M, let's read the CP/M manuals. This is where the fun and games begin. There are SEVEN manuals provided by Digital Research, the authors of CP/M, and these all come in the G513, PLUS a 23 page manual from Gemini on their actual implementation of the CP/M provided. Despairing at the look of the DR manuals you open the Gemini one and find it recommending yet another book! Zaks, to be precise, and this is in fact good advise. Why? Well the Digital Research manuals are concise and extremely useful once you have an idea of what you are doing, but as an introduction they are not very readable. On the other hand the Gemini manual is very readable, but limits itself to the BIOS implementation and the routines supplied by Gemini (a Format routine and a very useful Backup routine.) So to the CP/M beginner something like 'The CP/M Handbook' by Zaks should be obtained.

But what about the CP/M itself? Well several friends, owners of other unprintable systems, have had a good look at my Nascom and declared the Gemini CP/M implementation to be far better than their own (Tee Hee). Why? Well, for a start you get TWO CP/Ms for your money. One is set up for the ordinary Nascom memory mapped screen (now residing at F800H), and the other is set up for the Gemini 80x25 IVC (Intelligent Video Controller) card. You just choose the one you want, neat eh? When I started I used the Nascom screen and the relevant CP/M, then, when funds allowed, I bought the IVC and used the other CP/M. (Ed. - the author also sent us a review of the Gemini G513 IVC, but we're holding that for the next issue.) The Nascom screen driver also incorporates a number of the functions available on the IVC. These include Cursor addressing, Clear to end of line/screen, plus another very useful feature. (See below.)

The CP/M BIOS also supports 'Auto-density' on any drive other than 'A'. This means that drive A must have a Gemini double density format disk installed, but other drives may read or write disks in either double density or in the same single density format (SD Systems) as the G805/Henelec, completely automatically. (Ed. There is a slight problem here, see elsewhere in this issue for the solution.) The FORMAT routine supplied allows either S or D density formatting. Also supplied is a BACKUP program that allows entire disks, including the system tracks, to be copied very quickly - even on a single drive system. This is much better than having to 'PIP' and 'SYSGEN'.

There are several very useful features in the keyboard routine. Pressing 'Control/Enter' toggles an 'Upper Case Lock', handy for assembler work and for running some CP/M programs that don't seem to respond to lower case. Another feature is 'Control/Backspace' - this toggles the function of the () keys so that square brackets [] are returned instead. This is invaluable to the N1 keyboard owner who otherwise cannot use many of the options provided by certain CP/M utilities. And on the N2 keyboard the 'CH' acts as a 'TAB' key - very useful in CP/M editors.

Printer support is extensive, allowing either a serial, serial with handshake, or parallel printer to be driven. The parallel driver allows for an 8 bit interface and connecting my Epson was a doddle. You can also set a form feed control, the number of lines per printed page, and the gap between pages by modifying a 'Patch area' in the BIOS. This is easily done, and other conditions that can be altered are the default I/O byte, the cursor blink rate (or cursor type if the IVC is in use), the repeat rates of the keyboard (both initial and running rates), the ports used for I/O (useful if an I/O board is being used), and - wait for it - the keyboard code used to select EDIT mode!!

Now Nas-Sys owners are probably unimpressed with the last sentence, but for CP/M users an Edit facility is a joy very rarely found elsewhere (if at all?). The ability to enter Edit by typing 'GRAPH 1' (unless you have changed the code in the Patch area) and hop around the screen is great. (My enthusiasm is because I didn't read the manual fully for a while, and did not realise that this facility was included!) Whilst in Edit the cursor changes shape, reverting to normal when 'Enter' is pressed and Edit mode left.

CONCLUSIONS

I bought the Gemini disk system with a certain amount of reservation. After all, Lucas are supposed to be introducing their own card soon, and I wondered whether I should wait yet longer for this. However, I am told that this still uses the design started well before Nascom went into Receivership, and a nearly 3 year old design is very old in this business. Plus the Gemini software is excellent and supports the superb IVC; will the Nascom?

I have no reservations in recommending this system. No disks are cheap, but this package costs little different to most. A great deal of thought has gone into the CP/M BIOS making the CP/M on this system a lot nicer and easier to get on with than on any other I've used (and I have used quite a few). The author (David Parkinson, I believe) has to be congratulated on his work.

On the hardware side, the system is fast, stores a respectable 350K per drive (340K without the system/directory track), and has been absolutely 100% reliable in the thrashing that I have been giving it. With the IVC my 'Nascom' (?) is no longer a hobbyist's toy, but a professional system

One minor closing niggle though, Gemini. Considering the overall cost of this lot, howabout throwing in the Zaks book for free? Then it's unbeatable!

SUBS

INMC80 SUBSCRIPTIONS

I would like to subscribe to INMC80 News for 12 months, starting with the INMC80-6 issue. I enclose my cheque / postal order / international money order (but NOT a French cheque) payable to 'INMC80', value:

UK Subscriptions: 6.50
Overseas Subscriptions: 8.00

This is a re-subscription / new subscription. (Please delete as appropriate)

Name :

Address:

To: INMC80 Subscriptions,
c/o Oakfield Corner,
Sycamore Road,
AMERSHAM,
Bucks. HP6 5EQ.

Disks & CP/M

Disks and the Floppy

=====

A brief over view of how they work

by Richard Bateman

We are all familiar with the cassette recorder attached to our Nascom, it provides us with low cost mass storage, and permanent storage for programs and data. The method of recording is governed by the type of signal the cassette can record. Although stereo recorders allow two channels to be recorded, the Nascom in fact is designed to record on only one channel, that's because it's easier and better to do it that way. In order to store our information on to one channel, we have to convert it to a serial stream of bits that can be recorded. Now this is the same process used for years with teletypes, and why the UART was invented. This device converts the input byte into a stream of 10 or 11 bits which consists of a start bit to tell the device to get ready, then the 8 bits of the byte, followed by either 1 or 2 stop bits to say it's finished. When the tape is replayed, the reverse occurs.

This system is known as asynchronous transmission, as each byte is sent independantly of the others. When NAS-SYS records a block using the "W" command and reads it back again with the "R" command, it knows there are 256 bytes in the block but the electronics handles each byte quite seperately. As far as the electronics is concerned, there is no difference between the typed input and the recorded input.

Disks are not quite the same. The recording media is the same type of material but arranged in a disk. The disk has only one surface for recording purposes. (I know they are double sided, but so is paper. You only write on one side at a time.) The recording head of the disk can be moved radially on the disk surface, but does not move while reading or writing. This allows a single head to be moved to sit over different "tracks" of the disk. As the disk turns, the same piece of disk keeps coming round and can be read or written to each time it passes. The floppy turns about 5 times per second and can record one change of flux (one bit) in 8 micro seconds. Thus a single turn would allow 1000000/8/8/5 bytes. Or 3,125 bytes per track.

Formatting. To get the most out of this space, a different system of coding is used, synchronous data transmission. This way we don't send a start bit and a stop bit, and so only send eight bits/byte. We have of course to pay a penalty for this, we need to know when the data started, so as to know which is the first bit. Because we can move the head about, we also need to know where it is, and limit the number of places where it can be. This gives a fixed number of tracks on the disk. On 5 inch floppies there are 35 or 40 or 77 tracks depending on who makes the drive. This gives a maximum drive capacity of 3,125*35 (*2 for systems with two heads capable of writing to both sides of the disk) bytes. That's a lot of storage, nominally 218,750 bytes for a double sided 35 track system. In order to access this data and to check for errors, we need to format it. This involves writing data onto the disk which does not change, but helps us find where we are.

Keep tracking. We divide the track up into sectors, each of which holds a nice round (HEXadecimal) number of bytes of data, 128, 256, or perhaps 512 bytes per sector. A few systems even use as much as 1K per sector. However, to know where we are, we must also include a number of bytes that will allow error checking and location. The most reliable method of error checking is known as 'CRC' or cyclic redundancy checking. This is simple and very fast to do in hardware but takes time in software (so CRC checks and generation is usually performed as a function of the disk controller chip). We also have the address for the sector, that is the track and sector header at the start of each sector with its own CRC, plus some spare bytes to allow time to switch from reading to writing, so that we do not have to rewrite the address data each time.

This information must also include the synchronization bytes along with each track/sector header. If we want to record in small amounts, we can divide the tracks up into small sectors, each one say 128 bytes long. After subtracting all the bytes used (in the above example) we are left with only 161,280 bytes on our disk. With a format that divides it up into 18 sectors per track, and 35 tracks per side with two sides.

FM and MFM Recording. Each byte is recorded as a series of pulses, between each is a clock pulse so that we do not lose synchronization. This system of recording is called FM and bears a strong resemblance to the CUTS system when run at 2400 BAUD, where only a single cycle of 2400Hz carrier is used for a 0 or a 1. Now MFM is a slightly different way of recording. It is possible to leave out the clock pulses, since they do not carry information, and still recover the data. This removal does not work if all zeroes are sent, so the clock is not removed if a zero was transmitted last time and will also be transmitted next time. Thus there are 2t, 3t, or 4t periods between pulses where 2t is the clock rate. For FM recording there are only 1t or 2t periods. Given MFM recording at the same data rate, only about half the information is recorded, as the majority of clock pulses are skipped, thus halving the recording density. If we double the data rate, there will only be the same number of pulses recorded on the disk as with FM recording. So by using MFM recording techniques the data density can be doubled and the drive (or the media) will not notice.

The MFM double density system of recording is more fragile as it has not got so many transitions per byte, but is widely used so must be reliable. It is necessary to restore the missing clock pulses, so a data separator circuit is employed and write precompensation to improve the chances of recovering the data correctly.

Addressing the disk. The disk is addressed by first positioning the head on the right track, a task performed by the controller chip, and then reading a sector header. The controller reports that the correct track has (or has not) been found and then, if the track/sector header was as required, proceeds to read or write the sector as instructed. When writing, the CPU sends the data to the controller chip byte by byte, the controller converting it to serial data and providing the necessary clock pulses, then feeding it to the drive. When reading, the controller chip receives the data pulses from the drive, re-inserts the clock pulses, sorts it out, and feeds each bit into a shift register. When a byte is complete, a 'ready' signal is sent, and the CPU comes and grabs the byte.

Controlling the controller chip is a piece of software known as the 'disk primitives'. This software is concerned with the mundane business of reading and writing a sector. It requires to know which drive is required, whether you wish to read or write, which track/sector is to be read or written to, where the data is to go to/come from; and other things such as reporting errors, keeping the drive motors running etc. In all this piece of software is usually about 1K long, very involved, and a perfect example of using a CPU to control and supervise the goings on of an external device. An interesting exercise is to try to understand the 'disk primitive' source listing supplied with the Henelec FDC kit.

Of course, the 'disk primitives' in turn require to be told what to do, and this is the function of the 'Disk Operating System' or DOS. A DOS may be as simple or as elaborate as you wish. The simplest in common use for the Nascom is D-DOS, which is nothing more than (nor does it pretend to be anything more than) a number of keyboard routines which accept instructions as to how many sectors are required, which track/sector is the start, whether it is a read or write command, and where the data is to go to/come from. No

attempt is made to keep track of where data was stored. It being up to the user to keep a record of that. D-DOS is rather like using a cassette tape recorder and keeping notes as to where data went by noting the numbers on the tape counter. Make a mistake, and it will lose something. Its advantage is that it is easy to understand, easy to modify for your own purposes if you are thinking of writing your own DOS, and, of course, being very simple, it's incredibly fast. The next level of cleverness comes with the new NAS-DOS, DCS-DOS, and the new Polydos. These work like D-DOS internally, but instead of you keeping notes on paper, the notes are recorded on the disk in a reserved area known as the directory. The only input to the routines required are whether a read or write is required, the name you chose for the program, and in the case of some writes, the length of the program. The rest is automatic. The DOS looks at the directory for the name required, and the information about it is recorded after it.

CP/M and the Floppy. Much has been written about CP/M in this newsletter of late, but no-one has explained what it is. CP/M stands for Control Program for Minicomputers. When that name was coined, minicomputers were pretty dumb animals by today's standards, and micros are now more powerful than minis then, so for minicomputers, read microcomputers. It was originally written to take the chore of handling DOS away, and make the disk system as transparent as possible. This end is achieved quite well, although CP/M isn't the friendliest piece of software around. Anyway, CP/M is an operating system much as Nassys is, but revolves around disks, particularly floppy ones. CP/M is divided into 3 parts CBIOS CCP and FDOS. The CBIOS does all the I/O and is written for each system, as disk I/O and keyboards differ. Now NAS-SYS and CBIOS are much the same, and indeed could be the same if the entry points were looked at closely.

We all know how NAS-SYS calls routines, so how does CP/M do it?

CP/M has a number of routines, 36 for revision 2.0, which are called by passing a routine number in C and an address in DE. Results are returned in A with double bytes in HL.

As CP/M only uses RST OH, and for the same as NAS-SYS, it should be possible to simply move NAS-SYS up to F000H to F800H and the video to FC00H to FFFFH and incorporate the CBIOS, plus initialise the RST 1 to 7H addresses to jump into NAS-SYS. The only problem then is to modify the VRAM, or rather calls to it. ZEAP is the biggest offender here, mainly because I don't have the source.

The NAS-SYS routines require about 2K, and the disk routines require 1K, NAS-SYS workspace is 128 bytes, so it's a bit of a push, but no doubt it can be done. The work space could be put somewhere else. That way it would provide the best of both worlds.

CP/M reads the Floppy

CP/M reads the disk by using a directory, that relates the logical name (that's what you call the file) and its physical position on the disk. CP/M does not use sector and track as in DDOS but uses block numbers, each block may be, say, 8 sectors, and may therefore be 1024 bytes. The directory entry has 16 bytes set aside to keep the block numbers for that file. If the block size is 1K, then each file can have up to 16K bytes per directory entry, this is known as an extent. Files longer than 16K require two or more consecutive directory entries, the first telling CP/M that it extends to the next extent, etc.

Each directory entry is 32 bytes long. Sixteen bytes for the block allocation map (telling CP/M where it put the blocks), and sixteen bytes used to store the file name, the extent record and the actual number of sectors used for that entry. Most files will have only one extent, but large programmes like MBASIC take more than 1. (MBASIC is 24k three times the Nascom BASIC).

BIT mapping

CP/M knows for each file which sectors (or strictly each block) that has been used, but needs to know how many and which blocks or sectors are free. This it does with a bit map. It uses a sector of 128 bytes to store the sectors used as 1 bit in each byte. It can then know exactly which and how many blocks are used up. A second advantage is that if it wants to set aside a portion of disk not to be used it only needs to set the bit map and the system will not use them. Clever, eh.

CP/M Commands

There are a number of commands within CP/M, such as DIR (directory), ERA (erase), etc. Note that these are single word commands. If CP/M was given a single word command, but could not find the name, it assumes it is the name of a file ending with the 'file type' .COM and loads that in and executes it, so you can add your own commands. Better still!

For example if you rename BASIC as RUN.COM and you type RUN STARTREK, CP/M will load BASIC and then load the basic program STARTREK. It is almost friendly.!

Well so much for this week/month/quarter/year.

SPEECH

Review of the Arfon Speech Synthesis Board

by Steven Hanselman

=====

The board comes well packed in a cardboard box. The board itself is contained in an anti-static treated bag. The board is Nasbus compatible although at 8" x 4" it is half the size of most Nasbus boards. Input to the board is through port F6 (hex) that's 246 (decimal). Output from the board is audio through a small but adequate speaker attached to the pcb, or a 3.5mm jack socket to an external speaker or amplifier. A status bit is returned via port F6 to tell you when the board is ready to accept another word. The speech synthesis part contains 144 pre-programmed words contained in two 64k bit ROMs. These ROMs are the same type of ROMs as the Nascom Basic ROM, so if you plug them into the Basic socket of the Nascom, you could tabulate them and probably work out how the speech is stored. Having done that you could purchase some more 64k roms (Maplins sell them) and find some way of programming your own speech set. (Doubtful but interesting premise. Ed.)

The board is exceptionally easy to program in Basic, all that is required is that you use the WAIT statement to monitor the busy line until free and when free output the value of the next word that you want the board to say. Although the ROMs only have a 144 word capacity, it is possible to make up almost any word that you want by truncating parts of words and using truncated parts to make the new word. This is not ideal, as, unfortunately, the way the Digitalker produces such good quality speech is by adding emphasis to certain characteristics within speech, such as raising the tone near the end of the word, and because of this when you put pieces of words together you sometimes find the words sounding slightly wrong. Apart from this the Digitalker board seems to be good value for money and great fun to use. The documentation is splendid as it reveals everything that you require to know about the board, how to use it, and how it works.

EDITOR'S MISC. NOTES

The Video Map published on page 52 of INMC80-4 was sent in by Mr W.R. Smith. Sorry we didn't mention it at the time.

Would any Nascom fans in ABU DHABI care to phone 820129 for exchange of ideas, programs etc? If so Ian Slater would like to hear from you.

Apologies for the length of time it sometimes takes to get articles that you submit into print. There are a variety of reasons for this, so if you sent something in some time ago, and haven't seen it yet, don't despair.

When is an Ad. not an Ad.? If you have some equipment to sell that you no longer use, then a small Ad. of 2 or 3 lines is free. If it is something that a private individual is selling for profit, e.g. copies of one of your programs, then there is a nominal charge of 5.00 for 7 or 8 lines. If you are a commercial organisation then it's 100.00 per page, or proportional charge.

Finally, if there is anyone out there who would like to be of hardware &/or software assistance to fellow computer users, drop us a line and when we have a couple of dozen names and addresses we'll print them.

EPROM PROG.

EPROM PROGRAMMER REVIEW

by Paul Wilson

I have just purchased a "Gemini EPROM Programmer" from Bit's & P.C.'s and thought you might like to know how I rate the kit.

Circuit board is superb quality and all hardware is included with the exception of connector plug(s) and a case. I.C's are socketted and two Zero insertion force sockets are provided for EPROM. Why two? Well, the blank EPROM may be programmed from RAM data, or from a Master EPROM, providing quick copying facilities.

Documentation at first appears sketchy, but is adequate, construction is quite straight forward (provided that one is familiar with Tantalum capacitor polarities!) An ON/OFF switch is provided, also a 2708/2716 Selector switch. The omission of a Power-on LED indicator is, I think, a disadvantage, but is easily added. Connections to my Nascom 1 were straight forward, requiring DIL header plugs and ribbon cable plus 12V & -5V supplies. Testing is limited to checking the 25 volt supply generated on the board, and other chip supplies.

Software supplied comprised a tape (in N2 format, so useless for me!) and an assembled listing with operating instructions. The program resides in memory from 1000 up, (less than 1K, so ROM'able!) and provides; checking for erased EPROM, verifying a Blown EPROM, and Blowing an EPROM from a Master or from an area of Memory. Since the Programmer interfaces to the PIO and this was to be the first time I had used this device (blush!) I was apprehensive - however, my first blank Prom was plugged in, software loaded and the unit switched on - the first check was to run the program to check for an erased PROM - all OK, so Blow EPROM from memory contents specified - the screen indicates Blowing and cycle No., after cycle zero an automatic verify occurs; all OK! Then disaster - There are 2 latch switches, push on and push off and I pushed the EPROM select switch instead of the power switch. This put 25 volts on my 2708 where it did not like it, and I suspect, fed back to my Nascom which filled the screen with garbage and would not reset. A power-down (and prayers) in seconds saved my Nascom and programmer, and even the 2708 checked OK, but runs too slow to use on the system (I have to copy it to RAM to use the programs on it). Subsequent EPROMs have copied perfectly and function OK in my system - I have fitted a blank plate over the EPROM select switch to prevent "accidental" operation (the switches are identical and side by side).

To summarise then the kit was easy to build, easy to connect and is easy to operate and good value for money - nuff said? As a confirmed idiot I can honestly say it is not quite idiot-proof. (By the way why are EPROM Erasers so expensive?)

Z80 Guide

THE KIDDIES GUIDE TO Z80 ASSEMBLER PROGRAMMING

D. R. Hunt

=====

The Crossroads of personal computing

Part: The Fifth

Getting stuck in

Please note, book your place at the funny farm now.

To date, these little insights into the Z80 processor, and my somewhat sordid little life in general, have been liberally scattered with some of the most awful puns and cracks imaginable. Well, if you think I intend to sober up and do this job properly, I'm afraid you will be very disappointed. You see, as the title of this piece suggests, by owning a home computer you are already a suitable candidate for the local nut house. You don't think so? You go and ask the wife, or any of your acquaintances who knew you in quieter and saner days!!!

Of course, this is nothing really new. It's a strange fact of life that electronics, (and in the sense I'm talking about, home computing) has always had some doubtful mystique about it. Now everyone has a hobby of some sort, whether it be getting your enjoyment on the allotment, or the keeping and breeding of pet boa constrictors. It's a further fact that the hobbies which are the joy of the perpetrators bear little or no relation to way in which those people earn their bread and butter. You can have people like main line train drivers clocking off and rushing home to an evenings basket weaving, or, like my old man, the bursar of a hospital, doing his thing by building pipe organs in his spare time. All this is the normally expected thing. Not only are people willing to discuss their variant hobbies, but others are equally interested in listening. Talk to an accountant about cars, and you will probably find he is well into things like 'dwell angle' and the merits (or otherwise) of upper cylinder lubricants. Yet, mention you are into electronics or computing, and the other will be struck dumb and regard you with the suspicion normally only reserved for tax inspectors. If this is not enough to give you a complex, then the lack of comprehension and instant boredom of the other is guaranteed to make you think that you are some sort of social pariah and that you are in need of treatment. At the very least it makes you rapidly resort to talking about your job, pointing out that you are really the bloke who fits the doors in the local fridge factory, so you really must be sane and a member of the human race. That your affection for home computing is only a passing aberration, and that next week you'll probably be into collecting brass doorknobs or some other more normal leisure time relaxant.

When I started work as an electronics development engineer (trainee) at the grand salary of five pounds ten shillings a week (and that wasn't in the middle of the last century either, just in case anyone under 30 reads this), my lab boss, Alan, once said, "They say that by the time an electronics engineer is 25, he will be either totally sane, or totally, round the bend.". Now Alan was only 24 at the time, so he hadn't quite made up his mind. He did get married the following year, so I guess he succumbed in the end. Mind you, he was only preparing me for the cruel facts of life, which I would learn during my time of endless night classes, day release, and all that rot. (Not that it did me a lot of good anyway, I've forgotten 95% of it). Well I passed the dreaded '25 year break' about 10 years ago, and I know what happened to me. If you've managed to plough through the last four episodes of this never ending cesspit of scattered and disjointed thoughts, and not worked out which way I went, well, you just haven't been paying attention.

Anyway, to business, to date we've been through most of the important bits of learning to program in machine assembler. Yes believe it or not, you have. From now on in it's learning the tricks of the trade and figuring it out for yourself. Of course there are thousands of little tricks that can be learned, but most of these will either come with practice, or by pinching them from elsewhere. As far as pinching ideas from elsewhere goes, every Nascom owner has a complete library of software waiting to be lifted. Yup, you got it, NAS-SYS (or NASBUG). It's completely within the rules to use bits of the monitor as a guide for things that you might find difficult to figure out for yourself. That's why the complete listing has been provided, and in Richard's view (he wrote NASBUG T4 and both NAS-SYSii) imitation is the sincerest form of flattery (I know, 'cos I asked him). If anyone wanted to make a complete mystery of the monitor, they would not have published the listing, like some other machines I could mention. So, try to understand the monitor. The comments are a bit brief (where they exist), because the monitors were written with the aid of the ZEAP assembler, and at the time Richard only had 32K of RAM to work in. So it meant that there wasn't much space to waste on comments. I'm not suggesting you kick the moggie out of the comfy chair and sit down and digest the lot in one go. If you did manage it, I doubt that you'd remember much of what you read, and even less understand it. No, break it up into nice little chunks. Pick the shorter and easier routines first, and, if you'll take my advice, leave the KBD routine till last. If you start on the KBD routine, you WILL end up on the funny farm.

As my contribution to the exercise, I'm going to attempt to describe the routine known as B2HEX. I've chosen this one, partly because it's short, partly because for the most part it's easy, and mainly because the later version in NAS-SYS employs one of those programming tricks by way of example. The listing below is from NAS-SYS 3, but apart from its absolute address, it's identical to the NAS-SYS 1 version. Now remember from the last episode I mentioned the need to make the labels mean something, well B2HEX is the epitome of the cryptic label writer's art. Those of you who complete the TIMES crossword on the way to work will have got it already. For those not so quick on the uptake, B2HEX means 'Binary To HEX'. Neat isn't it?

Before we get bogged down in the innards of B2HEX, there are a few things that have to be explained. Firstly, lets get a few definitions out of the way. A nibble!? This is an expression that has crept into use with 4 bit processors. A 16 bit processor works with 16 bits WORDs, an 8 bit processor works with 8 bit BYTES, so some comedian decided that as a '4 bit unit' was half a BYTE, the definition of a ought to be a NIBBLE, and it's stuck. So we have:

16 bit WORDS
8 bit BYTES
4 bit NIBBLES

To date we haven't dealt in detail with the STACK POINTER register. Now the STACK (that's the area of memory the SP points to) is rather important. It's best thought of as a vertical column of pigeon holes, the top being the highest address assigned to it, the bottom being the lowest address it reaches in normal use. (In abnormal use, it could be anywhere and any depth, and that is a common cause of 'crashes' where the stack comes so far down that it overwrites the program beneath it.) Now there are many occasions where we need to tuck things away out of sight where they won't get corrupted by normal processor register manipulations. In other words, we want to save something till later, and then retrieve it in one piece. That's what the STACK is for. The STACK normally works in pairs of pigeon holes, as more often than not, it's two byte addresses we want to save. To make this process a bit more manageable, the Z80 looks after the STACK automatically, and saves everything else in pairs as well, so if, for instance we wanted to save the B register, we'd actually save the BC pair. A point to note when retrieving data from the STACK, as when B is restored, C is also restored to its previous value.

Now the STACK works like this. When a program is initialized, (that's the run up to the start of the program proper), it's normal to set the SP to the top address that the STACK will occupy. With the Nascom monitors, this is normally set to 1000H for you by the Execute command, but this may be inconvenient, and anyway it's good practice to initialize the SP yourself, as you may one day play with other computers which aren't so obliging, and which leave the SP any old place. So for the sake of argument, we initialize the SP to 1000H, the first time we use the STACK by executing a CALL or a PUSH instruction (there are some others that use the STACK, but these two are the most common), the SP is first decremented by one. So in this instance it will point to OFFFH. It then places the high byte of the two byte data in memory AT THE LOCATION POINTED TO BY THE SP. The SP is the automatically decremented again, and it places the low byte AT THE LOCATION POINTED TO BY THE SP. The SP is now pointing at address OFFEH. Next time something is put on the STACK, the first thing the SP will do is to decrement by one, to point at OFFDH. The reverse is true on a RET or POP instruction. The SP will already be at the point where the last byte was deposited in memory, so it will restore that data first without incrementing, then it will increment and restore the high byte, and last it will increment again to point at the low byte of the next data to be restored. See how the stack management is both automatic and orderly. The SP is always left pointing to the low byte of the previous data, or, looking the other way, one byte above the next data. When putting stuff on the STACK, the SP is decremented first, when getting stuff off the STACK, the SP is incremented last.

The STACK is an orderly 'LAST IN FIRST OUT' (LIFO) file, the last thing you put on the STACK is the first off the STACK. This normally works well, and under normal circumstances, you should never end up in the position where the data you wish to retrieve is not the next off the STACK. If you do have this sort of problem, then it is best to rethink your program, as manipulating the STACK with the INC SP and DEC SP commands is best left to experts. Unless of course, you are the masochistic type who goes looking for headaches.

One important instruction involved in this version of the B2HEX routine is the DAA. Now, the routine uses DAA because of it's properties, and not because it is the logical instruction to use. In other words, it's a trick. How this particular use of the DAA was discovered, I don't know. But it proves the case for knowing exactly HOW and instruction work as opposed to the superficial knowledge of what it does. This is what it does and how it does it. The DAA instruction is primarily intended for use with 'packed Binary Coded Decimal' arithmetic. That's where each nibble represents a separate decimal number, and when we add another packed BCD number to it, an adjustment has to be made after the addition in case the low nibble overflowed (in decimal) and the carry has to be subsequently added to the high nibble. Don't forget the addition would have been performed in binary, as the Z80 normally doesn't know anything about packed BCD arithmetic. But, one thing the Z80 does provide is what is known as the 'Half Carry flag' which indicates this very 'Half Carry' overflow condition during the various addition or subtraction commands. As far as I know, the DAA instruction is the only instruction to MAKE USE OF the 'Half Carry' flag. If the high nibble overflows, the DAA instruction causes the C flag to be set to indicate that there was an overflow.

Now, back to B2HEX. First, let's examine the need for this particular routine. A lot of the monitor commands require the contents of a register to be printed out on the screen. For starters, the Tab command uses it to display the contents of the memory locations, the Memory command uses it for the same purpose. Less directly, the Single Step, Arithmetic, Breakpoint, Print registers, Query ports, Read and Write routines all use it for the purpose of displaying HEX characters in either single or double byte form. So you see, this routine is used an awful lot. Now, why 'Binary to HEX', well, we discussed

in episode one the fact that the contents of the registers is in binary, and that binary was somewhat unmanagable for the normal mortal. It was demonstrated that HEX was a more useful format for displaying the binary information. So everything in the registers has to be converted from binary to HEX before the computer lets us have a look at it. In fact, this process becomes so transparent after a while, that it is all too easy to think of the registers containing only HEX characters, and not binary bits at all. This is a way of thought which should be avoided. Anyway, I think there is a conclusive case for the existence of routine B2HEX.

Now to its specification. Those Nascom owners with NAS-SYS already have the specification written out in the software manual, for the benefit of those not using NAS-SYS, and for the benefit of those NAS-SYS owners too lazy to look it up, it says:

Output the value in the A register in ASCII. The A register is modified.

Well, it's not quite as clear as it could be, but what it means is, 'put a binary character in the A register, call this routine, and the contents of the the A register will be sent to the screen as two ASCII characters expressing the value of the A register in HEX'. It also notes that the contents of the A register will be something different at the completion of the routine.

So the need for B2HEX is established, and what it does. How does it do it? Well, it's rewritten here, lavishly smothered in comments. As a pointer, there's nothing to stop you smothering your copy of the NAS-SYS listing with comments to make it more understandable. Before reading on, see if you can follow it.

ROUTINE B2HEX

=====

This routine part of the routine puts the top four bits (most significant nibble) of the byte to be printed into the bottom four bits (least significant nibble) by rotating the byte to the right four times. It then calls the second routine to convert the four lowest bits into an ASCII character and then to sends it to the screen.

```

                                4470 ; OUTPUT A
0375 F5                        4475 B2HEX      PUSH AF          ; Save the character for second half
0376 1F                        4480              RRA              ; Move the top four bits into the
0377 1F                        4485              RRA              ;   bottom four bits by rotating
0378 1F                        4490              RRA              ;   everything right four places
0379 1F                        4495              RRA
037A D701                      4500          RCAL BIHEX          ; Now call the conversion routine
037C F1                        4505          POP AF              ; Get the character back for the four
                                ;   low bits and drop through

```

This half of the routine first gets rid of the top four bits by ANDing it with 0FH, then it performs a bit of arithmetic on it to convert it into a printable ASCII character. It then calls the screen output routine to print it on the screen.

```

                                4515 ; OUTPUT LOW HALF A
037D E60F                      4520 BIHEX      AND 0FH          ; Mask out the high nibble
037F C690                      4525              ADD A,90H       ; Now convert to a HEX character
0381 27                        4530              DAA              ;   and further convert to ASCII.
0382 CE40                      4535              ADC A,40H
0384 27                        4540              DAA
                                4545 ; OUTPUT CHAR
0385 F7                        4550          RST ROUT            ; Send the character in A to screen
0386 C9                        4555          RET                  ; Return from the routine.

```

Got it, good, you can forget the rest of this article, and any subsequent ones, 'cos you're there already. For those who don't thoroughly understand it, here follows the detail blow by blow description.

When dealing with a subroutine in isolation, we have to make some assumptions, firstly that the data to be printed has already been put in the A register, and for the purposes of demonstration, this is going to be B5H. Next we assume the routine has been CALLED in an orderly fashion, implying that the STACK is already correctly set, and left pointing at the last byte saved (in this instance, because the routine was CALLED, pointing at the low byte of the return address (as the PC will have already been advanced to the correct return address by the CALL instruction before the return data was placed on the STACK, see episodes 3 and 4).

Ok. Oft we jolly well go!!

Line 4470. This is a brief comment as to what we propose to do, viz:
OUTPUT A.

Line 4475. This is the start of the routine, and is labelled B2HEX accordingly. The instruction is PUSH AF, which will put the current contents of registers A and F onto the STACK. The STACK will contain B5H as it's higher byte (because I've already said that A contains B5H), and XX as it's lower byte (because that state of the F register was indeterminate when we PUSHed the AF register, and we don't care a damn what it contains). The SP has been decremented by two. We PUSHed AF because the calculation process will destroy the current contents of the A register, and we need it back to work out the second half. As a matter of interest, we could have equally well saved the contents of A in some other register, which would have been slightly quicker in terms of execution time, but as the specification implies that all other registers except A remain unchanged, we would have had to save the contents of the other register anyway, so what odds!!

Lines 4480 to 4495. This is a fun one! If you think about it, an 8 bit byte expressed in HEX contains two alpha numeric characters, and each character can be said to represent a nibble of the 8 bit byte. Now the order they appear on the screen says that the highest nibble will be printed first. So because of the way the routine works (wait for it), we've got to move the high nibble into the place currently occupied by the low nibble. This is done by moving everything four places to the right (no political cracks please). More precisely, four 'Rotate Right Arithmetic' (RRA) instructions are used (shift instructions would have worked just as well, but someone in pre-history dictated that rotate instructions would be used). The rotate right moves the contents of the C flag into the highest bit, everything else right one place, and the bit that drops off the end into the C flag. Lets follow that diagrammatically:

B5H equals	10110101	X	in binary bits, the X is the C flag.
			Now rotate it one
	X1011010	1	the C flag, X has been moved into the highest bit
			everything else has moved right one place, and
			the lowest bit has moved into the C flag.
			Rotate again
	1X101101	0	Rotate again
	01X10110	1	and for the last time
	101X1011	0	

Simple really, and the end result is that the 'B' of B5H has been moved into the low nibble.

Line 4500. Now to call the calculation routine using the NAS-SYS RCAL routine. Don't worry how that works, just pretend it's a CALL as per the Z80 manual. The CALL instruction PUSHes the return address (037C) onto the STACK, and loads the PC with the address CALLED, 037D, so we skip along to line 4515.

Line 4515 tells us we are about to output the low nibble of A.

Line 4520. Now the fun really starts. The rotate instructions have left a lot of indeterminate garbage in the high nibble. Indeterminate, because we niether knew nor cared what the state of the C flag was before the start of the rotates. We want to get rid of the high nibble, so we use a logical AND instruction. We AND the X5H with 0FH, which like magic gives us 05H. Ok, so how is it done? Well I don't intend to go into a discourse on Boolean algebra here, save to say that in the middle of the last century a fella called Boole (french, I think), proposed a 'logical' method of performing algebraic functions which caused a few sniggers at the time. He suggested that there shouldn't only be the normal arithmetic operators like, plus, minus, multiply and divide, but there should also be what he call logical operators like AND, NOT and OR, etc. No-one payed much attention to him and what he had to say didn't really become relevant until way after his death. I often wonder what would have happened if Boole and Babbage had got together, may be we would have had vast steam driven mechanical computing engines to rival Asimov's Multivac, and this in the middle of the last century. The nett result of ANDing a logical 1 with a logical 1 is 1 (and no carry), and of ANDing logical X (either 1 or 0) with 0 is 0.

```

101X1011
          AND
00001111
-----
00001011

```

See!! If you don't, then get Boole's book out of the library, it's still in print, and his methods will get more important as you progress in this business. Clever, what? We've got rid of the bits we didn't want, and are left with those we want. The process has a name, it's called masking. The 0FH is the mask, and it allows through what is required, whilst holding back that which is not.

Line 4525. We've now got to convert the four bits in a into an ASCII character which will represent the four bits in HEX. And that's not so easy. We start off by filling the high nibble with the binary pattern for 9, 1001, and that is achieved by ADDing 90H to A. The ADD instruction is used, as the C flag is still indeterminate, and we don't want any spurious carries added to our low nibble. We have used the ADD A,90H instruction, because we are going to use a Decimal Add Adjust (DAA) instruction next. By having 9 as the high nibble, it is our intention to make the high nibble overflow if there is an overflow from the low nibble. The ADD instruction also set the 'Half Carry' flag as the A register contents are now 9BH, the overflow being to warn that the low nibble contains a number greater than (decimal) 9, in case there is a following DAA instruction. Please think carefully, I'm doing my best.

Line 4530. We have 9BH in the A register, and we perform a Decimal Add Adjust on it. Now the low nibble was B, which the DAA instruction takes as 11 (decimal), so it puts 1 in the low nibble. As the low nibble was 11, the previous ADD instruction had already set the 'Half Carry' flag, so there is a carry through to the high nibble, making it 10 (decimal), so it puts 0 in the high nibble, and sets the Carry flag to show there was an overflow. The A register now contains 01H, and the C flag has been set to indicate an overflow.

Line 4535. We're half way there, all that's needed now is to add the ASCII offset to the number in the A register, take account of any carries, and we're all but home. Now, the ASCII code for 0 (nought) is 30H, 1 is 31H, upto 39H for 9. Fortunately for our purposes, the ASCII code for 'A' is 41H, though to 46H for 'F'. Let's add (with carry) 40H to the A register.

```

00000001 1
01000000
-----
01000010      which is 42H

```

And 42H is the letter 'B' in the ASCII code. Weren't we lucky that the ASCII code works so neatly in our favour. Well not quite, because we cheated and added 40H, which is hardly what you would expect as an ASCII offset. 30H would be more likely. Still, lets plod on, and see what's next.

Line 4540. Another DAA? Well what purpose could that serve. In this instance, none, there are no carries from the low nibble, and none from the high nibble, so our 42H remains 42H.

Line 4545. Just another comment, telling us were about to print the character.

Line 4550. The idea is to send the character in A to the screen where it will be printed. Of course, it would be simple to CALL the CRT routine, but this would seriously reduce the usefulness of the monitor. In NAS-SYS, use is made of a generalized output routine called ROUT, ROUT takes the character and directs where ever the various output options dictate, this would normally be the screen, but not always. There are some very clever ways in which NAS-SYS or the user can trap data sent through ROUT and reprocess it. Perhaps we'll discuss some of the cleverer bits of NAS-SYS next time. Anyway, the CALL will PUSH the return address on the STACK, go away and do its thing, and return to line 4555.

Line 4555. This is a simple return. The return address is already on the STACK, and (unless some other routine screwed it up, and it won't get it wrong without outside help) the SP will be pointing to it. The address is retrieved from the STACK and placed in the PC register, decrementing the SP as it goes. The routine makes an orderly return to line 4505, with the SP pointing to the data placed on the STACK by the PUSH in line 4475.

Line 4505. The POP AF instruction puts the original contents of the A register back in place, and in so doing, decrements the SP to point at the return address of the routine which originally CALLED B2HEX. Now this is one of the sneakier tricks of writing machine code programs. Notice, previously, we CALLED BIHEX (a meaningless label, except to indicate 'send one character of B2HEX'), this time we don't CALL it, all the relevant information is in the right place, namely, the low nibble of A now contains the character to be converted and sent next. By simply 'dropping through', we have avoided another call, and it's associated return, and made for a nice tidy job.

Line 4520. Mask out the high nibble to leave 05H in the A register.

Line 4525. ADD 90H, so that any overflow in the low nibble will be transferred to the C flag after the following DAA instruction.

Line 4530. DAA. In this instance there is no overflow, and the contents of the A register remain unchanged, and the C flag is not set.

Line 4535. ADD with Carry, the ASCII offset, which previously we had 'fiddled' to make the earlier answer right. This time our 95H in the A register has become D5H after having 40H added to it. As there was no carry from the previous DAA, the low nibble, 5, remains unchanged. D5H is a suspicious number, as, if it were treated as decimal, there would be an overflow, and a correction in the high nibble, which makes it a likely candidate for the following DAA instruction, which last time round seemed to serve no purpose.

Line 4540. DAA. Guess what? The previous ADC has not affected the low nibble, so there is no 'Half Carry' through to the high nibble. But the high nibble, D, is 13 (decimal), so the high nibble is replaced with 3, and the C flag is set. So by some sort of jiggery-pokery, the byte in A has become 35H,

which is, of course the ASCII code for 5. Now it wasn't really magic, it was knowing the effects of the DAA instruction which is the heart of this whole routine. I confess, that given the original specification, my approach would have been a lot more long winded, and I don't suppose for one minute that I would have thought of using the DAA instruction in this way. The ADC A,40H can be thought of as a 'fiddle factor', but the result is what was required. The whole piece of code from line 4525 to 4540 must be thought of as a single piece of bit manipulation, and can only be arrived at by very careful thought and thorough knowledge of the effects of the instructions.

Line 4550. Go and print the character on the screen.

Line 4555. Get the return address off the STACK and go back to where you came from.

And here endeth the lesson. What's been learned? Well, firstly and most important, the fact that instructions may be intended for one purpose, but that does not impose a constraint on their imaginative use for other purposes. The two crucial instructions are the DAA and the ADC A,40H. Knowing exactly the effect of the DAA allows the programmer to deduce the required 'fiddle factor' to make the routine work correctly. Another thing has been the correct and orderly use of the STACK. See how neatly the whole thing comes together. The SP is always in the right place when data is wanted, and simple PUSH, POP, (not so simple) CALL and RET instructions always retrieved the correct data at the right time. The way the routine CALLED the second half the first time round, but simply dropped through the second time, thus saving one CALL instruction, and one additional RET instruction, so saving four bytes. (Not that space is usually at a premium, but there is an awful lot crammed into the 2K of NAS-SYS). I hope you all understand how the B2HEX routine works now, and hopefully, my commenting of the routine made for easier understanding.

Try commenting some of the other routines in the monitor. If you can't see how they work from the listing, there's nothing to stop you writing a simple piece of code which will pre-fill the registers with the right data and then CALLing the routine in question via a single step command. It's quite revealing when you do it that way, you see all the registers change as the routine steps through, and you can see the sometimes hidden effects of the flags as it goes along. If you have NAS-DIS and D-BUG, so much the better, the display is so much clearer. Personally I find the register display of NAS-SYS 3 a retrograde step, as, although it gives much more information, it's confusing to read.

Now for the ultimate confession. When I started to write this piece, I didn't know how B2HEX worked either. Of course I'd looked at, but the significance of the DAA instructions hadn't caught my eye. I had just taken it as a 'black box' module and not given it any attention. Imagine my surprise when I actually sat down and single stepped through it (I've got to do that, 'cos if I get it wrong, think of the delight you'd get writing me rude letters) and then tried to figure out exactly how it performed the magic conversion. I phoned Richard to confirm I'd got it right, and to ask how he'd invented that particular way of doing it. It was then I discovered that Richard wasn't the perpetrator of the particular piece of code. In writing NAS-SYS 1, Richard had run out of space by about 20 - 30 bytes, so he spoke to David Parkinson (of REVAS, NAS-DIS and GEMINI disk system fame) and asked if he could think of any ways of losing a few bytes. Dave came up with this one, and a couple in the KBD routine which saved the day. Incidentally, I now know why I've never been able to fully understand how the keyboard routine works. David Parkinson has been at it!!!

Still, give David his due, he's far cleverer at this programming lark than I will ever be. My excuse? I gave that in episode one, when I said that I started off knowing nothing, and I don't suppose I learned much since. I'll remember B2HEX though!!

BLS PASCAL

BLUE LABEL SOFTWARE PASCAL

a review by Rory O'Farrell

=====

This is one of the most interesting Pascals which has recently become available for the Nascom. It is available on tape (1200 Baud, to run under Nas-Sys) living at 1000H, or in EPROM, 6x2716s living at 0D000H. Distribution is through Electrovalue Ltd., and I understand that it will be available through other Nascom dealers. I only have approximate costs. These are, for the tape 50.00, and for the EPROMS 90.00.

The tape is recorded at 1200 baud CUTS, and loaded first attempt, with no errors, as it should. (As the interface on the N2 is the Cottis Blandford, there is no reason why you shouldn't be able to record and read reliably at 1200 baud. If not, try adjusting VR1 again!)

The tape loads from 1000H to 4000H. The package is entered by E2180 XXXX, where XXXX is the highest address to be used. If the XXXX is omitted, then all available RAM is used. (If you have a 64k system, watch out! The address it will set as the highest is 0000, and you will get memory full messages! Re-enter the package using FFFF as the limit address). You get a commercial, and then a '>' prompt for a command.

The operating system recognises 11 commands, which can be divided into four groups:

1. Loading and saving text to/from tape
2. The Editor
3. The compiler
4. Miscellaneous commands

It is probably best to deal with these in order. The Load and Save allow a named program to be loaded/saved. There is also a Verify command, to check for a good save.

The Editor is a very remarkable piece of work. It allows 80 character lines, and the VDU appears as a window on these lines. This window can be moved up and down or right and left. If you go off on the right, suddenly you come around on the text on the left. There are 27 commands in the Editor, which are obtained either by the dedicated keys for cursor movement, or by depressing the cursor around, insertion/deletion of block move characters, movement of marked blocks, tabulation, searching for named strings, and return to main command loop of the compiler. There is one other one which the world has been waiting for. This is CTRL/G, which turns the Graphics key into an ALPHA LOCK key. Each time the Graphics key is pressed when this command is in use, the keyboard changes from lower to upper case, or upper to lower, as appropriate. It is also possible to allow the Graphics key to behave as normal.

The best way to describe the editor is to say that it is not unlike the NASPEN editor. There are no line numbers, and you can move about the text, inserting and deleting as the fancy takes you. It takes a little bit of getting used to, but as it allows 80 char. lines, it is a much nicer editor for programming than a line oriented editor which will only allow 48 chars.

Having entered the source program, one exits the Editor by a CTRL X, and returns to the command mode of the operating system. After saving the program on tape, using the named Save facility, one can proceed to try a Compile. If in compiling any error is found, compilation is halted, and you are returned to the Editor, with the screen set to the line in error. You are prompted by a message on the top line of the screen to "Hit Space", and this done, you lapse into the edit mode, to make whatever corrections you think necessary.

A list of 45 error numbers is given in the Manual, with a fairly helpful explanation of what is wrong. The restriction of available memory space prevents the compiler translating the error number into the message on screen. If, during compilation, the buffer overflows, then compilation is aborted, and an Overflow message is printed. In this case, the MTOP pointer should be moved, or compilation to TAPE should be selected. This latter option is a very nice one, as it allows programs of very large size to be compiled onto tape, which can overlay the text buffer and even the compiler itself, provided the runtime support package, which runs from 1000H to 2180H is maintained. The normal position of the compiler is from 1000H to 4000H, with the text buffer following on immediately. For the tape version, a minimum 32k system is recommended, and for the EPROM version, a 16k system will suffice.

The Run command will jump to the compiled program and run it, or, if no compilation has taken place, will force one, and then execute the program. If runtime errors occur, then a message to that effect is printed, and execution is aborted, with a return to the operating system or to Nas-Sys. With such a runtime error, you are given an address which indicates the location of the error relative to the start of the compiled code. Knowing how difficult it is to locate such a position, the authors of this compiler have given the operating system a new facility, called Find.

If you reenter the operating system, and tell it to Find nnnn, then it will locate the statement which compiled to live at that address. You may then make whatever alteration to the code you wish, and try another compile.

The miscellaneous commands are simple: Memory, which prints the extent of the text buffer, and if present, the extent of the Code, ZAP, which clears the buffer and Quit to return to the monitor. All commands may be abbreviated to their first letter only, with the exception of ZAP, which must be spelled out in full to avoid accidental erasure.

So much for the interface between the operating system and the user. How about the Pascal? This Pascal offers the following:

Statements: BEGIN END
 FOR TO/DOWNT0 DO
 REPEAT UNTIL
 GOTO (bringing joy to Dr Dark's disordered existence)
 IF THEN ELSE
 WHILE DO
 CASE OF OTHERS
 INIT TO (used to initialise data)
 := (assignment)
 PROCEDURE statement

Data Types: REAL 11.5 digit arithmetic !!!
 INTEGER - 32768 to 32767
 STRINGS up to 255 chars of 255 different values
 BOOLEAN logical variables
 ARRAY .. OF with multiple dimensions

Operators: +-*/ DIV MOD SHIFT AND OR EXOR = <> > < >= <=

Procedures: WRITE WRITELN READ READLN LOAD SAVE CALL SCREEN PLOT

Functions: ABS SQR SQRT SIN COS ARCTAN LN EXP INT FRAC SUCC PRED ODD TRUNC
 ROUND ORD CHR LENGTH MID LEFT RIGHT CONCAT ADDR RANDOM POINT
 KEYBOARD

Declarations: LABEL CONST VAR PROCEDURE FUNCTION

Others: User written machine code subroutines are supported, declared as EXTERNAL. Memory is treated as an array, and can be accessed readily, both for reading and writing. Hex constants are supported, as are capital and lower case letters.

In this Pascal, the data type Character has been replaced with a STRING type, where a string is up to 255 characters long. Predefined functions allow for the manipulation of these strings in much the same way as we have learned in BASIC. (Go and wash your mouth out with soap!) Predefined PLOT and POINT allow easy interface with the Nascom Graphics.

As you can see from the preceding list, a fairly full subset of Pascal is supported. The omission of Sets, user-definable types (including records) and file types is to be regretted, but can be programmed round with a little ingenuity. Figures are supplied of the PCW benchmark times for this package. Running at 4MHz, with one waitstate, it is faster than an APPLE 2 and a Heathkit H11A, which uses an LSI 11/2 16 bit processor! How much faster, you ask? In round figures, it is three times faster than the APPLE, and in general, twice as fast as the Heathkit. It loses out to the Heathkit on the Real algebra and Maths tests, where the LSI 11 beats it, but is still comfortably faster on all other tests, particularly on the housekeeping overheads. In addition, it offers 11+ significant digits, whilst the APPLE is only offering 6+ !

The arithmetic accuracy is remarkable - the strings of figures marching across the screen will impress even the uninitiated, but when you hear that it has predefined the ability to format figures to so many decimal places, in fields of a predefined width, you will realise that this package is a must for number crunchers!

It is to be hoped that in the future the authors will expand this package to include set operations, and user defined types - perhaps even variant records? In spite of the omission of these, this package is quite capable of doing anything an extended BASIC can do, and doing it considerably quicker.

There will be points I have not covered in this review, due to limitations of space and/or time (I'm sure you'd rather have the review in this INMC80 rather than the next one) but your Nascom dealer will be able to help you. Any errors and omissions are regretted, but neither I nor the INMC80 committee, or anyone else are responsible. (Responsible for what? - Ed.)

48K RAM B board with full complement of memory. New Link block.
100.00 or nearest offer.
Tel Stoke-on-Trent (0782) 324639 evenings.

Nascom 2 with 32K, Kenilworth case, Toolkit, Naspen, Nasdis, Zeap, Pascal, Nascount, Games, Etc. Total cost over 750.00 will sell for 500.00.
Full documentation including INMC mags. available.
Also Nascom Imp, 200.00. Genuine reason for sale.
Phone Great Dunmow (0371) 3119.

SYS

DISK SYSTEMS - WHAT IS THIS THING CALLED SYS??

by R. Beal

=====

Old-time INMC readers may remember some articles which I wrote about the PIO and the "Nascom One-Two", which were a result of trying to interface a Nascom 1 to my Nascom 2 to act as a "Giant Intelligent Print Buffer". I am pleased to report that my trusty Nascom 1 (one of the first) sits under my printer waiting for interrupts, and always faithfully responds.

My system seemed complete - until the age of the floppy disk. Unable to resist the compulsion, I soon had a disk system. I never imagined how much I would learn about microcomputing by that move. My first impression of CP/M was that it was crude and unfriendly, but now I have known it for some time I can forgive its faults and enjoy the benefits of the superb software which is available to run under it, such as the Microsoft languages (Basic, Fortran, Cobol) and most importantly the Z80 assembler/linker, Macro 80. My Nascom now provides quite a powerful software development system, even though I use its abilities only to develop it further.

When I first ran CP/M, an immediate problem arose - I had lost my printer interface software, and there was no obvious way to get it back! The official method is to patch the CP/M image generated by MOVCPM. To be fair, that is quite possible, with patience, but is a poor solution. Also, each different size system would have to be patched separately. The even more official solution, which is restricted to CP/M distributors, is to patch the BIOS module which is part of MOVCPM. I have since learned how to perform this procedure, and it is quite horrible. Even if I had had the official CP/M redistribution disk I wouldn't have wanted to use that method.

It was then that after much thought SYS was invented. SYS is a special CP/M program which consists of a new BIOS, together with a program which relocates this BIOS automatically for any size CP/M system, and slots it into place where the original BIOS used to be. The BIOS is the part of CP/M which contains all the input and output routines. All that remained was to get SYS to run automatically when CP/M was started up. Originally I solved that problem by using a special boot ROM which pretended to be a keyboard with a person sitting at it typing in "SYS". In more recent versions of SYS a better method has been used, which requires only a tiny patch to MOVCPM. This runs SYS on a cold boot, but not on a warm boot, which would be pointless. SYS can now be configured in many ways. For example there are four configurations for the Gemini/Henelec single density disk card. These are for CP/M 1.4 and CP/M 2.2 with or without the Gemini video card.

SYS has developed quite a few features over the generations. Firstly it meets the original intention of supplying good printer support by handling serial printers with optional handshaking and optional automatic page throw, and also Centronics compatible parallel interfaces. My personal version supports an interrupt driven interface to my Nascom 1. Secondly it provides a most unusual feature for CP/M systems - screen editing. Once one is used to screen editing, for example with NAS-SYS, it is very hard to return to the old days. The screen editing works in a similar way to NAS-SYS, except that a key must be pressed once at the start of each line to enter screen edit mode. This also works with the Gemini video card, and it is nice to be able to hop back up the screen and reenter incorrect CP/M commands, or even 80 character long lines of a Basic program. Another feature is screen dump, which means that when you press a certain key, the contents of the screen is printed out.

I have just finished a new version of SYS for the new double density disk card from Gemini, but this is not so important because the MOVCPM software provided with it is very good anyway. I have also created a special version for this disk card which has a "pseudo disk" which is really two 64K RAM cards on other pages, pretending to be a very high performance disk! I also look forward to seeing the new Nascom disk card, also double density, which is rumoured to be well under way.

Before criticising CP/M it is important to remember that much of its appearance to the user depends on how it has been implemented on a particular machine. I have seen several machines with errors or inadequacies in the CP/M implementation. Both users and suppliers then unfairly blame Digital Research for what is in fact a poorly written BIOS. To put both sides of the argument, CP/M is crude and unfriendly, but it is also a simple and effective means of providing a standardised software interface for a vast variety of good software, provided it is installed correctly.

--ooOoo--

IMPORTANT NOTE for owners of Henelec/Gemini G805 disk systems and Henelec FDC controller card kits using CP/M.

With the recent development of the Gemini G809 double density disk controller card, a deficiency in the original Henelec Format program has come to light. Two things about the Gemini G809 card have forced a change. Firstly, the G809 uses the more recent Western Digital WD1797 controller chip set, and secondly the G809 BIOS automatically selects between double/single density on any drive other than drive A. The single density format is identical to the Henelec/Gemini G805. This should mean that you can zap disks from the G805 into drive B of the G809 and PIP stuff in either direction with impunity. Unfortunately, the WD1797 senses which side of the disk is in use by picking up a 'sides' identifier byte slotted in the gap between the original track and sector identifiers of the G805 format. Now a disk formatted with the original Henelec software will not supply the 'sides' flag, so the G809 will think that the disk has only one side, with possibly disastrous results. To get round this problem a new format program has been written. This provides compatibility with the G809, as well as providing a worthwhile 37 to 45% improvement in operating speed for G805 systems running at 4MHz without WAITs. This new format routine is available from Henry's and Interface at a nominal handling charge of 50p to customers who wander in clutching a disk in their mits. Or 75p (plus the cost of a disk if you don't send one) through the post.

RAM 'A' board complete except for RAM or ROM.
Operates at 4MHZ with wait state 65.00
Naspen in EPROM 20.00 Bits & P.C's Toolkit 20.00
Tel 01-994-2938 (evenings)

Nascom 8 amp Power Supply at 85.00
Nascom 2 with Nas Sys 3 & Graphics ROM at 270.00
Nascom RAM B 48K fitted at 150.00
Nascom RAM A 32K fitted, 4MHz De-Plagued at 70.00
16, 4116 Dynamic RAMs 250 ns (-4) Offers?
Softy 1, with PSU and case at 100.00
Will haggle on above, ring Paul on 02518-2639

PRINT Routine By P. Forrester

Someone out there may be interested in the solution to a little problem which perplexed me for a while. My Olivetti TE300 teleprinter seems to be a bit slow on the carriage return, and, when using the excellent Naspen, tended to produce a ragged left hand edge, which was a little distressing. The solution would be to make the printer wait at the start of the line by outputting a non printing character, but the problem was - just how do you do that? The Naspen manual states that the printer routine is reflected through address 1010H to 101FH, and if you look there you find that it contains DF 6E C9, which is a call to, and return from, the serial output routine XOUT. So the first thing to do is to replace these bytes, using the M command, by a jump to somewhere else, say to 0C80. Then at 0C80 you put in a routine which checks each character passing through the A register to see whether it is a CR (0DH in ASCII). The character is first printed, then if it is not a CR, control is returned and the next character is checked, but if it is, a non-printing character is inserted. The following code, derived with the help of the equally splendid Zeap, did just that and cleaned up the left hand edge beautifully. So all you have to do is modify 1010 to C3 80 0C, and 0C80 to

```
06 0D B8 20 04 DF 6E 3E
07 DF 6F C9
```

The assembler program which produced this is as follows:

Z80 Assembler - Source Listing

```
0C80          0010      ORG  #0C80
0C80 006E      0020 XOUT  EQU  6EH  ;XOUT
0C80 060D      0030      LD   B,0DH ;CR
0C82 B8        0040      CP   B ;CHAR IN A
0C83 2004      0050      JR   NZ,OUT
0C85 DF6E      0060      SCAL XOUT
0C87 3E07      0070      LD   A,07  ;INSERT
0C89 DF6E      0080 OUT   SCAL XOUT;IF CR
0C8B C9        0090      RET
```

So now the Naspen output is cleaned up, and Zeap also produces clean printing because it has its own built-in line delay using the K command, but it would be nice to be able to LIST BASIC programs as well using the delay routine. NasSys 1 provides the U command for user-defined print routines, which are pointed to by an address stored at 0C78; so the first thing to do is to M 0C78 80 0C. But as it stands the routine doesn't work. Eventually it dawned that the routine never returns with a CR in A since it was sent to the printer and then changed to a non-printing character; thus the VDU display routine never gets CRs and you can't enter any commands! The solution is to put CR back in A just before returning, so at 0C89 we insert SCAL XOUT, LD A 0DH, RET (thereby creating two exits from the routine, which is said to be bad practise, but who cares?) and the code insert starting at 0C80 becomes:

```
06 0D B8 20 09 DF 6E 3E
07 DF 6E 3E 0D C9 DF 6E
C9
```

The second routine can, in fact, be used for printing both Naspen and BASIC inputs, and can be conveniently loaded from a short piece of paper tape.

Another problem which has caused me some difficulty, is how do you use more than one USR function in a BASIC program? To call a single machine code routine from BASIC is fairly straightforward, and, following the instructions in the manual, all you do is insert the starting address of the routine in USRLOC at 1004H, using the M command, before calling it. But there is only one entry point though which the machine code routines must pass; however, the parameter of the USR function can be accessed from machine code by calling a routine called DEINT whose address is contained in E00BH, which turns out to be located at E98BH - this provides the clue as to how to differentiate between the different functions. The answer is to produce a jump table of addresses of the routines to be accessed from BASIC and use the parameter passed through DE to determine which one to jump to.

The ZEAP listing below shows how this can be done. The first few bytes poke the starting address of the jump-table routine into USRLOC; in this example I used 0C80. Then you collect the USR parameter from DE by calling E98BH, and first make sure that it is not greater than the number of subroutines available - if it is, you print an error message and return to BASIC. If the value contained in DE is within range, it is doubled (because the addresses are two bytes long) and added to the address of the start of the table. The contents of that address are then jumped to and away you go. To illustrate the method, the listing shows three trivial subroutines which poke a character onto the screen, and then return a value to the appropriate USR function by placing it in the AB registers. The routine whose address is stored at E00BH (which turns out to be F0F2H) is then called and the number placed in AB is returned as the value of the USR function. Although only three subroutines are shown, you can, of course, use as many as you like.

Incidentally, if you are using ZEAP to develop the USR routines it is worth moving the edit buffer away from 1000H since BASIC will overwrite it, and this can be very frustrating. The buffer can be moved, for example, by calling ZEAP from NasSys by the command E 0000 3000 2000, which puts the buffer between 2000 and 3000H.

1004	0010	ORG #1004
1004 800C	0020	DEFW #0C80
	0030	;This puts the start
	0040	;address in USRLOC
	0050	;at 1004H
	0060	;
	0070	;Now for the routine
	0080	;itself
0C80	0090	ORG #0C80
0C80 0D8BE9	0100	CALL #E98B
	0110	;This returns with the
	0120	;value of the parameter
	0130	;in DE
0C83 210200	0140	LD HL,2
0C86 ED52	0150	SBC HL,DE
0C88 3810	0160	JR C,0BDS
	0170	;This checks that the
	0180	;parameter is not larger
	0190	;than the number of
	0200	;subroutines and jumps
	0210	;out if it is
0C8A 21940C	0220	LD HL,TABLE
	0230	;start of subroutine table

0C8D	19	0250	ADD	HL,DE	
0C8E	19	0260	ADD	HL,DE	
0C8F	5E	0270	LD	E,(HL)	
0C90	23	0280	INC	HL	
0C91	56	0290	LD	D,(HL)	
0C92	EB	0300	EX	DE,HL	
		0310	;Address of subroutine in HL		
0C93	E9	0320	JP	(HL)	
0C94	000D	0330	TABLE	DEFW #0D00	;subroutine
0C96	200D	0340		DEFW #0D20	;addresses
0C98	400D	0350		DEFW #0D40	
0C9A	EF	0360	OBDS	RST	28H
0C9B	50617261	0370		DEFM	/Parameter out of Bounds/
	6D657465				
	72206F75				
	74206F66				
	20426F75				
	6E6473				
0CB2	00	0380	DEFB	0	
0CB3	C9	0390	RET		
		0400	;		
		0410	;First demonstration subroutine		
0D00		0420	ORG	#0D00	
0D00	E5	0430	PUSH	HL	
0D01	C5	0440	PUSH	BC	
0D02	F5	0450	PUSH	AF	
0D03	21D008	0460	LD	HL,#08D0	;Pokes an A
0D06	3641	0470	LD	(HL),65	;on the screen
0D08	3E00	0480	LD	A,#00	;and returns
0D0A	06FF	0490	LD	B,#FF	;with 255 in AB
0D0C	0DF2F0	0500	CALL	#F0F2	
0D0F	F1	0510	POP	AF	
0D10	C1	0520	POP	BC	
0D11	E1	0530	POP	HL	
0D12	C9	0540	RET		
		0560	;Second subroutine		
0D20		0570	ORG	#0D20	
0D20	E5	0580	PUSH	HL	
0D21	C5	0590	PUSH	BC	
0D22	F5	0600	PUSH	AF	
0D23	21D808	0610	LD	HL,#08D8	;Pokes a B on the
0D26	3642	0620	LD	(HL),66	;screen and returns
0D28	3E14	0630	LD	A,20	;another number in AB
0D2A	0614	0640	LD	B,20	
0D2C	0DF2F0	0650	CALL	#F0F2	
0D2F	F1	0660	POP	AF	
0D30	C1	0670	POP	BC	
0D31	E1	0680	POP	HL	
0D32	C9	0690	RET		
		0700	;		
		0710	;Third subroutine		
0D40		0720	ORG	#0D40	
0D40	E5	0730	PUSH	HL	
0D41	C5	0740	PUSH	BC	
0D42	F5	0750	PUSH	AF	
0D43	21E008	0760	LD	HL,#08E0	
0D46	3643	0770	LD	(HL),67	
0D48	3E1E	0780	LD	A,30	
0D4A	061E	0790	LD	B,30	
0D4C	0DF2F0	0800	CALL	#F0F2	
0D4F	F1	0810	POP	AF	
0D50	C1	0820	POP	BC	
0D51	E1	0830	POP	HL	
0D52	C9	0840	RET		

Book

BOOK REVIEW

=====

By Rory O'Farrell

SOFTWARE TOOLS by B.W.Kernighan and P.J.Plauger, published by Addison Wesley.

There are few books that make a significant contribution to one's thinking. I doubt if anyone who has read this book has been utterly unaffected by it. The authors state that the purpose of the book is to teach how to write good programs that make good tools. They do this by presenting a set of programs which are designed to stand on each others shoulders to build into a full set of tools. The point is made that frequently one has to lash up a little program to do a specific thing - as it might be to count words - and when the purpose of the program has been met, the program is discarded. It is the contention of the authors that such a program is a tool, and should not be discarded. Moreover, they suggest that the quick lash up, while it gets the job done, suffers from a lack of thought and documentation. This lack of thought and documentation means that, if the program is resurrected at some stage, it is almost incomprehensible, and that the resurrector, even though he may be the author in the first instance, may well find it easier to rewrite the entire program from scratch.

Kernighan and Plauger set out in this book to illustrate a collection of general purpose tools - a series of programs which can readily be called on to do certain jobs. They discuss in a readable and clear way, the criteria involved in the design, the problems of implementation, and above all the flexibility of a good tool. If after reading this book, all one got from it was the discipline of clear thinking, that alone would be worth the price. But it is possible to actually implement the programs in this book on a machine, and their utility might be worth many times the book cost.

The programs are written for the most part in a language called RATFOR (RATional FORtran). This is so similar to structured English that it need cause no great concern. Some of the examples are also given in PL/1, which is similar to Pascal. It is a trivial matter to translate the clear structure of these programs to any machine language, or (better still) to higher level languages. I remember that the point has been made (but I can't find the reference) that the programs are not necessarily the most efficient in the world, but that one's own time is more important than the computer's.

How true this is of the Nascom owner. Do you care if your poor hardworked Nascom has to stay up all night? No, you don't, as long as the job is done when you want it!

In this book, a number of simple programs such as count a character, count a word etc., are introduced early on. Building on these, the book then proceeds to discuss file structures, sorting, text patterns, text editing, formatting of output, up to the level of formatting for composing machines (the book is photset, using some of their own tools, and one couldn't tell it from a "real" book!), a microprocessor, and concludes with a RATFOR - Fortran Preprocessor. It would even be possible to convert that to a RATFOR-BASIC translator. It makes the minimum assumptions about the underlying operating system, dealing in a few primitives such as "Get a char", "Put a char".

I would suggest that this book is not for the absolute beginner, but anyone with a little experience fiddling around on a microcomputer should find it exciting and profitable to read.

VIDEO ROUTINE

SOFTWARE FOR THE GEMINI VIDEO CARD

R. BEAL

=====

=====

The Gemini video card is "port mapped" instead of "memory mapped". This means that instead of occupying an area of memory in the computer which can be accessed directly by the processor, it is instead accessed through the Z80 ports by IN and OUT instructions.

This has the advantage that it doesn't tie up 2K of RAM, which the 80*25 display requires, and it also means that the screen driving software, which is quite complicated, becomes no more than a few instructions to the program driving the Z80 processor on the video card. Therefore I think that this was the best approach.

However, there is also an obvious disadvantage of this method. Many programs have been written which access video RAM directly, in particular various games programs with fast moving objects. Normal programs tend to produce output in the conventional way, and these work with the video card without any difficulty.

One solution is to modify programs to send the special escape sequences to the video card in the normal way to position the cursor and then write the characters required. This is quite possible, and will be suitable in some cases, but it tends to be a rather complicated and tedious job to program.

Therefore after some thought I have developed a little piece of software which allows programs to access an area of memory which acts as "pseudo video RAM". All you have to do is to assemble the code below with VAREA set to the start of a 4K area of RAM which is not in use. Then call the routine VINIT at the start of your program. You can then read and write to this area exactly as if it is video RAM. Then whenever you want to update the screen, just call the routine VIDEO. This routine works out the most efficient sequence of commands to send to the video card, by maintaining a record of the previous screen contents. You can mix calls to VIDEO with conventional I/O to the screen, which will not be affected. For example, calls to VIDEO will not disturb the cursor position. Using this system, I was able to convert the PIRANHA program which was published in an earlier INMC to use the video card, with only minor alterations to allow for the increased screen size.

Naturally, this system can never give the very high speeds achieved by true video RAM displays. Also, the screen area is much larger and this means that programs take longer to process the additional data. However, most programs of this type spend most of the time sitting in delay loops, and it should only be necessary to reduce the delays to get the system running at a good speed.

The VIDEO routine uses the transparent mode of access to the screen, so there is no speckling on the display whatsoever, which is an improvement over the black specks on the Nascom 2, to say nothing of the snowstorms of the Nascom 1.

SPECIAL INMC COMPILATION ISSUE

=====

We are pleased to announce that the 'best' and 'most relevant' bits of INMC issues 1 - 7 (Oct. '78 to May '80) have been compiled to produce a special issue. This is available from your distributor for 2.50, or from ourselves at 3.00 including postage.

TITLE VID Video card routines
SUBTTL 6 August 1981
.Z80

EXTERNAL VAREA

GLOBAL VINIT,VIDEO,VIDC

; VAREA is the address of a 4K area of free memory.
; The first 2K (AREA1) is addressed by the program
; exactly as if it is Video RAM.
; The second 2K (AREA2) is used to hold the previous
; screen image, and this area is automatically
; updated.

; To use this system, call VINIT at the start of
; the program and then call VIDEO to update the video
; card image whenever required.

; Richard Beal Copyright 1981

; Definitions

001B	ESC	EQU 1BH	; Escape character
0050	COLS	EQU 80	; Columns
0019	ROWS	EQU 25	; Rows
0800	VLEN	EQU 2*1024	; Length of each area (2K)
00B1	PVDUD	EQU 0B1H	; IVC Data port
00B2	PVDUS	EQU 0B2H	; IVC Status port

; Work area

0000'	VIDS:	DEFS 2	; Start of changed area
-------	-------	--------	-------------------------

; Initialise video areas

; Call this routine at the start of the program

0002'	E5	VINIT: PUSH HL	
0003'	D5	PUSH DE	
0004'	C5	PUSH BC	
0005'	21 0000*	LD HL,VAREA	; Clear screen
0008'	36 20	LD (HL)," "	
000A'	11 0001*	LD DE,VAREA+1	
000D'	01 07FF	LD BC,VLEN-1	
0010'	ED B0	LDIR	

; Set AREA2 to impossible values to ensure

; entire screen copied on first call to VIDEO

0012'	21 0800*	LD HL,VAREA+VLEN	
0015'	36 00	LD (HL),0	
0017'	11 0801*	LD DE,VAREA+VLEN+1	
001A'	01 07FF	LD BC,VLEN-1	
001D'	ED B0	LDIR	
001F'	C1	POP BC	
0020'	D1	POP DE	
0021'	E1	POP HL	
0022'	C9	RET	

; Routine to update video card as quickly as possible
; based on the current and previous screens
; Call this routine to update the video card screen

```

0023' F5          VIDEO: PUSH AF
0024' E5          PUSH HL
0025' D5          PUSH DE
0026' C5          PUSH BC
0027' 21 FFFF*    LD HL,VAREA-1 ; Start of AREA1-1
002A' 11 07FF*    LD DE,VAREA+VLEN-1 ; Start of AREA2-1
002D' 01 07D0*    VID2: LD BC,VAREA+COLS*ROWS ; End position
0030' 23          VID3: INC HL ; Next position
0031' 13          INC DE
0032' B7          OR A ; Test for end
0033' ED 42       SBC HL,BC
0035' 09          ADD HL,BC
0036' 28 1F       JR Z,VID9 ; Jump if end
0038' 1A          LD A,(DE) ; Test for no change
0039' BE          CP (HL)
003A' 28 F4       JR Z,VID3 ; Loop if no change
003C' 22 0000*    LD (VIDS),HL ; Store start of changed area
003F' 23          VID4: INC HL ; Next position
0040' 13          INC DE
0041' B7          OR A ; Test for end
0042' ED 42       SBC HL,BC
0044' 09          ADD HL,BC
0045' 28 0D       JR Z,VID7 ; Jump if end
0047' 1A          LD A,(DE) ; Test for change
0048' BE          CP (HL)
0049' 20 F4       JR NZ,VID4 ; Change, so loop
004B' E5          PUSH HL
004C' D5          PUSH DE
004D' CD 005C*    CALL VIDOUT ; Output changed area
0050' D1          POP DE
0051' E1          POP HL
0052' 18 D9       JR VID2 ; Continue with unchanged area
0054' CD 005C*    VID7: CALL VIDOUT ; Output final changed area
0057' C1          VID9: POP BC ; Restore registers and return
0058' D1          POP DE
0059' E1          POP HL
005A' F1          POP AF
005B' C9          RET

```

; Output a series of characters to the video card
; starting at (VIDS)

```

005C' E5          ; HL must point to the end of the changed area+1
005D' 3E 1B       VIDOUT: PUSH HL ; Save address of end+1
005F' CD 009E*    LD A,ESC ; Escape W sequence
0062' 3E 57       CALL VIDC
0064' CD 009E*    LD A,"W"
0067' 2A 0000*    CALL VIDC
006A' 54          LD HL,(VIDS)
006B' 5D          LD D,H ; Store VIDS in DE
006C' 01 0000*    LD E,L
006F' B7          LD BC,VAREA ; Calculate position in display
0070' ED 42       OR A
0072' 7D          SBC HL,BC
0073' CD 009E*    LD A,L ; Low order offset
0076' 7C          CALL VIDC
0077' CD 009E*    LD A,H ; High order offset
007A' E1          CALL VIDC
007B' B7          POP HL ; End+1
007C' ED 52       OR A ; Calculate length
          SBC HL,DE ; DE is start

```

```

007E' 7D          LD A,L          ; Low order length
007F' CD 009E'    CALL VDC
0082' 7C          LD A,H          ; High order length
0083' CD 009E'    CALL VDC
0086' 3E 54       LD A,"T"       ; Transparent move
0088' CD 009E'    CALL VDC
008B' 44          LD B,H          ; Put length in BC
008C' 4D          LD C,L
008D' 21 0800     LD HL,VLEN      ; Set HL to position in AREA2
0090' 19          ADD HL,DE
0091' 1A          VOUT4: LD A,(DE) ; Output character
0092' CD 009E'    CALL VDC
0095' 77          LD (HL),A       ; Set AREA2 to AREA1
0096' 13          INC DE
0097' 23          INC HL
0098' 0B          DEC BC          ; Count number output
0099' 78          LD A,B
009A' B1          OR C
009B' 20 F4       JR NZ,VOUT4
009D' C9          RET

; Send character to video card
009E' F5          VDC: PUSH AF
009F' DB B2       VDC2: IN A,(PVDUS) ; Wait until ready
00A1' 0F          RRCA           ; Test bit 0
00A2' 38 FB       JR C,VDC2
00A4' F1          POP AF
00A5' D3 B1       OUT (PVDUD),A   ; Output data
00A7' C9          RET

```

END

No Fatal error(s)

Multi-load

Automatic entry of several Machine Code programs

By A.J.Fry

The Generate command in Nas-Sys 1 automatically loads and starts a program. However if the 'Execute' address which is given to start the program is merely the address of the DF 5B monitor return instruction in that program (There is not a DF 5B?!! - go and read the manual on how to end a program) then the only action will be to return to monitor control.

A second program may now be entered automatically by using the same process. The second program need not contain a DF 5B provided the address of the DF 5B in the first program is given as the 'Execute' address when recording the second program using the Generate command. Subsequent programs may be similarly entered. Thus a cassette tape may be prepared which will load any number of programs (or a main program and several subroutines) without the need to enter a single command via the keyboard. No program will be executed until the command is given either manually or as the execute address of the final program/subroutine which is loaded.

The individual programs or subroutines need not occupy adjacent blocks of memory and no 'unoccupied' memory need be recorded on the tape as would be necessary if the whole group were widespread through memory and were then saved as a single program.

Get some STYLE !

NOTES ON PROGRAMMING STYLE.

by Rory O'Farrell.

=====

When writing programs, one is of course anxious to get the program finished and running as fast as possible. This is very understandable, but it leads to trouble. In one's haste to get running, one usually omits one very important item - COMMENTS! Comment your program fully. If in assembler, use as many comments as possible, and if in BASIC use REM statements. Use such comments or REMs also to space out the program and make it more legible. Admittedly in BASIC the REMs can slow down the speed of execution of the program, but they can always be removed with a Toolkit (.K Command in Henry's, as reviewed in INMC80 no 2). If intending to do this, please, please don't GOTO or GOSUB a REM statement, or you will have errors when you try to run the program. In machine language, the comments will slow the assembly slightly, but they will not effect the runtime speed. In any event, make sure that you have a fully commented or REMed tape of the program to file.

Another point of style. Use Manifest Constants as much as possible. If in machine language you wish to designate the Top Line of the Screen, don't write LD HL, OBCAH. Instead use an EQU statement at the program start:

```
TOPLIN:EQU OBCAH
```

and in the program

```
LD HL, TOPLIN
```

Should it now prove necessary for some reason to change the line, e.g. using the second line instead, we need only change the EQU statement, like this:

```
TOPLIN:EQU O80AH
```

and the assembler will change all occurrences of TOPLIN to point to the correct line. No chance of missing a LD HL,OBCAH inadvertently. Similarly, common values such as for carriage return (ODH) and backspace (08H) should be declared in the same way. This makes the program more readable, and consequently helps cut down on the comments. It also has side effects. Suppose one writes a program to accept 8 digit numbers, and to sort them into order. Obviously, such a program will have a number of occurrences of the number 8. It will occur wherever backspaces are processed, as one ought to be able to make and correct an input error fairly easily, and it will also have 8s in the counting loops. Some months after writing this program, you decide, for reasons best known to yourself, that you wish to modify this to process 13 digit numbers. "No problem", you say, "I've just got to fly through the program and change all the 8s to 13s."

So you are sitting down at your trusty Nascom, changing 8s to 13s with gusto, and the phone rings. It is a) the hospital to say that the baby has arrived, b) the bank manager to say that you can have the extra overdraft, c) the Prime Minister to ask you if you'd accept a Peerage (delete as appropriate), so you have to answer. After a long conversation, you return to your patching. Where were you? Was that 8 a backspace or an entry length counter? Is that 13 (or ODH) a carriage return or an entry length counter? HELP! All this could have been saved if you had used manifest constants at the top of the program (or didn't talk to a) women, b) bank managers, c) Prime Ministers).

The use of manifest constants at the top of the program greatly helps the alteration of it in the future, the conversion of it to other systems, which may have different parameters to yours (think of the T2/T4/BBUG monitors use of different control characters), and considerably improves the legibility and hence the clarity of the program. It also, at the expense of a slightly longer assembly time, ensures that any changes are correctly installed in the program. If your assembler supports multiplication and division, it can also calculate displacements up and down tables, and computer constants, so that they can be manifested in this way. For example, in preparing a table to hold 4 byte entries, one can write:

```
NUMENT:EQU 131 ;Number of entries expected
```

```
ENTSIZ:EQU 4 ;length of individual entry
```

```
TABLE:DEFS ENTSIZ*NUMENT
```

and if the entry length or the number of entries is changed, the assembler makes the change automatically. Should the assembler not support multiplication or division, one could write

```
TABLE:DEFS NUMENT+NUMENT+NUMENT+NUMENT
```

This is a bit more cumbersome, but it does clearly show how the size of table is allocated, particularly if accompanied by a comment to the effect that each entry is four bytes long.

We haven't spoken of BASIC for some time. Has this business of manifest constants anything for that? Yes! BASIC supports a considerable number of variables, which are a letter, followed by a letter digit or space, to give a larger number than could reasonably be exhausted in the course of a program. It is possible to define a number of variables at the start of a program such as:

```
100 CR=13:BS=8:FF=12:REM values for carriage return, b/s, and form feed
```

Similarly, if testing for a value in a list, one could define:

```
110 YES=1:NO=0
```

and use these as manifest constants. This considerably improves the legibility of the program, and anything which does that in BASIC has to be a good idea! It also improves the running speed, as BASIC stores all constants in ASCII internally, and each time it comes across one, it has to translate it into binary, but each variable is already translated ready to go! Notice also that it is possible, at the expense of a slight increase in running time, to use long variable names. Note however that BASIC only uses the first two characters of the name to identify it. This means that FIRST and FIZZ are the same to BASIC. An easy way out is to put a unique padder in front of the meaningful part of the name (meaningful to you, but not to BASIC) - say Z1FIRST and Z2FIZZ. It becomes very easy to disregard the leading characters, which make the variable distinguishable to BASIC, but you the human look mainly at the end.

In "Writing Interactive Compilers and Interpreters", P.J. Brown says "Thus some compiler writers say that a compiler should contain no numbers at all, except perhaps, as a special treat, the occasional zero or one." (p69). While this may be a little extreme, it is none the less a good guideline.

The controversy over the GOTO has been raging for some years now (GOTO Statement considered Harmful, Dijkstra, E.W., CACM, Vol 11, No.3. Mar 1968 and other papers) and it would be wrong to go very deeply into the depths of the dispute here. As careless use of the GOTO in BASIC can lead to an utterly incomprehensible program, I think no one will support its indiscriminate use towards that end. Use of the GOTO can cause problems in BASIC which can be very difficult to find.

For example, consider the following program:

```
10 N = 6
90 FOR J=1 TO 100
100 FOR I=0 TO 10
110 IF I=N THEN 200
120 NEXT I
....
200 PRINT "N = ";I
210 NEXT J
```

At line 210 one will get an NF error. The program has jumped out of the FOR I loop, and now meets a statement NEXT J. But it is in an I loop, so it can't cope. The answer is to force the closure of the I loop thus:

```
110 IF I=N THEN II=I:I=10:NEXT I:I=II:GOTO 200
```

This form of error only occurs when jumping from an inner loop to the outer one and can be overcome as illustrated. When jumping out of a single non nested loop, a plain GOTO is often considered sufficient - "It doesn't give any errors", I hear a chorus. Each time such an abnormal exit is made from a loop there is housekeeping space allocated on the BASIC's stack - and there is only space for a number of loops. I have not heard of anyone running out of space, but they probably wouldn't boast about it! The space for the number of nested FOR loops is limited, and it is bad practise to waste it - as after all, someday you might need 20 or 30 nested FOR loops!

The controversy surrounding the GOTO is particularly strong in those areas where writers of compilers with variables of limited scope forgather (as opposed to BASIC, where most variables have global scope). Because of the limited scope in the variable structures of languages such as Pascal, there is much more housekeeping associated with subroutine calling, and the GOTO (or in Pascal the Label/goto) causes havoc with this housekeeping information. So it is perhaps easier to outlaw the use of the GOTO in such languages. It is nevertheless one of the fundamental commands of BASIC, so we need feel no compunction about using it, but should do so wisely.

Always keep a tape of the fully commented (or REMed) final listing, in its full uncompressed form. If you have a friend with a Nascom and printer, it may be possible to arrange that you bring over your library copies of tapes, and get a listing off them. File your listings carefully. If you can get two listings, so much the better. Use one for planning improvements, and leave the other alone in the shelf. Make sure that you put a date and version number on each - a REM or comment at the start is the best way.



Misc.

Jottings

=====

by R. O'Farrell

Pascal

Due to a prolonged stay in hospital, Christmas, and the pressures of business, progress on the Pascal front has not been as quick as desired.

It is hoped to obtain a major subset of Pascal in the near future, which will be compiled for the Nascom under Nas-Sys. Watch this space for details. There is also available from the program library an implementation of the Chung and Yuen Tiny Pascal, which uses a Basic program to compile to P-code, and then runs the P-codes in a machine language interpreter. The whole can fit in a 32K machine! (using ROM BASIC). Xtal Basic will require more memory, or alternately to write the P-codes to tape and reload with the interpreter. Remember that P-codes are not position sensitive within the memory map of the machine. They cannot, of course, be moved around relative to each other within a block, but can be put anywhere in the memory map as a block, provided the Interpreter knows about it. Rewriting the Tiny Pascal in terms of itself, so that the first level bootstrap has been made, is under consideration, but such a project is marking time pending the arrival of the major Pascal subset.

It is hoped shortly to have a language called BASEX (BYTE Books) implemented for the Nascom, and there has been a suggestion that there may be a Tiny C available. This latter is dependent on permission from the authors, which is being sought.

The UNIX operating system for the PDP11 is written in the C language. The book Software Tools (reviewed elsewhere in this issue) is written on machines running under this system, and admits its debt to that system. Reading that book and what else has been written about the UNIX operating system would lead one to believe that it is a most interesting system. There have been some look-alikes published for microcomputers. In particular, one called OMNIX, which has been temporarily withdrawn for delousing, and CROMIX for Cromemco machines.

Interesting Points.

If one has a Nascom IMP, and suffers an occasional stutter or repetition of the character being transmitted when the handshake occurs, or other garbage being printed at about that time, then the cure may well be to pull the handshake line to +5V with a 200-300 ohm resistor, which may be connected from pin 8 of SKT 1 on an N1, or from TP3 on an N2 to a suitable connection point. According to our beloved chairman, D.R.Hunt (please can I stop banging my head on the ground and grovelling on my knees? It's very difficult to type accurately like this.), according to our b.c., D.R.H., the handshake line inside the Imp is pulled up with a 1k resistor, which he says is wrong, Wrong, WRONG for an o.c. output. He says that it should be maybe as low as 150 ohm (grovel, grovel - he'll say I'm wrong - grovel, grovel). Anyway, the mod. most certainly does work, and cuts out all duplicated and otherwise erroneous characters. Thanks, David (grovel, grovel).

Those using the Nascom 1 power supply, which has three three-legged device heatsinks in a row (current say two to three years ago) may experience intermittent system shut downs, bars on the T.V. screen, and other undesirable happenings. Try putting a heatsink on the little square bridge rectifier. If

that works, but the problem recurs, replace the rectifier. I know of one case where the rectifier was replaced by a 6 amp rated device, which was bolted to the case. The connections were made from the P.C. Board in heavy wire. No further problems occurred on either of the machines I know of that could be attributed to the P.S.U.

Has anyone heard of the National Semiconductor MM58174 Microprocessor Compatible Real Time Clock? This is a very neat 16 pin i.c. which costs about 10.00 (depending where you buy it). It requires a 38768Hz clock from a crystal controlled oscillator. It can run in power down mode as low as 2.2V, and keeps a track of time from 1/10 sec to leap years! It offers the option of interrupts on 0.5 sec, 5.0 sec, 60 sec. Would any highly intelligent person who has designed a circuit for this to run on a Nascom like to send me a copy?

I would like to thank the people who wrote to me with information about the IBM33FD floppy disks. The day they start to run is coming nearer. I intend, subject to changing my mind about the matter, to drive them from an input/output board, having cunningly substituted an SIO for the PIO. Beware!! The SIO is not pin-compatible, and an adapter is necessary. Also the SIO does not calculate the CRC with the correct initialisation to give IBM 3740 standard, so disks recorded in this way are system dependant. However doing it this way will allow me to see if there is any point in going further with these disk units, or if I throw them away (having first removed the stepper motors for a plotter) and buy some real 8" Shugarts or Qumes. Should this method of driving them work, in spite of being system dependent, I intend to build a Z80 based disk controller, which will be intelligent enough to handle the spooling of data to a printer, and perhaps even tape input and output. I've recently acquired an NCR 6000 baud tape unit, which is also intelligent I'm now using the term very loosely!

Nascom's BOOBS (or why are there no lady Nascomers?)

My reference to the method of checking the capture range of this Cottis/Blandford interface was in error. The correct date was the December 1978 P.C.W.

The reference to the A.C.C. newsletter should have been to Vol 6, No.5.

Who did it?

It would give me, and I'm sure many other Nascomers, the greatest satisfaction to hear that the **** who designed ("designed"? Daddy, Daddy, I thought you said it was ****. Ouch, oh!) the Famous Nascom series A RAM board had been shot at dawn, without benefit of a) a blindfold, b) a final cigarette. If I knew his name, I could label my picture of Hitler with it, and throw darts at it.

Teletype Printer mechanism (no keyboard).
Working but very noisy 35.00 o.n.o. 041.942.2482

Nascom 1, Nas Sys 3, and Buffer Board 95.00 or will sell separately.
Ring 041.332.3841

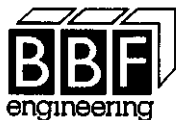
NASCOM 1 & 2 - MUSIC BOARD

- Extremely simple to program in basic or machine code.
- Generates notes over an 8 octave range.
- Driven from parallel output ports.
- Plugs into Hi-Fi for full sound.
- Sustains notes with no CPU overhead.
- Complete with full documentation and programming examples.
- £22.80 (kit), £24.50 (assembled and tested).
- Multiple boards may be used to great effect!

for further details phone 0582 35930

REPLACEMENT NASCOM INTER-CONNECTING CABLE WITH MUSIC BOARD HEADER CONNECTORS - £4.99

SOFTWARE FOR INSTANT ENJOYMENT - Comprehensive music entry program, 'MUSIC MAKER', can handle multiple channels, (16k min.) £6.00



R&P 65pence per order.

Dept. N, 82 Buckingham Drive, Luton, Beds.

Futura nascom-2 Software

Exciting and unusual graphical games for the Nascom 2 micro computer.

Sci-fi programs

Astrafire	16K,B,G	£ 5.50
Beam Me Up Scotty	16K,B,G	£ 5.00
X-Wing Star Battle	16K,B,G	£ 6.50
Alien Bombardment	8K,B,G	£ 4.00
Laser Duel	8K,B,G	£ 4.00
Martian Crossfire *	8K,B,G	£ 3.50
Star Wars *	32K,B,G	£10.00

Sports & other programs

Grand Prix	8K,B,G	£ 5.00
Downhill Skier	8K,B,G	£ 3.50
Computer Darts	8K,B,G	£ 4.50
Speedway	8K,B,G	£ 5.00
Dodgems *	8K,B,G	£ 4.50

B=Basic, G=Graphics, *=New program

Send SAE for program details. All programs are supplied on cassette and prices are inclusive.

Send Cheque/P.O. to FUTURA SOFTWARE
63, Lady Lane, Chelmsford, Essex, CM20TQ.

MicroValue

New Software for Nascom Systems

POLYDOS 1 A disk operating system for use with Nascom 1 or 2 and Gemini G805 Disk Systems. An incomparable and extremely well presented DOS that includes an editor and assembler and adds disk commands to the Nascom BASIC. MicroValue price £90 + VAT

MATHSPAK Double precision maths package on tape. MicroValue price £13 + VAT

MATHSPAK Handler Used in conjunction with MATHSPAK. MicroValue price £9.95 + VAT

Command Extender For use with MATHSPAK it extends BASIC's reserve word list. MicroValue price £9.95 + VAT

Logic Soft Relocater An integrated assembler and disassembler package which allows disassembly and reassembly from anywhere on the memory map. MicroValue price £13 + VAT

SAVE MORE MONEY

Standard Firmware for Nascom at Reduced prices

NASPEN	RRP £30 + VAT	MicroValue price £20 + VAT
Nas-Sys 3	RRP £25 + VAT	MicroValue price £20 + VAT
NasDis - D-Bug (EPROM)	RRP £50 + VAT	MicroValue price £30 + VAT
NasDis - D-Bug (TAPE)	RRP £40 + VAT	MicroValue price £20 + VAT
Imprint	RRP £30 + VAT	MicroValue price £20 + VAT
Bits & PCs Prog Aid	£28 + VAT	MicroValue price £20 + VAT

80 x 25 Video for Nascom

Nascom owners can now have a professional 80 x 25 Video display by using the Gemini G812 Intelligent Video Card with onboard Z80A. This card does not occupy system memory space and provides over 50 user controllable functions including prog. character set, fully compatible with Gemini G805 and G815/809 Disk systems. Built and tested £140 + VAT

NASBUS Compatible DOUBLE DENSITY Disk System - Available Ex Stock

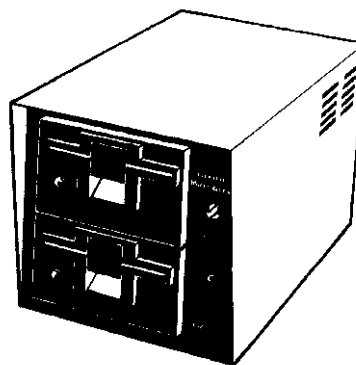
With hundreds in daily use the Gemini Disk system is now the standard for Nascom and Gemini Multiboard systems. Single or twin drive configurations are available, giving 350K storage per drive. The CP/M 2.2 package supplied supports on-screen editing with either the normal Nascom or Gemini IVC screens, parallel or serial printers, and auto single-double density selection. An optional alternative to CP/M is available for Nascom owners wishing to support existing software. Called POLYDOS 2 it includes an editor and assembler and extends the Nascom BASIC to include disk commands.

**Single drive system
(G809, G815/1)
£465 + VAT**

**Double drive system
(G809, G815/2)
£690 + VAT**

**CP/M 2.2 package
(G513)
£100 + VAT**

**Polydos 2
£90 + VAT**



*MicroValue Warranty

All products, except kits, sold by MicroValue dealers are supplied with 12 months' warranty and will be replaced or repaired by any dealer (even if you didn't buy it from him) in the group in the event of faulty manufacture.

YOUR LOCAL MICROVALUE DEALER

All the products on these two pages are available while stocks last from the MicroValue dealers listed on right. (Mail order enquiries should telephone for delivery dates and post and packing costs.) Access and Barclaycard welcome.



BITS & PC'S
4 Westgate, Wetherby, W. Yorks.
Tel: (0937) 63774.

ELECTROVALUE LTD.
700 Burnage Lane, Burnage,
Manchester M19 1NA.
Tel: (061) 431 4866.

**28 St Judes, Englefield Green,
Egham, Surrey TW20 0HB.
Tel: (0784) 33603. Tlx: 264475.**

TARGET ELECTRONICS
16 Cherry Lane, Bristol BS1 3NG.
Tel: (0272) 421196.

INTERFACE COMPONENTS LTD.
Oakfield Corner, Sycamore Road,
Amersham, Bucks.
Tel: (02403) 22307. Tlx: 837788.

HENRY'S RADIO
404 Edgware Road, London W2.
Tel: (01) 402 6822.
Tlx: 262284 (quote ref: 1400).

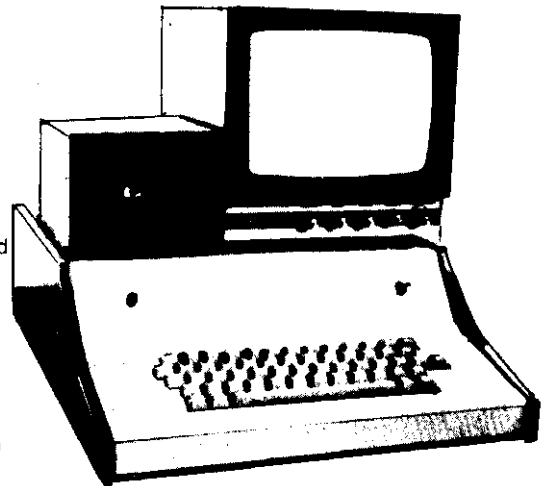
C.F.S. IS HERE

An economic, easy to use, reliable digital Cassette File Store — C.F.S. is available now for use with NASCOM 1 and 2 microcomputers.

FEATURES

- Sophisticated operating system provides a file handling capability superior to that of many floppy disc systems.
- High reliability ensured by use of professional digital recorder and automatic data verification.
- Fast data rate of 6000 b.p.s. equivalent to 1 Kbyte in 1.3 secs.
- Automatic storage and retrieval of Machine Code, BASIC, ZEAP and NASPEN.
- Controls up to 24 individually named files.
- 96 Kbytes of storage on single digital mini-cassette.
- Easy to install — connects to PIO port.
- Supplied as self-contained unit in compact rugged case with its own integral power supply.
- Low cost — complete system only £170 + VAT.
- For full details S.A.E. to:

**GRANGE ELECTRONICS LTD., (Dept. E), STONE LANE INDUSTRIAL ESTATE,
WIMBORNE, DORSET BH21 1HD (Tel: 0202 884752)**



HULLFORTH

A new FORTH compiler for NASCOM 1 & 2. HULLFORTH is a language well suited to almost any programming task, especially those applications requiring fast execution such as :-

Fast interactive games.
I/O interfacing etc...

The documentation provided with the program cassette is a 54 page A4 size booklet which not only serves as a reference manual but also has an extensive 'HANDS-ON' SELF-TEACHING section for newcomers to FORTH.

HULLFORTH runs in 16K minimum and is supplied on cassette in either 'CUTS' format at 300 baud for NAS-SYS or in NASBUG format for NASBUG.

When ordering please quote monitor and cassette format required.

PRICE --- £25 inclusive.

Send SAE for further information.

Mr A.F.T. WINFIELD,
148 GODDARD AVENUE,
HULL, HU5 2BP.

MERRY CHRISTMAS FROM

SKYTRONICS

2 North Road The Park Nottingham
Tel (0602) 45215

We are newly-appointed dealers, appointed by NASCOM covering Nottinghamshire, Derbyshire and Lincolnshire.

We will offer full NASCOM hardware and software support and service.

We will also supply the full Gemini range of NASBUS compatible products.

We have also started a new computer club for NASCOM in the East Midlands.

PLEASE PHONE US FOR ORDERS AND DETAILS.

ACCESS AND BARLAYCARD WELCOME



PROGRAM POWER

LUNAR LANDER SUPREMACY(16K/B/G) - classic space/alt landing simulation. Short, medium & long range scans show planet surface in varying detail. Continuously updated STATUS REPORT gives vertical, horizontal & relative velocity, altitude, fuel level, G factor & surface scan for suitable landing site. 8 skill selections. Brilliant graphics. £9.95

STARTREK II(32K/G/B) - enthralling, real time version from our Invasion Earth author, using M/C code sub-routines to great effect. Special features include larger galaxy, shielded homing warheads (fired by Klingons), time slots & non stop action. £9.95

INVASION EARTH(MC/G) - New improved version! 4 complexity ratings. 10 overall speeds. Variable shot speeds & alien descent rate. 4 invader types. Intelligent homing, exploding, angled, direct, multiple warhead & radio jamming missiles. £8.95

INVASION EARTH(MC) - as above with SOUND EFFECTS using AY-3-8910 CHIP. £10.95

"NASCOUNT" - PERSONAL FINANCE(16K/MC) - Make life simpler with this finance planner. Budget income/expenses month by month and highlight likely surpluses & deficits. Can be used to check bank account & record past income/expenses. 50 entries each period. Five digit codes with analysis by code & sub-code. Calculate cumulative cash flow to specified month end. Output to cassette & printer. £9.95

CONSTELLATION(16K/B) - Turn your screen into a telescope & view the stars from any point in the Northern Hemisphere at any time & date. Display stars by magnitude, identifying number or constellation. The telescope can be raised & lowered, zoomed in & out. Also output of star map to printer. £6.95

** NASCOM 1 - Cotts Blandford cassette interface for N2 format, reliability & fast load £14.90
B = Nascom BASIC (State Tape BASIC if required)
MC = Machine Code, G = Nascom Graphics. 8K RAM required unless otherwise stated.
ALL PROGRAMS SUPPLIED ON CASSETTE IN CUTS/KANSAS CITY FORMAT.

NASCOM 1 & 2

WORDEASE WORD PROCESSOR(MC) - Professionally written 4K word processor. 14 line window on text buffer & extensive on-screen editing facilities. Insert & delete characters, lines & paragraphs. Text manipulation - copy from one section of text to another, or read in additional material from tape to any point in the text. FIND & REPLACE facility. Text buffer size according to available memory.

Exceptional formatting capability - commands embedded in text allow complete flexibility & a variable tab position. Indent, line length & page length. Use of up to 10 'MACROS' permits automatic inclusion of headings, footings & other 'text repeats'. & also automatic page numbering. Output to printer - can vary character delay, inhibit line feeds & force upper case if required. An extensive manual is supplied (itself prepared on WordEase). (MANUAL ONLY - £1 refundable against program order) £25.00

VORTEX(MC)(State 16/32 or 48K) - Speed up your display of pixel graphics. Cassette holds 29 separate routines to be called from BASIC. Extensive instructions and examples supplied. Give your programs that professional touch! £8.95

CLUB MEMBERSHIP(32K/B) - Create a file of 200 Members - containing Name, Address, Date of Joining, Number, Remarks, Paid or not, Amend or Delete. Comprehensive search & sort routines. Partial or complete listings. Output to cassette & printer. £9.95

Super Startrek (16K/B) £6.95
Alien Labyrinth (16K/B/G) £6.95
Super LIFE (MC/G) £6.95
Cliff Invasion (B/G) £6.95
Space Fighter (B/G) £5.95
Cowboy Shoot out (MC/G/Sound) £4.95
Fruit Machine (B/G) £4.95
Road Race (MC/G/Sound) £4.95
Labyrinth (B/G) £4.95
Remember (MC) £4.95
WIRRAL PILOT V4.0 £12.50

CLIFF INVASION(B/G) - the aliens have landed in droves. You have one remaining laser base. Your only chance - shoot the ground from under them as they descend the cliffs towards you. Landslides created. Errors in direction & elevation of shots are costly. 3 levels of skill. Like all aliens, they breed like rabbits! £6.95

MUSIC BOX

Now you can make music with NASCOM. Easy to follow program allows you to key in old favourites or have fun composing your own tunes. 7 octave range with staccato option. 8 tempos. Set note duration or tap in rhythm as required. Comprehensive editing. Delete, insert or amend notes. Single step forward & backwards through tune. Add new lines within declared array size. The program includes tape generating & play-back routines & is supplied with 2 demonstration melodies & instructions for connecting your Nascom to an amplifier/speaker such as our unit below. Min. 16K required - please state T4 or Nas-sys/2 or 4. £9.95

AUDIO INTERFACE BOARD/SPEAKER

Compact & ready assembled, suitable for use with "MUSIC BOX" & other 'sound effects' programs. 3 simple connections. Complete with instructions on programming for sounds. £10.75

AY-3-8910 SOUND CHIP

Program up to three independent channels with music & sound effects! Supplied with detailed write-up. £20.645

SOUND CHIP INTERFACE BOARD - Using the PIO, program up to four sound chips at once - i.e. 12 separate programmable sounds. Each board contains an interface allowing a further board to be attached. Only simple link changes required. Connect to amplifier/speaker such as our unit above. £13.50

SOUND CHIP DEMO PROGRAM - First mode gives direct entry to chip registers, making experimentation simple & thus rapid appreciation of chip's potential. Second mode turns keyboard into 7 octave 'piano', displaying state of registers & notes up to 31 being played. £5.95

60 page Data Manual (inc VAT) £2.25

BOARD GAMES

Games Graphics ROM/Adaptor £18.90
SARGON CHESS Book (inc VAT) £9.50
- with program £19.50
Book/program/ROM/Adaptor £35.00
Draughts (B/G) £7.95
Backgammon (16K/B/G) * £7.95
(* state ORD. or ROM version)

Please add 55p/order P & P + V.A.T. @ 15%
Sae for FULL CATALOGUE (Now over 50 items!)
PROGRAM POWER
5, Wensley Road,
Leeds LS7 2LX
Telephone (0532) 683186



VOCABULARY TUTOR (B)

French & German - learn vocab. the easy way. Translation tests in both directions. Results of tests given incl. part correct answers. Also quick 'self-test' option to save keying in. New vocabulary can be easily added. Especially useful for new students. £5.95

PROMPT (B)

Devised to take the strain out of learning text for acting, after dinner speeches etc. this program on a line by line system gives you either a word by word 'prompt' or a letter by letter in each word 'prompt'. The fewer 'prompts' required the greater you score. Excellent value at £5.95



PROGRAM POWER

PROGRAM POWER

PRESENTS "THE KEYS OF KRAAL" AN ADVENTURE PROGRAM for NASCOM

Legend has it that KRAAL - known by the Bedouin as the 'Temple of the Undead' - houses a fabulous treasure and the four Locks of Eternity. It is believed that anyone who finds the right key to one of the locks will break the curse of Kraal, release the souls of lost adventurers and escape with treasure of untold proportions. No-one has yet lived to prove this theory.

The temple is inhabited by Monsters and Magical Beings. Your sword and arrows may be sufficient to destroy Gargoyles, Minotaurs, Mummies & The Cyclops etc., but you will need the various spells you find to combat JUBILEX, ASMODEUS, GERYON and the other magical beings. Beware also the Vampire Bats who will sap your strength requiring you to find a life-giving Elixir, and the SPIDER GODS whose attentions are usually fatal!

The program requires 24K RAM and is exceptionally well presented. Nine chambers are depicted at one time with Monsters & Demons continually moving within their cells, and making 'real time' attacks. Swords flash, arrows fly & spells home-in on the victim! Each game is played against the clock & can be saved on tape after generating it - play it again & again. (Nascom BASIC/Graphics).

Send NOW for this excellent program. Only £8.95

to PROGRAM POWER

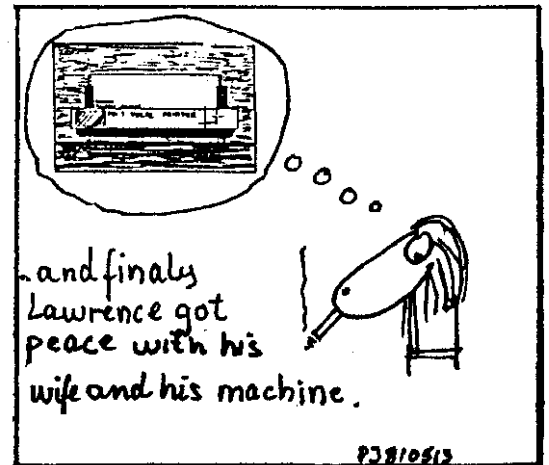
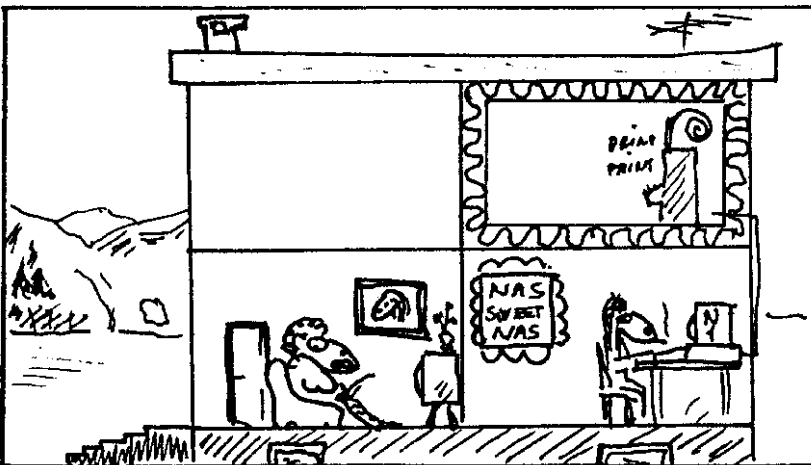
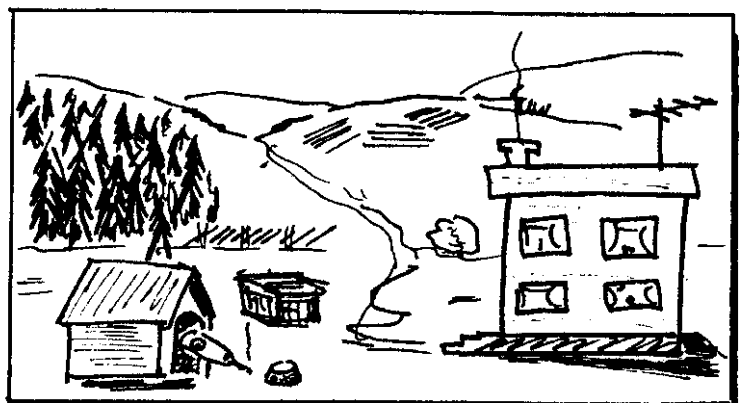
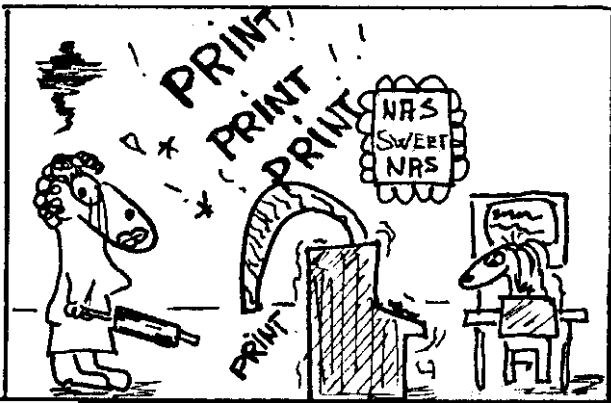
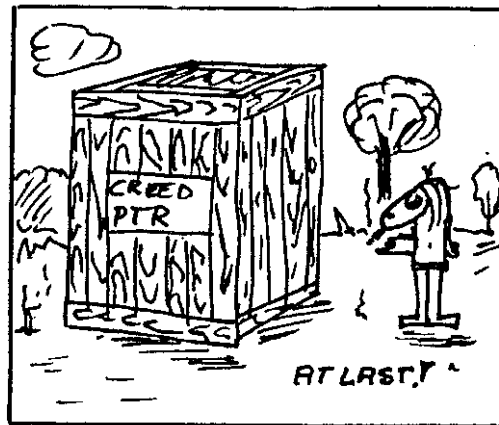
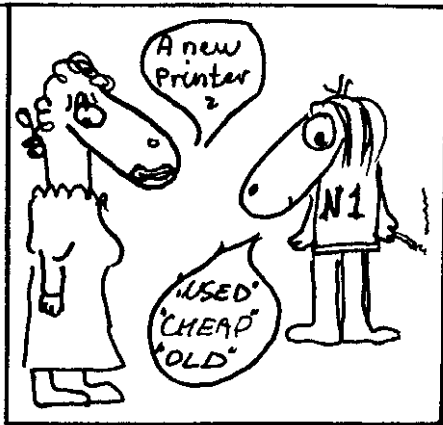
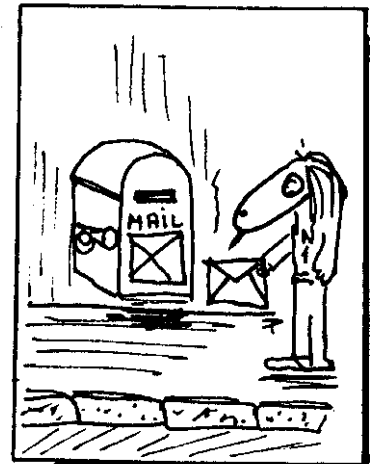
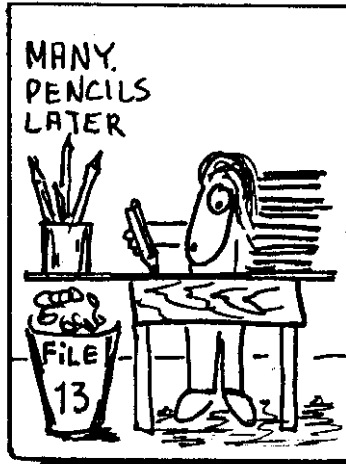
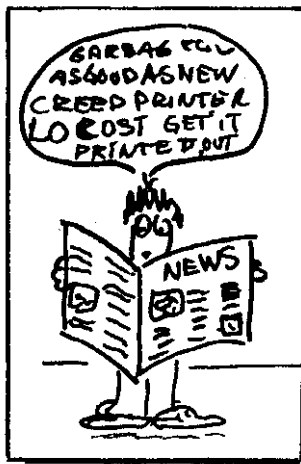
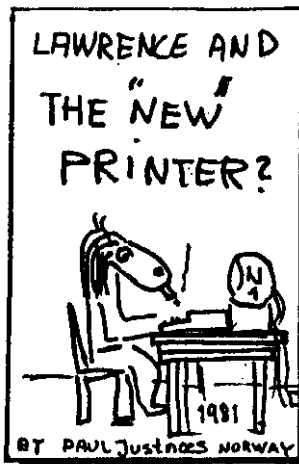
5, Wensley Road,
Leeds LS7 2LX.
Tel. (0532) 683186

or send Sae for full Catalogue.

Please add V.A.T. @ 15%
+ 55p/order P & P.

SEND FULL DETAILS OF YOUR NASCOM





All characters fictitious. Any relation to any person, living or dead, is purely coincidental and can be arranged for the appropriate fee.