INMC NEWS 80

Issue: 4

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

MAY-SEPTEMBER 1981

● **EVEN BIGGER STILL ISSUE 4:**
(and don't expect another
for a considerable time!)

CONTENTS

£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 on a part-time basis, and it is NOT possible for us to become involved in technical correspondence. Please contact Nascom or your Nascom distributor for this.

The Boss

Chairman's Bit

D. R. Hunt

THE GOOD NEWS

Nascom have been bought by Lucas Logic at Warwick. Yes, that's a bit of Joe Lucas the car parts and aerospace people.

A couple of weeks ago I went up to Warwick wearing my dealer's hat to a dealer's meeting to introduce the new company. We were entertained right royally and allowed to wander over their operation and put our sticky little paws over some of the things they are already doing. They already produce micro based industrial monitoring equipment. Lots of 8085s in pretty boxes talking to DEC PDP11s with hard disks. The idea is to monitor machines in a factory, and allow the system to keep a real-time record of everything that's going on. That way, if anything goes wrong, or is running inefficiently it's possible to back track the past history and see what areas (design, product, materials, management, etc) need attention.

Another thing they are into in a big way is Computer Aided Design. That's using a mainframe to emulate the performance of structures and predict design faults, or make savings through more efficient use of materials and all that. They also have a system for drawing three dimensional shapes and then feeding the data to profiling machines to make anything from pretty shaped coffee pots to car body panel tools. These last two prove that Lucas has considerable expertise in software in general and computer graphics in particular. Perhaps there will be some interesting spinoffs in Nascoms direction.

I've had a chat with Mike Hessey, the technical manager, and he says they intend to re-enter production of most of the existing Nascom products as soon as possible (I got the impression that would be fast). My comments in the last Chairman's Bit don't apply, as Lucas Logic already have a design team raring to go, and it shouldn't be anything like as long as I predicted before new ideas and goodies start to appear (although at this stage they're not letting on what).

Anyway, thanks lads for a fascinating day out and for the the insight into what you do. The whole outfit was very impressive and seemed most professional.

NOW, WHAT THE HECK DO WE DO??

When I wrote the last Chairman's bit, I think we were all resigned to the odds on probability that Nascom would be no more by the middle of May. Which left this newsletter well and truly up a gum tree. You see, during the decline of Nascom's fortunes, our membership has declined at much the same rate. Not new members, they are arriving as usual, but the re-subscription rate has dropped alarmingly, and the new members gained from the fewer Nascoms being sold have not stopped the decline. The nett result is that membership is on the slide. When the first news of a purchaser for Nascom broke, renewals increased with it, but since, down we go again.

Now we recognise there are three reasons for this, although the affect of each is difficult to judge. Firstly, there is the irregular appearance of this newsletter. We have always said that there should be six issues a year, but we've not yet quite reached our target. In our defence we will add that it was envisaged that the newsletter would be about 20 to 30 pages, whereas, the most recent newsletters have averaged 54 pages, so we reckon you've still had your money's worth. Also, of late the main preparers of the newsletter have been less busy (Speak for yourself! -Ed.) so the last two issues and this are more or less up to schedule. Secondly, there have been the ups and downs at Nascom. This has been seen to definitely affect membership. Lastly, there have been direct sales of the newsletter, in fact the print runs for the newsletter are bigger than ever, but the majority go to the retailers. Now our retail price for the newsletter has been 1.00, and it doesn't take much arithmetic to decide that an average of, say, four issues a year bought from your retailer will cost you

4.00, whilst re-susbcription will cost you 6.00 (even though for 6.00, you will be sure of getting one as this also covers the postage and subscription administration). Well we're going to stop that one, from now on the retail price in your local dealers is going to be 1.50 (and we're going to charge the dealers more for it), so that means a greater incentive to you to resubsribe, and that INMC80 can probably continue for some time without increasing the subscription rate.

With the imminent demise of Nascom in sight, we felt that to keep this newsletter going, it would be neccessary to 'latch on to' some other product that would provide the membership which in turn would provide the income to allow us to continue. You see, although this newsletter is NOT a commercial vehicle for the benefit of any manufacturer, without the support of manufacturers in producing products which they sell, and you buy and then want to know more about, we can not exist. To this end, in the last but one issue it was mooted that as there are more Nasbus compatible products around than those that originate from Nascom, we ought to become less Nascom orientated and more Nasbus orientated. We asked you for your views, and out of a membership of about 2,000, and judging by the numbers of newsletters sold in all, a readership well in excess of 4,000, we received no less that 4 replies. Staggering isn't it, 0.1% replied!!!! Does that mean that the other 3996 readers don't care what we do, or is the lack of positive action to be taken as disapproval, or that the whole question is so obvious that it doesn't require comment, or what? I might add that those four letters were all in favour. (Dr. Dark seems to be in favour as well, from his comments in the last Dr. Dark's Diary.)

All this leads up to the quandary we are now in. Nascom have had a last minute reprieve and, at the same time, what amounts to a new computer based on a similar bus structure is soon to be launched. The software for the new computer will be derived from much of the existing Nascom software. No it's not a rip-off, most of Nascom's software is either 'public domain' or privately owned by individuals not in any way connected with Nascom. These guys saw their products about .o die with Nascom and have done the sensible thing of adapting their software to the new machine, in many cases with considerable enhancements.

On Nascoms' death we intended to switch our allegience to the bus, and so by implication to the new computer, which with it's close adherence to existing Nascom design conventions would allow us to continue, to the benefit of our Nascom readership, and of course, to owners of the new machine.

Now, what are we to do. Support Nascom and Nascom products only? Or support the bus, and by so doing, support Nascom and anything else that comes along?

It is our belief that to support the bus would be the best course. That way we will be able to keep our readership informed of developments on all fronts. But it is up to you to let us know. We wish to be guided by our members. PLEASE PLEASE PLEASE drop us a line. Make it simple, Nascom only or the bus? We expect to receive hundreds of letters (not just four). If, you feel this matter should be debated, then write to the letters column. We don't intend taking any decisions until the issue after next, so if we've stirred up a hornets nest (I personally hope so) we'll publish the letters in the next issue and let the membership decide.

In the meantime, in this issue and the next, we will try to keep our usual balance, but so that you know what is coming, we include details of the new machine.

So, to finish up, life is almost back to normal Nascomwise. We look forward to the new developments on the Nascom front (and if they remember to tell us first, we'll pass any news on to the most important Nascom owners, YOU, as fast as we can). In the meantime we want to open the bus debate. So please write, I can promise you your comments will be considered fully, although I doubt if we'll have time to reply to individuals. Don't forget, we need material for this rag, so keep it coming. Until the next time...

Letters to the Ed.

Lucas Logic Ltd
Welton Road
Wedgnock Industrial Estate
Warwick CV34 5PZ

Warwick (0926) 497733

27 May 1981

Dear INMC80,

I've no doubt that you have heard before receiving this newsletter the news about the takeover of Nascom Microcomputers by Lucas Logic Ltd. Many magazines and newspapers carried the story and it was heartening to see how many thought it was worth front page treatment. Obviously we did not buy Nascom just to get into the newspapers but we were struck by the almost universal good wishes bestowed on us for keeping Nascom alive.

Before making our bid for Nascom we evaluated the products thoroughly (helped of course by Nascom users on the staff) and it was obviously too good a computer to let go. This view was supported by the fact that there is such an active users club and so many dealers still supplying parts and service.

So we know we have a good computer, a loyal distributor network and thousands of satisfied customers. Where to now? Well apart from knowing that Nascom had a lot to offer us we felt that we could offer Nascom a vigorous future, not just in its traditional markets but in industry and business. Please don't feel that we are going to concentrate solely on industrial users and ignore the enthusiasts. Here in Warwick we have the capability to do a lot with Nascom, in all directions and without falling flat on our faces because we have stretched ourselves too far.

What is in store then? We don't want to release too much information just yet but there are a number of products which we hope you will see quite soon. First a twin floppy disk system which Nascom had started to work on but receivership had prevented from progressing. There will also be some new boards such as a colour board. We intend that it should be possible to buy a Nascom board ready assembled and also a Nascom assembled in a cabinet. There are some very good enclosures around, the Kenilworth case being particularly suitable; there will still be a need for specific enclosures for particular needs. However to improve the general appeal of the Nascom computer a fully enclosed version is needed. Perhaps the most important development to keep us up in the running will be to ensure that the Nascom computer is compatible with the BBC microcomputing series in January 1982. Launch dates for all these products are still being worked out but we fully intend to have them all available this year.

As you can see we are working hard with Nascom and we intend to remain among the leaders in microcomputing. Obviously we will support INMC80 with information, but what is all this about rival computers in the INMC80 News? We fully support the principle of compatible products and we are about to launch a Nascom approved scheme for the best. But if INMC80 News wishes to back rival products as well then it should rename itself "Practical Computing" or similar appropriate name. Give us a chance Mr Chairman and if then we foul it up you can include every widgit that you fancy.

Having got that off my chest I must say that I am pleased to write my first letter to the INMC80 News and I hope that it will not be the last one. We will welcome your letters too; Ken Jones our Sales Manager would like to hear from you.

John N S Deane
General Manager

Nascom Rules !

Congratulations on the latest issues of the Club Newsletter. They were as usual very interesting. I was particularly interested in the `System Software' items such as `Load and Tab', `VARPRO', `CC.COM' and `GLOBAL.CRT', as it is from this type of item that one can get a much better understanding of the insides of NAS-SYS, BASIC, CP/M etc.

I am very glad, having recently had short encounters with a PET, a TANDY, and also with an Apple, to say nothing of a 6800 and an 8085 development system, that I chose (quite by chance at the time), Nascom and the Z80.

Since the Nascom 1, in early 1979, was just about the only available choice for a cheap single board system, the choice was largely decided for me.

I have now graduated to a Nascom 2 with full DOS/CP/M available and it only remains for me to add a colour board and an 80x25 screen format to make the system complete.

I do have one `moan'. It relates to publicity for NASCOM and INMC80.

Many of the short `What's Available' and `Buyers Guide' items in the Computing Magazines are hopelessly inaccurate when it comes to Nascom products and compatible accessories from other sources. It almost seems to me as if P.C. have a bias against NASCOM, and PCW are little better.

Similarly, the listings of Local and National Clubs have often omitted INMC80, although I see that the club is listed in this months' P.C.

Please keep up the good work. I only wish I lived nearer so that I could give more than verbal assistance.

Best wishes for continued success.

C. Bowden, Truro, Cornwall.

Just a few points.

Many thanks for the effort put into INMC80 which continues to be a mine of useful hints and tips and also provides a `clearing house' for the discussion of certain well-known (to some) bugs in NASCOM hardware.

For the amusement of all readers of INMC80 please mention the following misprint on P.123 of William Barden's "Z80 Microcomputer Handbook". Line 5 refers to "... the eternal interrupt..."!! (Is this what sends the CPU to that great mainframe in the sky!).

Regarding David Parkinson's request for comments on documentation, may I suggest a compromise in the amount of detail provided with a program? As was stated a complete source code listing is liable to be expensive but how about:-

(a) Indicating major sections and/or subroutines by memory address;

(b) indicating also by memory address the positions of Code, Tables, Workspace and Message Text so that disassembly may be carried out without too much Sherlock Holmes work;

(c) noting if any Monitor pointers or reflections have been modified.

A.J.Fry, Portsmouth, Hants.

Radio Hams

I think the INMC80 News is an extraordinary effort reflecting much enthusiasm and one hell of a lot of very hard work and I appreciate it enough to join the chorus for more issues, more often. No, I'm sorry, I've nothing to contibute (yet) but I would ask if you could put in a few lines stating that being a Radio Amateur (GM3TYS) and have an ICOM IC245E VHF Transceiver, I would like to contact any enthusiast who may have succeeded in interfacing an IC 245E/IC211 with his Nascom 2. My intention is to do this and eventually write a program allowing frequency control, scan, search, logging and RTTY/C.W. communication.

Secondly, my own system is a Nascom 2 with two 32K "Ram A" boards (full 4MHz) which I would like to add a ROM board to. It is housed within the Powertran case which I can recommend as very excellent value and easily modified to take the NASCOM keyboard with the main and memory boards behind and with the bus connectors to the left and the P.S.U. to the right. A suitable cooling (miniature) fan is badly needed though. Being steel and aluminium it is well suited to Amateur Radio use (for RFI shielding) and it will support any weight of equipment on top.

Thanks and good wishes.

I.G. Drysdale, 104 Mosside Drive
Portlethen, Aberdeen, AB1 4QY.

Delta Capacitor?

Just when I was feeling the need of communication - all that came was a tax return, then INMC80 News. Joy! After the first read, I wonder if anyone can tell me whether a "delta capacitor" is the same as an "interference suppression capacitor" Maplin P.62. &/or P.176. a "mains transient suppressor"? My set-up is Nascom 2 built with much help from Interface. A secondhand Digital Decwriter (no descenders) works faultlessly with (RS232) no special interface: it was bought from Electronic Brokers, who were very expert and helpful. Like the rest of us I suspect, what appeals particularly about the Nascom is that it is so flexible, and there's lots I can do to make mine work for me, yet, as I learn how!

Thank you, and Best Wishes

Laurence Fisher, Canterbury, Kent.

PL1 Error.

Like Mr. D. Ritchie (P.14 INMC80 News No. 3) I have had trouble adding PIOs to a NASCOM 1 and was interested in his letter and thought that this was the solution. Alas no. It took several more days to solve the puzzle of why all signals looked OK on scope but the circuit would not work.

I discovered that on my NASCOM board the data lines on PL1 are not as in the circuit diagram and hence when, for example, OFH was written to the PIO to put it in the output mode it received some other code.

The correct connections are as follows

D0 - PL1/1　　D1 - PL1/2　　D2 - PL1/4
D3 - PL1/6　　D4 - PL1/5　　D5 - PL1/3
D6 - PL1/7　　D7 - PL1/8

T.Bailie, Co. Antrim, N.Ireland.

Anyone got a Creed?

I should be most grateful if you could let me know of any members who may have successfully interfaced a Creed 7 series printer with a Nascom 2.

May I also thank you for the work you do in producing the most helpful and informative "News".

C.D. Macmillan, 7 Trentholme Drive,
York, YO2 2DF.

Cartridge Drive?

I was very interested in the article on p.49 of INMC80-3 concerning the cartridge drive. Do you know where I can get more information about the deck?

M.Millington, Edinburgh.

IO Systems Ltd

I read with amazement the review of our high resolution graphics board in your Feb/April issue by your reader Richard Bateman. What surprised me most was the totally unprofessional way in which the review was performed without checking the facts, as the review contained many inaccuracies which I would like to set right.

(i) Apart from the company name (IO not I/O), the address is wrong.
(ii) The display is NOT 3-dimensional!
(iii) The component count and manual size are both incorrect.
(iv) An explanation of how it works IS included.
(v) NO irreversible modifications have to be made to the board.
(vi) The number of connections required is incorrect.
(vii) The board is compact and WILL fit anywhere.
(viii) NO system RAM is lost as it is NOT dedicated to graphics (It may be used as normal when graphics mode is not selected.).
(ix) Overpriced... Well in judging this may I point out that the board produces a FULLY bit mapped high resolution graphics display (and is NOT a character generator), is fully built and tested and as such compares very favourably with any other graphics product for this machine on the market.

L. J. Noble (Director IO Systems Ltd), 6 Laleham Avenue, Mill Hill, London, NW7 3HL.

(Please read the review again - you have missed several key words. - Ed.)

Educational Software

Congratulations on an excellent magazine, each issue seems even better than the one before. David Hunt's "Teach Yourself Z80" is really good. I like his light hearted style of writing and at last Z80 language is starting to make sense. Keep up the good work.

I'm a teacher and use my Nascom 2 at a local Primary School. Children between the ages of 8 and 11 use various C.A.L. programs without any problems and they think a great deal of my Nascom.

Can I appeal through you to contact other Nascom educational users. I'd like to form a group of people that write and use Nascom Educational software.

Because of the time it takes to write and debug programs not much educational software is around and I believe that if Nascom educational users could get together to `swop' programs then it would be to the benefit of all.

Many educational authorities recommend either the RMZ380 because of its reliability or the PET because of its range of software and relatively low price.

Neither machine, I believe, is as good as the Nascom and price-wise the Nascom is or should be in a competitive position. Providing we educational users can provide a range of software then the Nascom could be used in schools a great deal more than at present. Without such software I imagine sales would be at a minimum despite the advantages of the machine itself.

A user club specifically aimed at the educational user would benefit all concerned as I believe the `380' is much too expensive for most schools and in addition their new system will likely take it out of the thoughts of many educational users because of the high costs.

The Pets are relatively cheap but are getting an increasingly bad name for reliability in the educational sector. Can Nascom take their place? Perhaps if Nascoms problems were sorted out quickly then Nascom might make inroads into the educational sector.

Anyway enough of that, I have a couple of problems which despite much book and magazine searching haven't been resolved.

The first is the INP function, would it be possible to write a section on how this works and when we use it. I've found it in a Computing Today program and cannot fathom out how it works at all.

The second problem is that of a timer to indicate how long a program has run.

To give an example - lets say that we wished to time how long a pupil took to do 'say 10 sums'. How could we do this? The PET has a time function but such a function is not available on our Basic. Is there a machine code sub-routine to do this - such a routine would be extremely useful to me.

Once again thanks for a great machine - hope it keeps going for many years.

W D Cooper, 476 Denton Road, Newcastle upon Tyne. NE15 7HE.

Funny Tapes?

As Nascom Dealers we frequently encounter customers with a low opinion of cassette tape as a storage medium despite us knowing that the hardware is totally satisfactory. Such customers invariable ignore what we now believe to be the true cause - sub-standard cassette tapes! In common with other Dealers we sell blank C10 or C12 cassette tapes believing them to be "screened against drop-outs" and therefore suitable for the recording of digital data. After trying the wares of many suppliers of "screened" tapes (this includes a number of well-known "branded" products) we have now come to the conclusion that if, indeed, they are tested for drop-outs, then the test criteria are totally inadequate. We name no names because it seems that all suppliers offer the same (abysmally low) standard.

Among problems that we have so far encountered are:-

Errors because the tape gets creased by most normal cassette recorders.

Errors because over-recording does not erase the old data.

Errors because a tape is read fairly frequently and wears out very quickly.

No (yes NO!) oxide layer on the tape. (It took a long time trying to decide if this was a "Read" error or a "Write" error!)

When asked, suppliers invariably say that since no other customers have problems, "it must be you" (does this mean all other customers are using low baud rates such as that used by TRS80 etc. and can therefore be supplied with low quality tapes without repercussion?)

In view of this widespread problem, have any of your readers found a source of supply that is always reliable?

Richard S. Marshall, Chief Engineer, Business & Leisure Micro Computers.

Classified

1 Nascom 3A PSU built - 25.00 ono
1 5" wide module for Vero frame - 5.00 ono.
1 16K RAM A board built - 90.00 ono
'Phone Kevin on Aberdeen (0224) 36160.

Nascom 2. Nas-sys, Nas-dis, Debug, Zeap, Graphics, 32K Basic, Basic Toolkit, Word processor (Wordease?), Music Box, 12/2400 Baud, all in Eprom. V.D.U. and cassette, approx. 40 tapes, all documentation, Epson TX-80B printer. Around 1,000.00 'phone 0924 496337 or 0924 490127.

Licon Switches with blank caps to update Nascom 1 keyboard 1.65 each or 15.65 for 10 incl VAT + 35p per order P & P (see INMC80 Issue 2 and Liverpool Software Gazette for connections). Chiatronix Ltd., 22 St. Michaels Avenue, Houghton Regis, Dunstable, Beds. 'Phone 0582 61697

Space Invaders

After reading INMC80 Issue 3 I realised I was in a position to help many people.

The Space Invaders game published in Issue 2 is the same one that a friend of mine bought. The program needs a little explanation:

1. The graphics option that is incorporated is for use with the William Stuart Graphics System although gives a reasonable display with NAS-GRA.

2. The program will work with any Nascom monitor - T2, T4, B-Bug, NAS-SYS 1. (NAS-SYS 3?) (Yes - Ed.)

3. There is a `Pause' command, operated by hitting the `P' Key. This temporarily stops the program so you can answer the telephone etc. Hit any key to resume play.

The answers to W. Squires' letter are:-

1. The program was written to run at 2MHz not the normal NASCOM 2 speed of 4MHz.

2. There are 3 places where the program has to be changed for different keys. The first two places display and check for those keys not available as the fire key. The bytes at 1058H and 105CH must be changed to check for the ACSII values of these keys. Lastly the codes at 1785H must be altered. This piece of code scans the keyboard, then looks for various keys in the keyboard map at 0C01H. Using the routine at the end of this letter it is possible to work out which bits of which bytes are altered by the keys you have chosen.

e.g. To substitute the `A' and `X' keys for `↓' (CD) and `→' (CR) respectively.

```
M1040   N/L
  1040   0B/1042   N/L   (HEX VALUE FOR A `↓')
  1042   09/1058   N/L   ( "    "    "  " `→')
  1058   14/105C   N/L   ( "    "    "  CURSOR DOWN)
  105C   12/1785   N/L   ( "    "    "      "  RIGHT)
  1785   3A 04 0C/178B N/L (CURSOR DOWN KEY ALTERS 0C04H
  178B   FE 40/1790 N/L  (bit 6.)
  1790   3A 05 0C   N/L  (CURSOR RIGHT KEY ALTERS 0C05 H
  1793   FE 40.          (bit 6.)
```

As regards David Parkinsons letter, I feel that the documentation on the Nascom 2 is very good indeed, particularly the invaluable listing of NAS-SYS. The documentation on his toolkit is more than adequate and I had no problems learning to use it.

The article entitled "WHERE" must have been written by a beginner to produce such a piece of code. Try the following:-

```
      RCAL LOCN       D7 00   (MONITOR RTN).
LOCN: POP  HL         E1
```

The address of "LOCN" is popped into HL.

Please warn NASCOM owners about writing BASIC programs with variable names longer than 2 letters. Although it is acceptable to do so, MICROSOFT BASIC V4.7 only stores the first 2 characters of the variable names and therefore cannot distinguish between 2 different variables having the same first 2 characters. This also applies to strings.

Finally could you print your prices for advertising in INMC80, as a friend and I have started a software firm to seel NASCOM Software for the NASCOM 2, and we would like to advertise in INMC80.

A. READ, Chelmsford, Essex.

(Rates are 100.00 per full page, part pages are charged proportionately - Ed.)

KBD ROUTINE.

```
EF  0C  00  21  2A  0C  36  0A
2B  36  0A  EF  56  41  4C  00
06  08  3E  80  F5  34  34  34
DF  68  F1  0F  10  F6  36  8A
23  35  2B  EF  42  49  54  20
20  20  00  06  08  78  3D  DF
7A  34  34  34  34  10  F7  23
35  2B  36  0E  E5  DF  61  E1
3E  0C  DF  68  0E  01  06  09
11  00  0C  79  DF  68  7E  C6
7E  77  1A  DF  68  7E  D6  7E
77  0C  13  10  EE  18  DB  00
```

This routine can be located at any location.

Multiple Processors

A couple of weeks ago, I received my copy of INMC 80/3. A week later I had built my THIRD Nascom 2. Can anyone beat that? At the time of receiving Issue 3, I did not have a Nascom, but have been sorely tempted each time a new INMC80 came through the post, or I got a 'phone call from the two or three other people I know who have NASCOM 2's.

My first one was built in March '80, followed by the second one in July '80 for use at work. I had to sell my first one in November '80 to buy a new car, but since then I have been planning to buy another. I am pleased to say that the third one is up and running. Two points though I would mention (1) could not NASCOM manage to print pages in the documentation squarely on the paper; on all three books I have had a number of pages are printed at an angle. (2) I feel, along with others, that NASCOMS should be repaired free of charge within say 90 days of kit purchase if the fault of non-operation is due to faulty chips or a faulty board. There appears to be a Nascom practise of charging regardless. My first model was fault free, the second had a board fault, and my third just completed had two faulty sockets, one on the Nas 2 main board and one on the Ram B board.

It is good to hear that Nascom have at last been taken over, lets hope some new products will see the light of day. I very much look forward to INMC80 newsletters dropping through my letter box, keep up the good work.

R. Scadden, Stratford-on-Avon, Warwicks.

(Faulty Nascoms should really be returned to your distributor, not to Nascom direct. Before buying check what your distributor's attitude will be. I think that you will find most willing to rectify faulty components FOC, if not then go elsewhere - Ed.)

WHITEWASH !

The mag is excellent. I find something valuable in it every issue. However, the article by Rory Johnston was reminiscent, in its attempt to whitewash the computer industry of moral and political responsibility for what it produces, of the blindness of atomic scientists to the consequences of their research forty-five years ago.

Most of the exaggeration of computer power is due to the industry's overselling of itself. It is not due to the supposedly ignorant laypeople who have, quite simply, been conned.

Furthermore, whistle-blowing over Big Brother is not in the least bit out of place but a vital part of the campaign for the protection and reconquest of human rights. He claims that voice recognition technology can not yet automatically convert telephone conversations into printed text. Yes, he's probably right - although even when it does become possible such technology would certainly remain secret for a very long time unless we can prevent it. What he fails to mention, however, is that computers can be "taught" to recognise keywords in a given person's speech - unreliably perhaps, but sufficient to make things a lot easier for Big Brother - such that tapped phones can signal to a listener when the conversation is getting "hot".

Unfortunately, human rights and principles do not remain the same - as he claims - they have to be fought for and protected. Any child watching the cavortings of our politicians and multinationals can see that they will do anything they can get away with to keep power and profit in their hands. Computers can and will be used to help them do just that.

On a more technical note, I wonder if you have come across A.S. Watkin's descriptive breakdown of the Nascom BASIC. I have only had it a couple of weeks but found it amazingly useful. It does, however, contain several minor errors and a couple of "howlers" which I discovered while using it to disassemble (by hand) some key sections. I am trying to get a list of these together. Would you be interested in it together with a review, perhaps?

M. York, London.

(Yes - do write the article - Ed.)

Machine Code Programming

I have just read in the local paper that Nascom have been bought by Lucas Logic. I hope that this will provide the necessary funds to continue to further success that this excellent product deserves.

Now yet another word on machine code programming. David Lorde's correspondence (INMC80 Issue 2) on the subject, I think, rings very true. I suppose that for a few lucky people it will come as second nature. However, for most of us there is no easy way to learn except for good hard slog. Articles such as David Hunts do make the slog less `sloggable' and there is plently of room for such useful material. I've had my Nascom 2 (Yawn - not another I've had my Superthunderstingcom for etc., etc., - again) (Yes) for almost a year and knew a little about BASIC, but absolutely zilch about machine code. I began by converting some programms out of previous INMC newsletters that were written for the T4 monitor to run with Nas-Sys. (with a certain amount of success).

When the Space Invaders programm arrived all hell was let loose (well, the odd hobgoblin here and there). After several months of hair-pulling, teeth-gnashing and yelling at the cat (which probably now knows more about machine code programming than I'll ever know, but has more sense and lies in front of the fire) I finally managed a working understanding of the programme. I've written several routines to control the speed of the programm (which was rather breathtaking) and to stop a lot of the on screen flashing.

So, I've still got plenty to learn, but I've managed to untangle some of the mystery. While fiddling about with the `Invaders' programme (which is excellent) what did amaze me is the speed that the computer works at. It seems, with that program, the problem is in slowing the computer down. A correspondent in your INMC80/3 mentions a preferable change in the use of the keys to control the movement. The method of mapping the keyboard is a mystery to me also. Any answers?

A little hint that probably everyone else has discovered but something I hit on a short time ago. When using the M command to modify memory locations the cursor may be taken back up screen and a previous location changed i.e. just as in editing lines in BASIC. This can occur across several completely different blocks of memory that may be on screen.

Although people may scoff, I am mostly interested in the graphics side of computing (and thus games). So any addition to the graphics capability (hardware or software) I find of interest. A review of the Bits and PC's PCG perhaps?

Enough of my waffle, the cats just writing a Basic interpreter,

Yours ADD.OR.INC.ly (moans from background)

D. Hirst, Birmingham.

MISTAKE !

There is a BOOB in INMC80/3. In the drawing of the Reset Jump circuit, pins 1 and 15 of the 74LS257 should be interchanged. The circuit works very nicely, and I have it running. I also changed my clock link to 4MHz, with great success.

R. O'Farrell, Co. Wicklow, Ireland.

Documentation

I am writing to reply to David Pasrkinsons letter in Issue 3.

Dear David, Your letter on documentation was very stimulating and prompts me to answer. Rather than take up your numbered questions I will get straight to the point.

I would like to see source listings included and would be more than happy to pay the increased costs. This is for two reasons:-

1) Having the source code enables me to easily tailor the program to my needs.

2) I have learnt much by studying program listings - particularly yours! (Revas is an excellent program.) I have just purchased a 1K program from a fellow in Glasgow that interfaces to a Selectric printer. It came with a commented source listing and the cassette tape also contained a Zeap file of the source code! This is perfect. I couldn't ask for more. I wish other authors would follow suit.

So - please make your toolkit source available as it would make the product twice as valuable as far as I am concerned.

B. Gilchrist, West Sussex.

UNKILLING

Just a note in case anyone has missed it. The magic method in Appendix III of the NASPEN Firmware Manual will not only recover a text from cold start but also from `KILL'.

If you have killed your text by accident, what you do is as follows:

Leave NASPEN by typing `N'.
Type `M101A' new line (backspace over the `n' if it is there)
Type `12 10 2.' new line
Type `E B806' new line (NASPEN warm start)
Press space bar twice.
Step to end of text using only new line and the space bar; don't go further than the last character in your text.
Type `K' then `Y'.
Type `Z' and remove strange character at beginning of text (use `I' and back space). Incidentally if you get the cursor stuck to the left of it just press cursor arrow down.
Replace missing characters at beginning of text and reset line length and page length (1 & 2, 3 & 4).

Hope this saves someone some typing.

P. Copping, Manchester.

Printers and Naspen

Although I have had my Nascom 2 for a year or so now, I have only just made a subscription to the news letter as I wished to wait for it to establish itself and to contain a little more information and articles on the 2 rather than the 1.

I recently got a Centronics 737 printer and with the help of a m/c programming genius, a 74123 and a few bits of wire, built a parallel interface, using, of course, the PIO. (The Centronics 737 is a delight to use and the script is most definitely correspondance quality. The write up in the March 81 issue of Practical computing is very fair and I agree with it.)

Now, not a lot of people know this as its never been published before, (I think), but it is possible to insert printer control codes into Naspen with complete success!

I have read the NASPEN documentation countless times and there are absolutely no clues as to this capability.

Some codes appear to be a little tricky, ESCape for example, but if you use the little `i' (insert), it works perfectly. I won't provide a list as working them out for oneself helps to memorise them, but if anyone would like a list, an SAE will do the trick.

For those interested in how Nascoms are housed, mine for the first 8 months of its life was housed in a settee, complete with PSU. The keyboard cable came from within to the keyboard which used to sit on my knees and the whole setup was very domestic but portability was a real problem!

I eventually bought a Ball Miratel Monitor from Electronic Brokers and shovelled the whole lot in. It was a tight squeeze I can tell you. None the less in that tiny cabinet I have the VDU and all its PSUs, Nascom 2 and all its PSUs, a board to give me seperate vert. and horiz. sync which the monitor requires, a tape drive relay board, a sound board (for the dreaded space invaders), a board for the Sargon chess graphics and last but not least a board for the printer interface.

I would like a 48K board as well as the 32K board but that is absolutely out of the question unless I bring the Nasbus outside the Ball Miratel housing but as everything is contained at the moment it is very neat and tidy.

Finally, I have worked professionally with Ap*le, Sh*rp, P*t and the like, and in my opinion, Nascom is streets ahead of any of them!

C.R. Bruce, Farnham, Surrey.

Space Sounds.

Space invader freak's may find this worth a try, it's a bit messy and slows the action down quite a bit, but visitors seem to prefer the game with sound.

The only hardware mod. is to hook a speaker onto port 0, bit 5, (that's IC 24 pin 15 for Nascom 2), via a suitable buffer of course.

I read somewhere that someone wanted an approximation to PI better than 22 over 7, how about 355 over 113?

R. Cutler, Birmingham.

```
   ZEAP Z80 Assembler - Source Listing

                   0010 ; * * * * * * * * * * * * * * *
                   0020 ; * SOUND for SPACE INVADERS. *
                   0030 ; * RAY CUTLER.       20-3-81 *
                   0040 ; * * * * * * * * * * * * * * *

                   0060 ; Modify the original prog. as follows :-
                   0070 ;  CHANGE        TO
                   0080 ; £1437-39   CD 00 0D - Hit by bombs
                   0090 ; £1695-97   CD C0 0C - Bombs hit shield
                   0100 ; £16D5-D7   CD E0 0C - Invaders moving
                   0110 ; £17B9-BB   CD A0 0C - Shoot
                   0120 ; £1A24-26   CD 80 0C - Hit target
   0C80            0130         ORG  £0C80
```

```
0C80 F5           0150 CALL1  PUSH AF
0C81 C5           0160        PUSH BC
0C82 062F         0170        LD   B, £2F
0C84 3E20         0180 LP1    LD   A, £20
0C86 D300         0190        OUT  (0), A
0C88 3E0B         0200        LD   A, £0B
0C8A FF           0210        RST  £38
0C8B D300         0220        OUT  (0), A
0C8D 3E15         0230        LD   A, £15
0C8F FF           0240        RST  £38
0C90 10F2         0250        DJNZ LP1
0C92 C1           0260        POP  BC
0C93 F1           0270        POP  AF
0C94 CD821A       0280        CALL £1A82
0C97 C9           0290        RET
0C98 00000000     0300        DEFB 0 0 0 0 0 0 0 0; FILL
     00000000

0CA0 F5           0320 CALL2  PUSH AF
0CA1 C5           0330        PUSH BC
0CA2 0664         0340        LD   B, 64H
0CA4 3E20         0350 LP2    LD   A, £20
0CA6 D300         0360        OUT  (0), A
0CA8 3E0B         0370        LD   A, £0B
0CAA FF           0380        RST  £38
0CAB D300         0390        OUT  (0), A
0CAD 3E0B         0400        LD   A, £0B
0CAF FF           0410        RST  £38
0CB0 10F2         0420        DJNZ LP2
0CB2 C1           0430        POP  BC
0CB3 F1           0440        POP  AF
0CB4 CDB919       0450        CALL £19B9
0CB7 C9           0460        RET
0CB8 00000000     0470        DEFB 0 0 0 0 0 0 0 0; FILL
     00000000

0CC0 F5           0490 CALL3  PUSH AF
0CC1 C5           0500        PUSH BC
0CC2 060C         0510        LD   B, £0C
0CC4 3E20         0520 LP3    LD   A, £20
0CC6 D300         0530        OUT  (0), A
0CC8 3E33         0540        LD   A, £33
0CCA FF           0550        RST  £38
0CCB D300         0560        OUT  (0), A
0CCD 3E33         0570        LD   A, £33
0CCF FF           0580        RST  £38
0CD0 10F2         0590        DJNZ LP3
0CD2 C1           0600        POP  BC
0CD3 F1           0610        POP  AF
0CD4 CDB919       0620        CALL £19B9
0CD7 C9           0630        RET
0CD8 00000000     0640        DEFB 0 0 0 0 0 0 0 0; FILL
     00000000

0CE0 F5           0660 CALL4  PUSH AF
0CE1 C5           0670        PUSH BC
0CE2 0601         0680        LD   B, 01
0CE4 3E20         0690 LP4    LD   A, £20
0CE6 D300         0700        OUT  (0), A
0CE8 3E33         0710        LD   A, £33
0CEA FF           0720        RST  £38
0CEB D300         0730        OUT  (0), A
0CED 3E15         0740        LD   A, £15
0CEF FF           0750        RST  £38
0CF0 10F2         0760        DJNZ LP4
```

```
0CF2 C1         0770        POP  BC
0CF3 F1         0780        POP  AF
0CF4 CDE619     0790        CALL £19E6
0CF7 C9         0800        RET
0CF8 00000000 0810          DEFB 0 0 0 0 0 0 0 0; FILL
     00000000

0D00 F5         0830 CALL5  PUSH AF
0D01 C5         0840        PUSH BC
0D02 0664       0850        LD   B, 100
0D04 3E20       0860 LP5    LD   A, £20
0D06 D300       0870        OUT  (0), A
0D08 3E63       0880        LD   A, £63
0D0A FF         0890        RST  £38
0D0B D300       0900        OUT  (0), A
0D0D 3E63       0910        LD   A, £63
0D0F FF         0920        RST  £38
0D10 10F2       0930        DJNZ LP5
0D12 C1         0940        POP  BC
0D13 F1         0950        POP  AF
0D14 CDE618     0960        CALL £18E6
0D17 C9         0970        RET
                0980 ; If you feel that "PLAYER" flashes too
                0990 ; many times, change £1CCA
                1000 ; After loading and modifying, write a
                1010 ; tape from £0C80 to £2000.
```

CLUBS

Once again a list of NASCOM USERS anxious to communicate with others in their respective areas.

KENT - I would like to help form a club/exchange information! Contact L.S. Fisher, 21, Manwood Avenue, St.Stephens, Canterbury, CT2 7AH.

SCOTLAND (West of) - I would like to make contact with other Nascom users. Mr. Tom Donald, 2 Glasgow Street, Glasgow, G1Z 8JN. (041-334 8931).

N.IRELAND - Help me set up a club! write to Mr. R. A. Lough, 92 Station Road, Greenisland, Co. Antrim.

ESSEX (Ilford). Anyone willing to help form a local group please contact Mr. S. P. Lee, 37 Malvern Drive, ILFORD, Essex, IG3 9DP.

BELFAST, N. Ireland - my friend and I are keen to form a local users club. Anyone interested please contact Mr. R. T. Martin, 30 McCaughan Park, Belfast, BT6 9QJ.

HELP NEEDED - Any Nascomaniacs in my area? I'm awful lonely - and struggling! Mr. D. Platt, 4 Royal Oak Close, Machen, NEWPORT, Gwent, NP1 8SP, Mid Glamorgan, S. Wales.

HELP OFFERED - "I work in the engineering department of Plessey where there are several NASCOM 2 owners, (plus ZX81's, Tandy's etc of the uneducated). If is not beyond the wit of man that if you need someone to help out with answers to hardware and system questions I will have a go!" Mr. P. R. Verity, 11 Liberty Lane, Addlestone, Surrey, KT15 1LU.

A NEW COMPUTER MAGAZINE. The first issue of "YOUR COMPUTER" contained articles on Basic language, kit building techniques, a page to answer technical queries plus games etc etc.

Computer Club will be a regular section of the magazine where members of local computer clubs can write advising new discoveries, special events/projects, find general advice about how to start and run a club.

Let's get them interested in Nascom's! - submit your progs and ideas to: Your Computer, Quadrant House, The Quadrant, Sutton, Surrey SM25AS.

Dateline Algeria

from Richard Bateman

Dear INMC80,

BLACK FLASHES

I agree with S.C. Willmott that the black flashes are irritating. To get rid of them should be an easy matter of connecting the BUSREQ line to the video blanking, so that the Z80 does not access the video memory during the display time. This would cause the Z80 to work a little slower than it does at the moment, but it is only like having a 2MHz clock. Note, it could still suffer from the dreaded black flasher , but only if the Z80 was making an access to the screen at that moment. The loss of speed can be calculated as:

$$\begin{aligned} loss\% &= (1-(16*14*48)/(320*64)) \\ &= 1-(\text{time displaying})/(\text{total frame time}) \\ &= \text{approx } 50\% \end{aligned}$$

This would be equivalent to a clock rate of 2MHz. The lines must be tied FALSE so that they don't do funny things while the bus is idle. I have not tried this but it should work.

BASIC TOOLKIT (Parkinson)

With regard to Mr Parkinson's documentation, I think that he has got it about right, except that as his toolkit tends to be disabled by any sort of reset, and has to be reloaded, (this is a serious drawback), a mention of how to effect a warm start would be a help. (You ain't read the instructions, execute an address three bytes less than the top address used by `Toolkit', and it'll warm start. Ed.) On the toolkit, a few unwelcome suggestions are as follows. The find command could be made to translate the codes by using the BASIC itself. If the string was entered as a line, say line 0, then let the Find do its comparison. After the command, the line could be deleted. Long lines could be handled by looking for a "\" as the first character, and then including the previous line as the start of the basic line, using a modified INLIN. It is possible to keep making suggestions, but then it gets a bit out of hand. The renumber and cross reference are worth it alone, with the others as a bonus. The find needs to use reserved words, and a warm start should be enough to make an issue 2.

NEW RAM64 BOARD and THE SON OF PAGE MODE

I have installed a RAM64 board onto my N2, which means that I now have 96k on line. The page mode is installed on both RAM B and the RAM64. The RAM64 worked first time and without problems, but the RAM B started giving problems for the first time in a year. I have not tracked down the problem yet, but sometimes it works and some times it doesn't. One of those! Since I only have a 3A psu, it may be loss of power, but I will try putting heavy wires from the psu to the bus.

An interesting problem arises when I try to use BASIC. The BASIC initialises Ok but as soon as I hit a key, it crashes. Particularly when I have the Toolkit loaded. It could be something to do with the IO select line used by the page mode hardware. It is surprising that Nascom used 128 ports! (What? - Ed.)

A weakness of the N2 memory map means that the workspace of a lot of the firmware is not, and cannot be put onto a page. This is a serious draw back to the system as designed. Another drawback is the inability to use DMA to the closed page. When (if) I design my DMA disk interface to run with my (also yet to be designed) super disk system, I won't be able to swap closed pages to provide a disk cache memory. I can't think what I would use it for.............

That aside, the page mode is a bit limited in its use because the system turns off whole boards at a time. That means it is necessary to copy the contents across from one page to the other of portions that are common. Also, addresses occupied by firmware can't be got at, such as the bottom 4k, the top 8k. A slight modification to the board would allow a substantial increase in flexibility. If each card had its own port number, and the 8 bit latch was used to enable either blocks of memory (in 16K blocks of seperate read and write, or 8K blocks of R/W memory, working on the memory decode signals), or to enable banks of memory. In this way, a single card could have all 4 banks addressed to the same 16k, and the software would only enable one at a time. This would allow programs to use this area as fast disk space. Or as "protected" space for saving back-up copies of programs you have just written but not tested (you know ... the ones that write all over memory for you in less time than it takes to hit reset).

SYSTEM GROWTH OUTSTRIPS PSU

My intention is to go over to a complete RAM system with a disk and an operating system. CP/M would be nice but I think the investment too high at the moment. I would have to throw away all my current software, unless someone wants to make an offer for my firmware. (Cost was about 200.00+VAT).

My latest addition has been a 12inch monitor which replaces a borrowed B/W TV I used to think was quite good until I got this. I am now thinking of shoe-horning my N2 into the box, as there is just room if ..... the whole problem is the noisy fan that I must have to keep the smell of burning off the 3A psu which is giving its all. If I build a new 5A psu on the back panel then maybe the rest will fit inside and I can throw away the noisy fan.

One mod I would like to see is an 80 by 24 line screen so that I can see what I am getting with NAS-PEN. (Just read this issue, but the snag is a new NAS-PEN. Ed.)

Rename the bus, NASty-BUSiness. You could have it on your adds - writ large, so to speak.

Keep up the good work, it wouldn't be the same without INMC80 coming out with so many goodies.

Richard Bateman.
Algiers, Algeria.

INMC80

INMC80 SUBSCRIPTIONS

I would like to subscribe to INMC80 News for 12 months, starting with the INMC80-5 issue. I enclose my cheque / postal order / international money order (but NOT a French cheque) payable to "INMC80", value:

UK Subscriptions:	6.00
Overseas Subscriptions:	7.50

Name :

Address :

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

PLEASE NOTE: This address has kindly been `loaned' to INMC80 as a postbox. It is NOT possible to communicate with INMC80 by `phoning this address or by calling in person.

PASCAL

HISOFT NASPAS FOR NASCOMS: A REVIEW.

by R. O'Farrell

The HISOFT NASPAS compiler comes on tape in two versions. One runs under HISOFT's own NASMON, the other, with which we are here concerned, runs under Nas-Sys. On the review machine, it is running under Nas-Sys 3, but I see no obstacle to it running under Nas-Sys 1. As supplied, you receive a tape, recorded in CUTS at 300 baud, and a documentation package consisting of a manual of 9 A4 pages to the NASLIN editor supplied as part of the package, a manual of 63 pages to the NASPAS itself, and a three page implementation note describing how to get the package running under Nas-Sys.

The tape loaded first go, with no tape errors. As the program is so long, the tape is recorded on both sides. After loading the first side, which is set up using the `G' command, the tape is turned over and the second side loaded. When loading is finished, the Nascom prompts with a message to

```
E**** XXXX YYYY
```

where the **** address is supplied (43B1H in my copy) on the screen, XXXX marks the start of the compiler proper, and YYYY marks the start of the runtime support routines. The compiler is 27AAH bytes long (just under 10K) and the runtimes are 0B88H bytes long (just under 3K). The implementation note supplied indicates that the compiler should not be located below 1C00H in memory, presumably as it will overwrite itself in the relocation. Normally, the compiler and runtime routines would live at the top of memory, and suggested addresses are supplied in the implementation note. In a 32k system, the compiler lives at 5CCEH, and the runtimes at 8478H. In a 48k system, the addresses are 9CCEH and C478H. The note doesn't give the addresses for a full system, but they are easily calculated to be CCCEH and F478H.

When the relocation has taken place, the Nas-Sys message is displayed, and an execute address for the compiler. In the case of the three memory sizes I have been discussing, these addresses are 5F9EH, 9F9EH, and CF9EH. This address should be carefully noted, as it is the entry point for the compiler at all stages. The relocated compiler can now be recorded on tape, including the runtime support routines. According to the implementation note supplied, the original compiler - that before relocation - can not be dumped to tape.

All set? Then we can enter the compiler. WAIT! The implementation note requires four parameters to be specified along with the execute address - the positions of where the code generated by the compiler is to be placed, the location of the runtime stack, used for variable storage etc., the address from which you would like the compiled code to execute, as opposed to where it resides during compilation (this facility allows the code to be placed on top of the source or even of the compiler itself. NOTE: NOT the runtimes!) and finally the position of the start of the current text file. However, this complication of specifying four parameters is simplified for us by the compiler writer. If these parameters are entered as 0 - NOT left blank!- then the compiler automatically allocates default values. For general messing about, this is quite satisfactory, and allows you the option of specifying your various workspace areas etc. as you need. Do not forget to enter the four zeros seperated by spaces after the execute address. Otherwise very unpleasant things may happen!

The NASLIN editor supplied is very similar to a BASIC line editor, or to the ZEAP editor. Each line has a number, and the editor puts them in their correct place relative to each other.

There are 10 commands. These are:

`C´ to cause compilation
`D´ nn mm to delete lines nn to mm inclusive
`F´ `string´ to find occurances of ´string´, which is delimited by single quotes
`I´ nn mm to cause automatic line numbering of input lines, starting at nn, increment mm. Defaults are 10 and 10
`L´ nn mm to cause lines nn to mm to list
`M´ returns to monitor
`N´ nn mm renumbers the existing program starting at nn, increment mm. Defaults 10 and 10
`P´ puts the program on tape in `generate´ format. The program can be read back into memory from the monitor, and the compiler cold started. Then the `W´ command is used to open the file.
`R´ runs the program
`W´ opens a file after reading t in from tape. It should be used everytime the compiler is restarted with a program in memory, as the compiler cold starts each time, and the W reopens the file.

So much for the editor. It is fairly straightforward, particularly if you have been used to BASIC or ZEAP. One drawback - the line length is limited to 48 chars by Nascom's screen editing facilities, but the compiler can run off sourcefiles written on other types of editor.

Writing a program is straightforward enough with this editor. Having written a program, we ESC to leave the I mode. L to list, P to save to tape (in case of catastrophes - which never, never happen to us! What never? No, never! What never? Well, hardly ever! with acknowledgement to Gilbert and Sullivan). Having a tape of the program, we can now hit C to compile. Naturally there will be errors. The error messages are a display of *ERROR* nn, and an arrow indicating the symbol of the program which caused the error to show up. The error numbers are listed on page 41 of the manual, with a good explanation of each. Hit the E key, and you are returned to the editor. Hit any other key, and compilation proceeds until the next error. Frequently, one error will cause a number of others, so it is a good idea to abort after about four or five. As compilation proceeds, listed alongside the lines is the address where the code produced resides.

Having corrected the errors, and achieved a successful compilation, one is then given the choice of Run/Tape/Editor? Answering this with R causes the program to run, and return you to Nas-Sys. Hitting T causes the object code to be dumped to tape in generate format, and you are returned to Nas-Sys when finished. Hitting E (or any other key) gets you back to the editor.

When the program is running, if the correct options are set, you can interrupt the compiler, and cause it to pause. If during such a pause you hit E, then you are returned to Nas-Sys, and the program aborted. Any other key causes resumption of the program. There are a number of compiler options which are described in the manual. These all default to reasonable values. They are concerned with checking the keyboard for interrupt commands, checking for possible stack overflow, checks on array bounds, checks on arithmetic overflow, and suppression of all lines being listed during compilation unless in error. There is also another option under NASMON, which allows an external printer to be driven. Much to my regret, the author of the compiler has suppressed this in the Nas-Sys implementation, and suggests that one uses the X option instead. This is the only complaint I have! It is possible to obtain listings using the Nas-Sys X and U routines.

In a seperate article I have listed all the major facilities offered by this compiler, so do not intend to repeat them here. Suffice it to say that this is a very powerful implementation of Pascal, albeit not a full Pascal. Compared with 8k BASIC, this is as powerful as a 12k BASIC, if not slightly more. A comparison with BASIC is hardly fair - a bit like comparing chalk with cheese. This compiler compiles fast, and produces fast code. An example: a bubblesort of 255 numbers in machine code is 2 secs, in Pascal 15 secs, and in BASIC 300 secs.(4 MHz machine)

This implementation of Pascal offers the ability to define special types of data, which would allow programs tailored precisely to a problem to be written. It also offers a few extensions to standard Pascal - in particular PEEK and POKE, to allow memory to be accessed and altered. The PEEK and POKE are especially interesing, as they allow TYPEs to be PEEKed and POKEd. This allows a very simple way of moving data around for processing. It would, for example, be possible to define a TYPE CRT=ARRAY[0..1023] OF CHAR, and then to say POKE(#0800,CRT) to change the entire screen at one fell swoop.

The compiler departs from standard Pascal in another few instances. It insists on Strong Typing, which causes the programmer to think a little harder, but cuts down on programming errors. Also available are the ability to recognise HEX integers, and to write them. If you have ever used PEEK and POKE in BASIC, you will know how difficult it is to figure out the decimal equivalents of hex addresses. Not a problem in NASPAS! Also supported are the Nascom Graphics, using the NASGRA ROM, for both N2s and N1s with Econographics. LINE(ON,0,0,44,44) will draw a line from (0,0) to (44,44), diagonally across the screen. The arithmetic is of the accuracy of 8k BASIC - 6/7 digits. The trigonometric functions are not implemented, but these are easily written using the ability of Pascal to support recursive calls. It is possible, using the supplied instructions, to add predfined functions and procedures to the compiler, in much the way that the XTAL BASIC is extendable. The manual supplied is much more complete on this aspect than XTAL BASIC manual. The display of real numbers is always in scientific notation, but again that could be easily altered by a procedure to format output as desired.

I have discussed this compiler with a few other computer users, and we have come to the same conclusion. It is most impressive, and will give an excellent introduction to the capabilities of Pascal.

The manual states categorically that it does not purport to teach Pascal, and refers you to several other books. These are:

"Pascal User Manual and Report", Jensen and Wirth, Springer Verlag
"Pascal, an Intro. to systematic programming", Findlay and Watt,Pitman
"Introduction to Pascal", Welsh and Elder, Prentice Hall

The last two of these are better oriented to the beginner, who has not programmed in high level languages before to any great extent. To them can be added the following two books:

"Programming via Pascal", Rohl and Barrett, Cambridge University Press
"Introduction to Pascal", Zaks, Sybex

The first of these two is my own favorite, but they are both most readable.

This compiler is available on tape from:
Hisoft,
60 Hallam Moor,
Lidem
Swindon SN3 6LS

It costs 35.00. In the manual reference is made to the fact that the runtime routines can be supplied in EPROM. I have no price for this, but am sure that Hisoft will be able to quote.

PASCAL IMPLEMENTATIONS FOR THE NASCOM.

by R. O'Farrell

There are currently available FOUR Pascal compilers for the Nascom. The purpose of this article is to tabulate them against each other, so that prospective purchasers can see how they differ in the facilities offered. It does not purport to be a review, as it is prepared solely from the circulated descriptions available from the suppliers. Prices are correct to the best of my knowledge at time of writing (2nd May 1981), but these should be confirmed by reference to the suppliers, who will no doubt answer any queries. I have only seen two of these implementations, the NASPAS and the INTEGER PASCAL, so can only write from personal experience on these two.

The available compilers are:

****

1. INTEGER PASCAL

Datron Interform Ltd,
2 Abbeydale Road,
Sheffield,
Cost 35.00

CONSTANTS:	-32767 to +32767 Hexintegers %0000 to %FFFF
VARIABLES:	integers:-32767 to +32767 Hexintegers %0000 to %FFFF ARRAY[] of INTEGER characters:ASCII
OPERATORS:	integer: + - OR * DIV MOD AND SHL SHR boolean: = < > <> <= >=
FUNCTIONS:	
PROCEDURES:	value parameters
CONTROLS:	BEGIN - END IF - THEN - ELSE CASE - OF - : - ELSE - END WHILE - DO REPEAT - UNTIL - FOR - TO/DOWNTO - DO CALL(-)

INPUT/OUTPUT:	READ(-)
	WRITE(-)

OTHER: Comments are supported. The MEM[array] allows access to any byte of memory, and the hexintegers allow easy reference to a particular byte of memory. Suffixes in the READ and WRITE statements allow values to be input and output as decimal, hex or ASCII

ACCURACY: Only integers are supported, so this is not a package for number crunching.

****

2. ENERTECH PASCAL 16/C

ENERTECH Ltd.,
32 Gildredge Road,
Eastbourne,
East Sussex BN21 4SH,
Cost 40.00

CONSTANTS:	numerical:ranges +/- 0.1469 E-38 to +/- 0.1701 E+39 character and character string:ASCII

VARIABLES: integer:range +/-65535
real:ranges +/- 0.1469 E-38 to +/- 0.1701 E+39
character:ASCII

OPERATORS: integer: + - * DIV MOD
real: + - * /
real(non-standard):** {exponentiation}
boolean: NOT = <> < > <= >=

FUNCTIONS: integer:ABS SQR
integer-real:ROUND TRUNC
real:ABS SQR SQRT SIN COS ARCTAN LN EXP
real(non-standard):RCPL SIND COSD ARCTAND {degrees}
LOG {to base ten}
integer-char:ORD CHR

PROCEDURES: value and variable parameters

CONTROLS: BEGIN - END
REPEAT - UNTIL
WHILE - DO
FOR - TO/DOWNTO - DO
IF - THEN
IF - THEN - ELSE

INPUT/OUTPUT: READ READLN EOLN
WRITE WRITELN: expressions and character strings

OTHER: Comments are supported, as is recursive syntax.

This package runs in 16k. It includes a system bootstrap, the MAPP 1-4Z floating point package, MAPP extension, Pascal interpreter and compiler and a screen editor. It compiles to compact, relocatable MAPP code. The editor required is also supplied.

ACCURACY: Eight significant figures, one before the decimal point.

****

3. HISOFT NASPAS

HISOFT Ltd.,
60 Hallam Moor,
Liden,
Swindon SN3 6LS
Cost 35.00

CONSTANTS: Identifiers may be specified as constants and assigned to numeric values, character values or the values TRUE and FALSE
Predefined constants: MAXINT = 32767 {largest integer}
TRUE or FALSE: Boolean constants
ON OFF INVERT: Identifiers of type COLOUR

TYPES: INTEGER:-32768 to +32767
REAL: 7 significant figures, 3.402825 E38 to 5.87747 E-39
BOOLEAN:TRUE AND FALSE
CHAR:256 characters
ARRAY:arrays, with elements of any type
SET:of any simple TYPE
COLOUR:used in conjunction with Nascom block graphics routines

OPERATORS: assignment: :=
arithmetic: +, -, *, DIV, MOD, /
relational: - <> < > <- >- IN
logical: NOT OR AND
set: + - *

FUNCTIONS: Predefined functions:ABS CHR EOLN INCH ODD ORD

PEEK PRED RANDOM SQR SUCC USER ROUND ENTIER FRAC
TRUNC POINT

PROCEDURES: Fully recursive procedures are supported. Value or variable parameters are supported.
predefined procedures: HALT PAGE POKE READLN RESET READ
WRITELN WRITE WRITEHEX LINE(COL,X1,Y1,X2,Y2) {to draw a
line from X1,Y1 to X2,Y2} USER GRAPH

STATEMENTS: BEGIN - END
IF - THEN - ELSE
CASE - OF - ELSE/END
WHILE - DO
REPEAT - UNTIL
FOR - TO/DOWNTO - DO

OTHER: This implementation comes with its own integral editor, which allows a program to be written and saved to tape. It allows the table of standard predefined functions and procedures to be extended readily and simply. It is well documented. It supports the block graphics of the NASCOM 2, or N1 with Econographics.

ACCURACY: Six to seven significant figures.

****

4. Polydata Blue Label Pascal

Poly-data microcenter ApS,
Strandboulevarden 63,
DK 2100 Copenhagen,
Denmark,
Cost approx 50.00

This implementation may also be available from some of the Nascom dealers in this country, but I have not been able to obtain confirmation of this report.

DATA TYPES: REAL 11.5 significant digits (!!!), 1E-38 to 1E38
INTEGER 16 bits, -32768 to 32767
STRING Up to 255 characters
BOOLEAN Logical variables
ARRAY..OF with multiple dimensions

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

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

PROCEDURES: WRITE WRITELN READ READLN LOAD SAVE CALL SCREEN PLOT

STATEMENTS: BEGIN - END
FOR - TO/DOWNTO - DO
REPEAT - UNTIL
GOTO
IF - THEN - ELSE
WHILE - DO
CASE - OF - OTHERS
INIT - TO

OTHERS: The specification of the editor supplied looks impressive, supporting a window on the text, to allow 80 char lines to be handled with ease. It supports 27 editor commands. User written machine code subroutines are supported using procedures/ functions declared as EXTERNAL.

ACCURACY: 11.5 digit accuracy claimed.

****

NEWBUS

NASBUS -- where to ?

D. R. Hunt

For the first time we publish details of `80-BUS', which for legal reasons can't be called Nasbus, but is for all intents and purposes Nasbus spec. issue 4. Most importantly it includes system timing, which has never been published before. This includes some minor changes which improve the bus whilst not changing its compatibilty with existing Nasbus. (The changes are discussed later.) This is the bus standard that the new Gemini computer uses, and it is hoped that it will be made `public domain' so that anything that is designed to use it will remain compatible with both Nascom and Gemini products.

It is my personal hope that Nascom and Gemini will get together to provide a single bus specification under the guidance of a technical committee independant of any manufacturer. That way it will be under firm control from day one and the chaos that developed around the S-100 bus can be avoided. It took six years before the IEEE stepped in and said, "Look lads, you've got a standard, don't keep `bending' it to suit yourselves. We're going to take it over, publish it as a standard, and if you don't conform, then you can't call it S-100." Wishful thinking? I think not, if common sense prevails at this stage, we could even end up with a European standard Z80 bus to rival S-100, to the benefit of all end users. Mind you, manufacturers might not be too happy about it. (Manufacturers don't like being told what they can and can not do by external committees.)

As to the steering committee, I hesitate to volunteer `us', although we have enough technical expertese available, (including access to the originators of Nasbus). It really requires someone with more weight. Some organisation like the BSI or IEE would be more suitable (and acceptable to the manufacturers), although they may consider `some poofling little computer bus' as being beneath them.

80-BUS, a functional description

by Gemini Microcomputers Ltd.

The Nasbus, unlike many other bus systems, has had a very ordered development. However, when we started developing a new range of cards it became apparent that a new revision was urgently required. After a great deal of careful thought and many hours of deliberation the following document was drawn up. It expands on the third issue of the Nasbus functional specification and also attempts to anticipate some of the possible future developments of the bus.

The original Nasbus specification made provision for the extra address and data lines of 16 bit processors. Careful consideration reveals that the bus would not be suitable for this, and so a number of new signals have been defined for the lines made free. The importance of good ground signals can not be overemphasised, and so extra ground lines have also been added.

When defining this bus a great deal of thought went into deciding whether or not to maintain the NAS MEM, NAS IO, and DBDR signals. These signals are particular to Nascom 1 (and NAS IO also to Nascom 2) and are unlikely to be required by any future cards. They therefore constitute a `nuisance'. However, for the sake of compatibility, to avoid the S100 situation, and with pressure from INMC80 it was decided that 80-BUS would maintain support for these signals.

Because of the above considerations 80-BUS remains fully Nascom 1 and 2 compatible. We therefore hope that Lucas will also adapt this specification. 80-BUS will probably not be registered by Gemini, and consequently will become public domain. It is our wish that an independant committee be set up to `guard' the specification of the bus, and to allow any manufacturer who produces a card that fully complies with it to advertise accordingly. The 80-BUS is a Z80 bus and no attempt has been made to make it compatible with any other processor.

One final point is that all cards, including the bus master, should provide a means for being switched out of the memory map under software control. This may be by means of implementing the Page Mode structure, or by some alternative method. This condition also applies to any I/O card that is memory mapped.

Many thanks to David Lewis for the many hours of assistance in drawing up this specification.

80-BUS pin allocation

PIN		SIGNAL	DESCRIPTION
1		GND	Ground
2		GND	Ground
3		GND	Ground
4		GND	Ground
5		CLOCK	System clock
6	*	/NMI SW	A low on this line initiates a short pulse on line 21
7		RSFU	Reserved for future use
8		AUX CLK	4MHz clock signal (optional)
9	*	/RAM DIS	Ram disable
10	*	/RESET SW	Reset switch
11	*	/NAS MEM	Memory decode to Nascom 1
12	*	/NAS IO	I/O decode to Nascom 1 and 2
13	*	/DBDR	Data bus drive, used to change the direction of the data bus buffers on the buffer board or Supermum.
14	*	/RESET	50uS reset pulse, resets entire system.
15		/HALT	Z80 halt signal
16		/BAI	DMA
17		/BAO	daisy chain
18		/BUSRQ	Z80 bus request
19		IEI	Interrupt
20		IEO	daisy chain
21	*	/NMI	Z80 NMI line, (not used by N1)
22	*	/INT	Z80 interrupt line
23	*	/WAIT	Z80 wait line
24		/RFSH	Z80 refresh signal
25		/M1	Z80 opcode fetch signal
26		/IORQ	Z80 input/output signal
27		/MREQ	Z80 memory signal
28		/WR	Z80 write signal
29		/RD	Z80 read signal
30		A0	
31		A1	
32		A2	
33		A3	
34		A4	
35		A5	
36		A6	
37		A7	
38		A8	Z80 16 bit
39		A9	address bus
40		A10	
41		A11	
42		A12	
43		A13	
44		A14	
45		A15	
46		A16	Optional implementation
47		A17	for extended
48		A18	addressing.
49		GND	Ground to seperate the data and address busses.
50		D0	
51		D1	
52		D2	
53		D3	Bidirectional data bus.
54		D4	
55		D5	
56		D6	
57		D7	

58	RSFU	Reserved for future use
59	INT 0	Interrupt
60	INT 1	request
61	INT 2	lines
62	INT 3	
63	/PWRF	Powerfail warning
64	AUX PWR	Backup power
65	NDEF 1	Not to
66	NDEF 2	be defined
67	GND	Ground to seperate power and signal lines.
68	-5V	
69	-5V	
70	-12V	
71	-12V	
72	keyway	
73	+12V	
74	+12V	
75	+5V	
76	+5V	
77	+5V	
78	+5V	

Notes

1) * is an open collector line.
2) IEI to be linked to IEO on cards not using the interrupt daisy chain.
3) /BAI to be linked to /BAO on cards not using the DMA daisy chain.
4) Bus drivers must be able to drive 75/15 U.L.
5) Bus receivers must not load the bus past 1/0.25 U.L.
6) Bus master to pull up all open collector lines with 2k2.
7) Bus master to pull up the following lines with 10k, /HALT, /MREQ, /IORQ, /RD, /WR, /M1, /RFSH.
8) Bus timing reference point is pin 6 of the Z80. As the bus is in essence a buffered Z80, the timing of bus signals is as the Zilog/Mostek Z80 data book. All Z80 signals are buffered onto the bus with 20nS +/- 10nS buffers, the sole exception being the bus clock which should be 20nS (+/- 10nS) ahead of the Z80 clock (pin6). The timing of other signals is detailed in the description of the particular signal. All expansion card timing must, however be referenced to the bus.
9) Cards using /BAI, /BAO, IEI & IEO should pull them up with 2k2.
10) Bus termination. Long buses may require termination. 220R on each line to a 2.6V low impedance source should solve 99% of problems.
11) Grounding. The ground line to the PSU should be as short as possible and as thick as possible.
12) The names of the various bus signals are as detailed above, please do not change them or abbreviate them, ie AUX CLK not AUX CLOCK or A CLOCK etc.

Comments

The following is a line by line description of the bus and should help resolve any ambiguities.

Lines 1-4, GND.

The quality of the system ground cannot be overemphasised. Ground noise problems were at the root of the now infamous Nascom "Memory plague". The faster that systems go the more critical the noise problem will become. Noise problems will manifest themselves as a generally unreliable system with a predilection to do "odd" things.

Line 5, CLOCK.

This line is important as all the bus timing is derived from it. It should spend at least 46% of its time below VOL (0.4V) and at least 46% of is time above VOH (2.4V), it has the other 8% spare to go up and down. The clock on the bus should be 20nS (+/- 10nS) ahead of the clock on pin 6 of the Z80.

Line 6, /NMI SW.

Provision has been made on the bus for an NMI switch and this line is to be held high by the bus master. Grounding it will initiate a short pulse on line 21 and the Z80 /NMI input. Users are cautioned that switch bounce may cause more than one NMI.

Line 7, RSFU.

This line is reserved for allocation at a later date, please do not use.

Line 8, AUX CLK.

This line is a new allocation. Many boards (eg disk controllers) require a 1, 2 or 4 MHz signal. This was easily provided when the CPU clock was 2 or 4 MHz, however the advent of the 6 MHz Z80 changes the situation. Any bus master not running at either 2 or 4 MHz must provide a 4 MHz clock on this line. Designers of expansion cards should take note of this and provide a link to allow the board to use this line instead of line 5.

Line 9, /RAM DIS.

This signal is intended to prioritise memory. Normally this signal would be generated by memory on the bus master, an EPROM card or any other high priority memory when a memory read took place. A RAM card would normally gate /RAM DIS with the output buffer, so that in the event of /RAM DIS being asserted the output buffer would fail to be enabled, this would have the effect of "overlaying" RAM with EPROM/ROM. /RAM DIS should not inhibit a write cycle; it should also remain high for any cycle apart from a memory read.

Line 10, /RESET SW.

A high to low on this line will initiate a reset cycle. It is intended that a switch be connected between this line and ground. The actual RESET line is line 14.

Line 11, /NAS MEM.

This signal is only used by Nascom 1 and is asserted when a Nascom memory address is detected. It would normally be provided by a memory board and would typically be 0000H to 0FFFH or F000H to FFFFH. This is an obsolete signal and no new boards that require it should be designed. This line used to be called MEMEXT.

Line 12, /NAS IO.

This line used to be called IOEXT and many people persist in still calling it that, even though the two are different. In its original form (note that it was active high) it would be taken high to indicate an I/O address external to the Nascom, in its current form it is taken low to indicate a Nascom I/O address, that is to say that it looks for a Nascom I/O address as opposed to looking for an external address. /NAS IO should be taken low within 50nS of a Nascom I/O address and /IORQ, (referenced to the bus). In its original form the onboard ports on the Nascom would remain enabled for a short fraction of an external I/O cycle (the time taken to detect an external address and assert IOEXT) and this was the cause of many obscure problems. If you have problems a good test is to write a short machine code routine to continuously write 80H to port 08H. If the breakpoint register display comes up you have a problem, if not you don't. /NAS IO is a obsolete signal utilised by Nascom 1 and 2 and all new designs should incorporate full I/O decoding.

Line 13, /DBDR.

This signal is used by the Buffer board and Supermum with Nascom 1. It controls the direction of the data bus buffers. When an expansion card outputs data to the bus this line must be taken low, normally this is the same signal as used to turn on the output buffers. Despite being only used by expanded Nascom 1s it is felt that this signal must be provided on all expansion cards. While /DBDR should ideally go low before the data bus driver is enabled, it must be low within 30nS of the data bus driver being enabled and must release /DBDR within 30nS of the data bus drivers being disabled.

Line 14, /RESET.

This line is the "cleaned up" version of line 10. It is important that the falling edge of the reset pulse on this line be synchronised with the falling edge of /M1, and the bus master must provide the appropriate logic to take care of this. The last issue of the Nasbus specification called for a 10uS reset pulse. This has now been extended to 50uS as chips in the 179X family require a 50us pulse. Deep investigation of the matter has yet to yield a 179X chip that can tell the difference between a 10uS pulse and a 50uS pulse. N2 owners who are concerned by this should substitute a 10nF capacitor for C1 (1nF). Supermum owners need not worry as this has been taken care of.

Line 15, /HALT.

Z80 halt signal. Up to this point in time nobody has used it, but it is there.

Lines 16 17 18, /BAI /BAO /BUSRQ.

/BAI and /BAO are the DMA daisy chain. If an expansion card wishes to take control of the bus (an expansion card is any bus card which is not a bus master) it asserts /BUSRQ; the bus master will respond by taking /BAO low. The mother board connects line 17 to 16 between each slot, ie line 17 of the bus master will go to 16 of the adjacent card, line 16 at the bus master is not used although it can be a test point. Between all subsequent cards line 17 goes to line 16 (for full connection details see the section on daisy chains). Cards that do not use the DMA facilities should connect 17 to 16. /BAO will be fed into the /BAI of any potential DMAing device, and the /BAO of the same device will go into the /BAI of the next and so on. If a potential DMAing device has not asserted the /BUSRQ line it will pass on the signal. When the signal reaches the device which originally asserted the /BUSRQ line it will hold /BAO high, at this point it will have taken contol of the bus. The highest priority device is that nearest the bus master.(CPU card).

Lines 19 20, IEI IEO.

Interrupt daisy chain for vectored interrupts. The IEI input of the highest priority device is held high, the IEO output of that device goes to the IEI input of the device with the second highest priority, the daisy chain is continued until the device with lowest priority is reached, its IEO is not connected. It is recommended that line 20 of the bus master is linked to line 19 of the adjacent card and so on down the bus. As the interrupt daisy chain does not involve the Z80 it is possible to move the bus master from slot to slot and vary the level of interrupt priority of the devices on the bus master. The DMA daisy chain however does involve the Z80, and the bus master must always be to one side of the expansion cards which may generate a /BUSRQ. If the motherboard is linked in the recommended manner the device with highest interrupt priority is nearest to the bus master. If the daisy chain is connected the other way arround a problem could arise as the Z80-DMA can also generate interrupts. For further details on connections see the section on daisy chains.

Line 21, /NMI.

A short pulse will be generated on this line by the bus master from a low on line 6 (/NMI SW). On Nascom 1 the NMI is used in the single step feature and is not available.

Line 22, /INT.

Used for the Z80 maskable interrupt. For full details see the book "Z80 family program interrupt structure" available from Zilog.

Line 23, /WAIT.

Used to insert wait states into Z80 machine cycles. Expansion cards that require wait states should provide them.

Line 24, /RFSH.

Used to control the refreshing of dynamic RAM. It should be noted that a refresh cycle is a memory cycle and designers should take appropriate steps. The I register contents will appear as the top eight bits on the address bus during a refresh cycle.

Line 25, /M1.

Z80 /M1 used to indicate an opcode fetch, also used (in conjunction with /IORQ) to indicate an interrupt acknowledge cycle.

Line 26, /IORQ.

Used to indicate a Z80 I/O cycle. The port address will be on the bottom eight address lines (A0 to A7). The top eight will have the contents of the A register on them. If /IORQ is asserted with /M1 it indicates an interrupt acknowledge cycle and the Z80 will expect to receive an interrupt vector.

Line 27, /MREQ.

Used to indicate a Z80 memory cycle.

Line 28, /WR.

Used to indicate a Z80 write cycle, asserted in conjunction with /IORQ or /MREQ.

Line 29, /RD.

Used to indicate a Z80 read cycle, asserted in conjunction with /IORQ or /MREQ. It should be noted that /RD is not asserted during an interrupt acknowledge.

Lines 30 to 45, A0 to A15.

Z80 address lines, should be tristated during a /BUSAK.

Lines 46,47,48, A16, A17, A18.

Optional implementation for extended addressing, should be tristated during a /BUSAK.

Line 49, GND.

An additional ground line to reduce system noise. Must be implemented on both the mother board and on expansion boards.

Lines 50 to 57, D0 to D7.

Z80 data lines, should be tristated during /BUSAK.

Line 58, RSFU.

This line is reserved for allocation at a later date. Please do not use.

Lines 59 60 61 62, INT 0 1 2 3.

Interrupt request lines, used to generate interrupt vectors from devices that are not capable of generating their own interrupts. These lines would be monitored by an interrupt controller which would be capable of generating interrupt vectors, the controller must be capable of being programed with the sense of a particular line (i.e. whether its active high or active low) and the vector. A device unable to generate a vector would assert one of these lines when it required to interrupt, expansion cards availing themselves of this facility should provide it via links so that the particular line can be selected by the user.

Line 63, /PWRF.

Powerfail warning. This line is to be taken low 100mS before the power rails drop by more than 5% and held low until 100mS after the power on reset. For use by backup memory etc. Optional signal which would be provided by the power supply circuitry if implemented.

Line 64, AUX PWR.

An auxilary +5 volt supply for the use of backup devices. Absolute maximum current when the main power supplies are off is 100mA. Implementation is optional.

Lines 65 66, NDEF1 NDEF2.

Not to be defined. These are lines for users to allocate as they require, there are only two restrictions and one provision.

a)TTL levels only, ie no voltage greater than 5 volts and no voltage less than 0 volts.

b) No transition until 100nS after the previous transistion, ie don't put a 16MHz clock on this line.

c) A link must be provided to disable the use of NDEF 1,2.

Line 67, GND.

An additional ground line to separate the power lines from the rest of the bus.

Lines 68,69, -5 volt supply.

Lines 70,71, -12 volt supply.

Line 72, Keyway

Lines 73,74, +12 volt supply.

Lines 75,76,77,78, +5 volt supply.

Daisy chains

```
/BAI  16 ---------X- 16 ---------X- 16
            I              I
/BAO  17 -X--------- 17 -X--------- 17


IEI   19 ---------X- 19 ---------X- 19
            I              I
IEO   20 -X--------- 20 -X--------- 20

    SLOT 2        SLOT 1        SLOT 0
      Expansion cards       Bus Master

     X = Cut of bus track
     I = Link between tracks
```

Basic

BASIC ROUTINES

We've had several goes at persuading various people to tear the Nascom 8K BASIC apart, because it must be just `chock full' of useful routines. Up to date, we haven't had much success, either those we approached were too busy, or something. So were we surprised when we were sent the following by one of those `precocious brats' who has taken the BASIC to pieces in odd moments whilst studying for his `O-levels'. Makes yer sick don't it?

Anyway, young Steve hasn't told the whole story, he hasn't filled in the details of what the registers have to contain when the routines are called. He's promised that as part 2 for the next issue. So for those of you who wish to figure out the remainder of what could be a very powerful machine code floating point maths package, here are the calls.

Bits of BASIC . . .Routines and all that.

by Steve Hansellman

Nascom BASIC contains a number of useful routines (if you know where to find them). So feeling masochistic, I decided to find the more useful ones.

Function	Address	Function	Address
SGN	F822	INT	F8E6
ABS	F838	USR	1003
FRE	F0D0	INP	F441
POS	F0FE	SQR	FAAC
RND	FB8B	LOG	F6C7
EXP	FAFA	COS	FC00
SIN	FC06	TAN	FC67
ATN	FC7C	PEEK	F5A3
DEEK	FDBC	POINT	1051
LEN(	F382	STR$	F19A
VAL	F41C	ASC	F391
CHR$	F3A2	LEFT$	F3B2
RIGHT$	F3E2	MID$	F3EC
NEW	E4B9	LIST	E6DD
FOR	E779	RESTORE	E846
STOP	E870	END	E872
CONT	E89E	NULL	E8B1
CLEAR	E9CA	RUN	EA10
GOSUB	EA1C	GOTO	EA2D
RETURN	EA4B	DATA	EA70
REM	EA72	LET	EA87
ON	EAE1	IF	EAFF
PRINT	EB23	INPUT	EBFD
READ	EC2C	NEXT	ECF6
DIM	EF28	OUT	F450
WAIT	F453	CSAVE	F4C3
CLOAD	F4F9	POKE	F5AA
CLS	FD8B	WIDTH	FDA5
LINES	FDAD	DOKE	FDC7
SCREEN	FDE6		

Other useful routines:

Print character in A	--	E69B
Input a line	--	E607
Output `?' then get line	--	E4FC

Routine to get next non space character.C flag set if valid ASCII digit	--	E836
Print HL in decimal	--	F9AD
Amount of memory	stored @	10DA
Ctrl `O' flag	stored @	1045
End of program	stored @	10D6
Start of text	starts @	10F9
Nulls	stored @	1041
Line length	stored @	1042
Location of text	stored @	1049
Line buffer	starts @	104B

While I was disassembling the BASIC I found a number of things that aren't mentioned in the manual such as ctrl `O'. "What is ctrl `O'?", I hear you ask. It suppresses output (long word `suppresses'). Can I hear shouts of, "Useless!"? Well its not, because a flag is stored in the workspace. Poke the flag and you suppress output, useful for passwords and the like. Mind you this only works while the BASIC has control of the input buffer. This occurs when the input is in the NAS-SYS or NASBUG XO mode, or during an INPUT statement.

Ever noticed how slow BASIC is on starting up? Take a look at locations FCCDH through to FCD4H these locations form a routine which is simply a delay. At this point I will grit my teeth and hope that I'm right and that it is all it does.

Ever wondered why you can't single step BASIC? It's because it changes the NMI jump vector in the workspace. If you want to single step BASIC ignore any calls to FEBBH because that's where it changes. Ok so it changes the NMI jump so what does it change it to? It changes it to the break function, so if you can generate an NMI you can use it as a break key. How do I generate an NMI ? Simple connect a push to make switch between pin 4 of the keyboard and ground. (Nascom 2 only. Ed.)

Hope that this helps you, Steven Hanselman
Nutty Nascom Owner.

HOMILIES (- are they legal !?)

Two little homilies (I've just looked that up in the dictionary, it said, "Tedious moralizing discourse.", I didn't know how close my choice of words was). Both of these concern the design and/or testing of new equipment, and both are printed here for those just on the point of desparation.

The first came from my father, who spent a lot of my early childhood constructing television sets out of war surplus radar sets. As I remember, they had long persistence green screens. When I first cut my teeth on a soldering iron (and that hurt), he told me, "It will always take so long to build a thing, and then at least five times as long to make it work.". That one has always stood me in good stead when doubtful homebrew circuits didn't do as expected. I must admit it's preserved my sanity, if only in a perverse desire to prove my old man wrong.

The origin of the second is more obsure, I may have even invented it myself. I certainly use it in the shop when customers present themselves with a handful of dud chips. It goes as follows. "One dud is too bad, change it, the second is coincdence, three is `Hmmmm', and four or more, is `you've done something wrong mate'." This one proved it's worth only yesterday, when a customer came in with no less than eight `dud' CD4028s. Now eight is stretching even my credulity a bit far, so after applying the above rule, it turned out that he'd forgotten to connect the power rails to any of the chips. QED.

Dr. D's Diary

Episode the Ninth. (Can't think of a better name for it, unfortunately, so that will have to do.)

Big decision time

As I probably neglected to mention, I recently added a Cottis-Blandford cassette interface to Marvin (for the benefit of new readers, this means "my Nascom 1"). With great care, I designed a method whereby both the old and the new tape systems could be used "at the touch of a button" (1 point for that one!) It didn't work. Only the new system would work.

Situations such as these separate the good tapes from the bad. Suddenly, instead of having to sit and convert the tapes to the new system, I was faced with the horror of having to re-type anything that I wanted to use in the future. These are the moments, Nasfans, when deep thought has to come into play. (I mean, am I going to be prosecuted by the Receiver for using the expression "Nasfans"?) Think long and hard about the priority you would give to Star Trek (I chose it) or some boring simulation of the British economy, converted from a program for the TRS80, pinched from Personal Computer World (who once took six months to fail to decide whether or not to print an extremely dull article of mine), and you will probably reach the same conclusions about what constitutes a worthwhile program as I did. The first thing to go in was my assembler. This was closely followed by the amazing graphics utility I use with Nascom BASIC, in order to get it to do fast graphics. (That is enough of the advertising, thanks - Ed) (That was not me - Real Ed.) Then Star Trek, of course. But, after those had gone in, and I had reached new levels of typing skill and boredom, there came a process of decision. And I can honestly say that I don't really miss the programs that are lost. Not at all. Well, actually, I miss Lollipop Lady, the most truly original computer game I have seen yet...

Ongoing hardware situation.

Yet another board has been added to the fabulous Marvin's motherboard. This time, it is a Winchester Technology WT910 sound board. (Actually, this is only one of the two boards I have added, of which more in a moment or two.)

The board has been widely advertised, so you don't need me to tell you too much about what it can do, do you? Oh, all right then, but only because I'm paid by the page. (This is an outright lie - Ed.) The board appears to the processor to be sixteen memory locations that you can only write to. One of the reasons you can't read them is that, presumably to avoid interfering with other RAM locations, they are at the same part of the memory map as the monitor. It is possible to "move" the board to the top of the memory map if you happen to have a disc based system, by changing one integrated circuit. This just happens to be a cheap one, which is not the usual thing, in circumstances such as these!

These sixteen "memory locations" are the control registers of an AY-3-8910 chip, which is pretty well the Rolls Royce (or, to put it another way, the Nascom) of the "sound synthesiser on a chip" world. It has three separate tone generators, each of which can be independantly controlled as regards volume, frequency, presence or absence of white noise. Automatic envelope shaping of the output is available, and this can provide fascinating rhythmic effects. At least, I find them fascinating! Several optional enhancements are available: sound modulator to make the sound come out of your TV, digital to analogue converter (so you can design wave shapes of your own, all you machine code addicts) and an amazing dedicated (well, it seems enthusiastic, at the very least!) microprocessor. The latter has been mask programmed with the first few notes of several well known tunes, and is the kind of gimmick I find quite hard to resist. In fact, I didn't resist, so that every time I switch Marvin on, he produces an impression of Westminster chimes that makes the dog think there is someone at the door.

The quality of the board is superb, the documentation tells you all you need to know, and the price is probably reasonable, if you are not the sort of person who enjoys hand carving his/her own circuit boards from the virgin copper laminate. But, if you are that sort of person, then read on......

Information for masochists.

I also bought a WT100. That is a prototyping board, as if you knew not! On it, I have constructed a stereophonic sound system, which has two AY-3-8910 chips, output to my hi-fi and will soon have two digital to analogue converters, plus anything else in the sound synthesis line that I can get to work with the system. The circuit, while it is not copied from the Winchester one, is rather similar, except for the addressing. The output from the Winchester board is mixed into both channels of my board, thus appearing in the centre of the stereo "stage". Anyone who is contemplating building a circuit at all "similar" to the WT910 (and you certainly mustn't copy it, or you will be guilty of some sort of crime thingy) will want to know that the circuit in the documentation, while it is the truth, is not the whole truth. Far be it from me to tell you that pins 3 and 4 of IC5 must be connected to the 0 volt line. If I told you that, you would be able to build the board from the information its makers are willing to sell you, and that would never do.....

I have written one or two "music" generating programs, of a fairly trivial kind in BASIC; when I have written something I can feel proud of, it will be sent in for the program library. (See also the "Whatever happened to..." section for my libellous remarks about the latter.)

Not a book review, really.

I have recently read all of (and even understood quite a lot of) Douglas Hofstadter's book, "Godel, Escher, Bach: an Eternal Golden Braid". I would tell you more about it, were it not for the fact that Malcodm Peltu has already reviewed it in Personal Computer World, and he writes a pretty nifty review. (It is interesting, however, to note the element of recursion involved in that last sentence - a review of a review - since a major theme in the book is recursion.) If I say buy it, and you don't like it, you will be upset, because it is expensive: I consider the money well spent, though. If you are interested in the great question of Life, the Universe and Everything, which is to say, "is it possible to write a program that is intelligent?" then I can assure you that this book has enabled me to think a great deal more clearly about such things. You will have realised that I dare not reveal the secrets the book holds. Besides, it is not as easy to summarise as Pzoust....

A book review, really.

Another recent purchase (notice the restatement of this major theme in the great computer hobby) is "The CP/M Handbook" by Rodney Zaks. I bought this because I hope one day to have enough money to buy a disk drive or two, and have read reviews that suggest CP/M is not too well explained by its manuals.

Rodney Zaks is even more well known than myself, on account of his far greater output; mind you, he started first, and I don't think he uses Naspen, so it may be that I will eventually be able to catch up, or even overtake him, who can tell? (Good plug - Ed.) This is the first book of his I have read, and certainly makes using CP/M sound considerably less intimidating than it did previously. In fact, the man makes it all sound suspiciously easy. The people who make a lot of money out of computing are those who have convinced their employers that they are doing something very difficult. Could it be that the mystique surrounding disk operating systems is just a paper tiger? (This is not an advertisement for a printer.) Let us hope so.....

A voice crying in the wildezness.

Would someone who knows how to connect a Nascom to a Frieden Flexowriter please contact me. For those who have never seen this product of the Dutch(?) branch of Singer, it is a sort of teletype, which can also be used to hammer steel plate into much thinner steel plate. The noise of the thing when running is beyond the power of my descriptive talents to describe. It gets even louder if the punch is switched on, and I worry whether the foundations of the house will put up with it for long, if I do manage to interface the thing.

The "Let's write an Interpreter" section.

All microcomputer users should, at some stage in their programming experience, try to design a programming language, and then implement it on their system. This is a very good programming excercise, and will also, it is to be hoped, teach them not to moan quite so much about the shortcomings of the work of others.

The first step, in all cases, is to specify what the language is. Pilot is used for writing conversational programs, used in education. A considerable amount of information about using Pilot can be found in a series in Computer Age; just as well, really, as I have no expertise in the field of educational methods. A program in Pilot consists of a series of lines. Each line consists of a label, or an instruction, or both. An instruction consists of the following parts:

1/ A valid Pilot op-code, consisting of one or two capital letters.
2/ An optional condition code, the possible forms these can take will be described later. If the condition is not met, then the line will not be executed.
3/ A separator, consisting of a colon followed by a space.
4/ The "text field", the contents of which vary, depending on the op-code type.

It is, of course, possible to state all the above with far greater precision by using the "Backus-Naur Form" of syntax definition. The actual op-codes can also best be defined in this way. BNF is splendad, but takes pages and pages, so we won't use it.

Some versions of Pilot get round the problems caused by the lack of line numbers in the language by adding pseudo-line-numbers. Of course, this is a shameful cop out, and we will have none of it. This will mean that the editor will not be as wonderful as the one the Nascom BASIC has. For a really good description of why it is easier to write an editor if the language has line numbers, see P.J.Brown's book, "Writing Interactive Compilers and Interpreters" at your library. Or buy it, if you have that much money, and are that interested.

Another thing we are going to do that Brown advises against, is store our language in the same form that it is written. This will save the effort of writing a translator, of the type the Nascom BASIC uses to convert reserved words into single byte tokens. Pilot is particularly suited to this storage format, because it has such short op-codes. There's a catch, of course. The section of the program that actually runs the Pilot program becomes slightly more complex. With other languages, this can be a major problem......

The op-codes allowed in this version of Pilot are as follows:

A = Accept input from keyboard.
C = Calculate.
E = End subroutine.
J = Jump to label indicated.
M = Match the contents of the text field with the contents of the input buffer.
P = Precondition, or defined process, instructions. PR, in Common Pilot.
R = A remark.
S = Stop.
T = Type, similar to PRINT in BASIC.
U = The equivalent of GOSUB.
V = Call machine code subroutine. (This is dreadfully non-standard, but handy.)

Purely for the sake of simplicity, I have ignored the standard form of Pilot, which may well result in my system being ignored. Not to worry, I can always claim that I am really writing about how to write an interpreter! In Brown's book, ignoring language standards is described as the fourth deadly sin.

The Command Level.

First, have a look at flow chart number 1, which symbolises the "highest" level of our interpreter. The box labelled INIT refers to the necessary initialisation that must be done, as most of you will have realised. Rather than describe the whole of the command level, I shall procede to code it, before your very eyes! (By the way, has anyone found a way of passing Zeap files to Naspen?)

The Pilot source code, part 1.

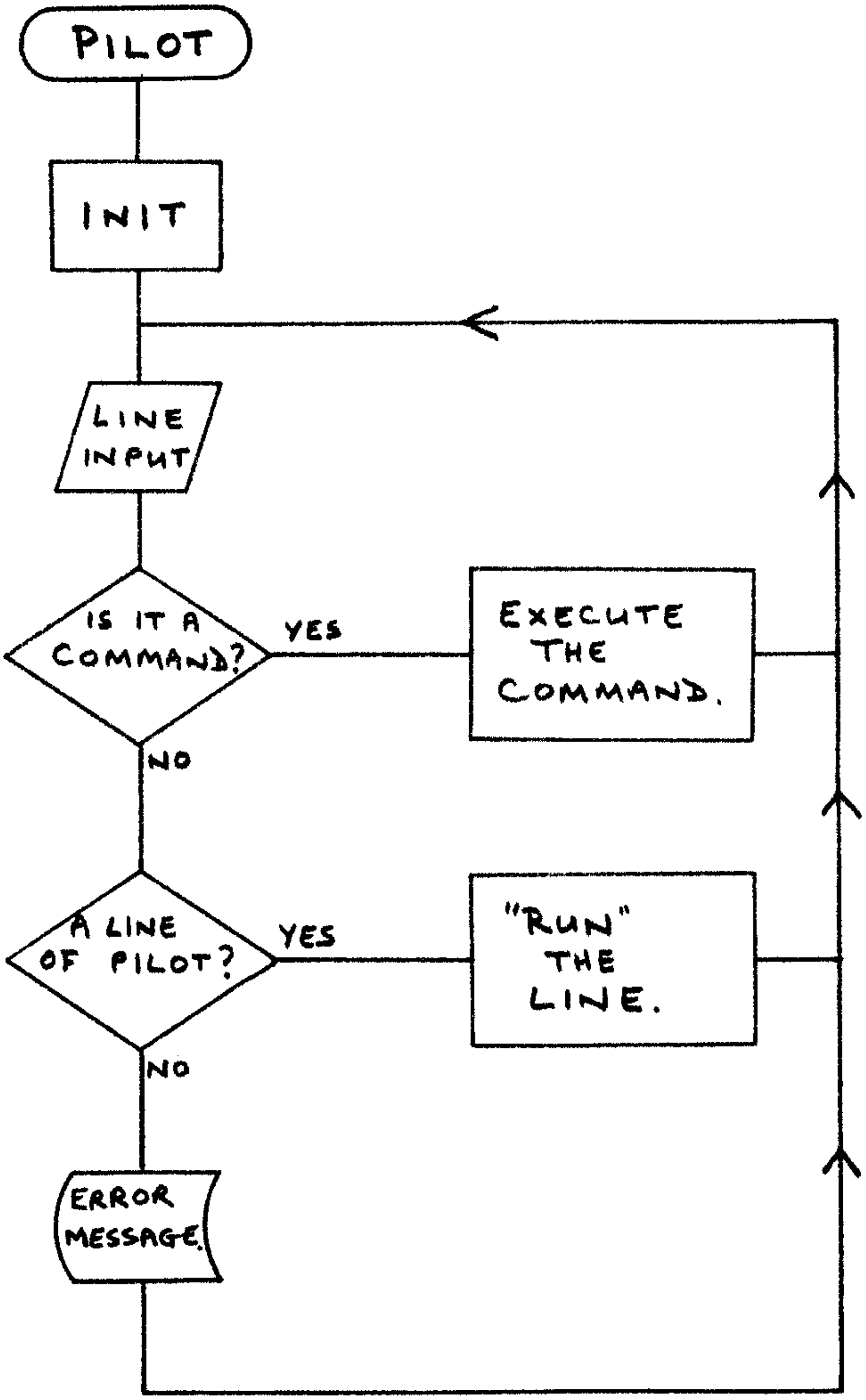
```
Z80 Assembler :      Source Listing

               0010 ;DOCTOR DARK'S PILOT SYSTEM
               0020 ;DEFINE MEMORY LOCATIONS
 8000          0030        ORG  8000H
 8000 1000     0040 RAM    EQU  1000H
 8000 8000     0050 RAMTOP EQU  8000H
 8000 0D00     0060 SCRAT  EQU  0D00H
 8000 1000     0070 STACK  EQU  RAM
               0080 ;DEFINE ASCII CODES USED
 8000 0000     0090 NUL    EQU  0
 8000 000C     0100 CS     EQU  0CH
 8000 000D     0110 CR     EQU  0DH
               0120 ;DEFINE RESTART NUMBERS
               0130 ;*** SYSTEM DEPENDANT ***
 8000 0018     0140 SCAL   EQU  018H
 8000 0030     0150 ROUT   EQU  030H
               0160 ;DEFINE NAS-SYS ROUTINE NAMES
               0170 ;*** SYSTEM DEPENDANT ***
 8000 005B     0180 MRET   EQU  05BH
 8000 0063     0190 INLIN  EQU  063H
               0200 ;START OF PROGRAM
 8000          0210        ENT
 8000 C30380   0220 BEGIN  JP   START
 8003 3A000D   0230 START  LD   A (SCRAT)
 8006 B7       0240        OR   A
 8007 2803     0250        JR   Z WARM
 8009 CD6581   0260        CALL INIT
 800C CD4280   0270 WARM   CALL MESSAG
 800F 50696C6F 0280        DEFM /Pilot ready:/
      74207265
      6164793A
 801B 0D00     0290        DEFB CR,NUL
 801D CDA380   0300        CALL VINLIN
 8020 217A80   0310        LD   HL COMTAB
 8023 CD5180   0320        CALL SEARCH
               0330 ;IF VALID COMMAND WILL NOT RETURN
 8026 CD7381   0340        CALL IMLIN
               0350 ;IF VALID IMMED LINE WILL NOT RETURN
 8029 CD4280   0360        CALL MESSAG
 802C 49206265 0370        DEFM /I beg your pardon?/
      6720796F
      75722070
      6172646F
      6E3F
 803E 0D00     0380        DEFB CR,NUL
 8040 18CA     0390        JR   WARM
```

```
8042 F3         0400 MESSAG DI
8043 E3         0410        EX   (SP) HL
8044 7E         0420 MES1   LD   A (HL)
8045 23         0430        INC  HL
8046 A7         0440        AND  A
8047 2805       0450        JR   Z MES2
8049 CDA680     0460        CALL VROUT
804C 18F6       0470        JR   MES1
804E E3         0480 MES2   EX   (SP) HL
804F FB         0490        EI
8050 C9         0500        RET
8051 D5         0510 SEARCH PUSH DE
8052 1A         0520 SEA1   LD   A (DE)
8053 FE20       0530        CP   20H
8055 2807       0540        JR   Z SEA2
8057 BE         0550        CP   (HL)
8058 2008       0560        JR   NZ SEA3
805A 23         0570        INC  HL
805B 13         0580        INC  DE
805C 18F4       0590        JR   SEA1
805E 7E         0600 SEA2   LD   A (HL)
805F A7         0610        AND  A
8060 2810       0620        JR   Z SEA5
8062 7E         0630 SEA3   LD   A (HL)
8063 A7         0640        AND  A
8064 2803       0650        JR   Z SEA4
8066 23         0660        INC  HL
8067 18F9       0670        JR   SEA3
8069 23         0680 SEA4   INC  HL
806A 23         0690        INC  HL
806B 23         0700        INC  HL
806C 7E         0710        LD   A (HL)
806D D1         0720        POP  DE
806E A7         0730        AND  A
806F C8         0740        RET  Z
8070 18DF       0750        JR   SEARCH
8072 23         0760 SEA5   INC  HL
8073 5E         0770        LD   E (HL)
8074 23         0780        INC  HL
8075 56         0790        LD   D (HL)
8076 EB         0800        EX   DE HL
8077 D1         0810        POP  DE
8078 D1         0820        POP  DE;WASTE RETURN ADDRESS
8079 E9         0830        JP   (HL)
807A 52554E     0840 COMTAB DEFM /RUN/
807D 00         0850        DEFB NUL
807E 8381       0860        DEFW RUN$
8080 4C4F4144   0870        DEFM /LOAD/
8084 00         0880        DEFB NUL
8085 9981       0890        DEFW LOAD$
8087 53415645   0900        DEFM /SAVE/
808B 00         0910        DEFB NUL
808C AE81       0920        DEFW SAVE$
808E 48454C50   0930        DEFM /HELP/
8092 00         0940        DEFB NUL
8093 AA80       0950        DEFW HELP$
8095 4D4F4E     0960        DEFM /MON/
8098 00         0970        DEFB NUL
```

```
8099 A880     0980        DEFW MON$
809B 45444954 0990        DEFM /EDIT/
809F 00       1000        DEFB NUL
80A0 C381     1010        DEFW EDIT$
80A2 00       1020        DEFB NUL;END OF TABLE
              1030 ;VECTORED MONITOR CALLS
              1040 ;*** SYSTEM DEPENDANT ***
80A3 DF       1050 VINLIN RST  SCAL
80A4 63       1060        DEFB INLIN
80A5 C9       1070        RET
80A6 F7       1080 VROUT  RST  ROUT
80A7 C9       1090        RET
80A8 DF       1100 MON$   RST  SCAL
80A9 5B       1110        DEFB MRET
80AA CD4280   1120 HELP$  CALL MESSAG
80AD 54686973 1130        DEFM /This is the command leve/
     20697320
     74686520
     636F6D6D
     616E6420
     6C657665
80C5 6C206F66 1140        DEFM /l of Doctor Dark's Pilot/
     20446F63
     746F7220
     4461726B
     27732050
     696C6F74
80DD 696E7465 1150        DEFM /interpreter. The command/
     72707265
     7465722E
     20546865
     20636F6D
     6D616E64
80F5 73206176 1160        DEFM /s available are:/
     61696C61
     626C6520
     6172653A
8105 0D       1170        DEFB CR
8106 20204C4F 1180        DEFM /  LOAD    SAVE/
     41442020
     20205341
     5645
8114 0D       1190        DEFB CR
8115 20205255 1200        DEFM /  RUN     EDIT/
     4E202020
     20204544
     4954
8123 0D       1210        DEFB CR
8124 20204D4F 1220        DEFM /  MON     HELP/
     4E202020
     20204845
     4C50
8132 0D       1230        DEFB CR
8133 466F7220 1240        DEFM /For further help, please/
     66757274
     68657220
     68656C70
     2C20706C
     65617365
814B 20726566 1250        DEFM / refer to the manual!/
     65722074
     6F207468
     65206D61
     6E75616C
     21
8160 0D00     1260        DEFB CR,NUL
8162 C30C80   1270        JP   WARM
              1280 ;ALL AFTER THIS IS TEMPORARY
8165 CD4280   1290 INIT   CALL MESSAG
8168 494E4954 1300        DEFM /INIT: /
     3A20
816E 00       1310        DEFB NUL
816F CDD581   1320        CALL NOTYET
8172 C9       1330        RET
8173 CD4280   1340 IMLIN  CALL MESSAG
8176 494D4C49 1350        DEFM /IMLIN: /
     4E3A20
817D 00       1360        DEFB NUL
817E CDD581   1370        CALL NOTYET
8181 C9       1380        RET
8182 C9       1390        RET
8183 CD4280   1400 RUN$   CALL MESSAG
8186 52554E20 1410        DEFM /RUN MODULE: /
     4D4F4455
     4C453A20
8192 00       1420        DEFB NUL
8193 CDD581   1430        CALL NOTYET
8196 C30C80   1440        JP   WARM
8199 CD4280   1450 LOAD$  CALL MESSAG
819C 4C4F4144 1451        DEFM /LOAD UNIT: /
     20554E49
     543A20
81A7 00       1452        DEFB NUL
81A8 CDD581   1453        CALL NOTYET
81AB C30C80   1460        JP   WARM
81AE CD4280   1470 SAVE$  CALL MESSAG
81B1 53415645 1471        DEFM /SAVE UNIT: /
     20554E49
     543A20
81BC 00       1472        DEFB NUL
81BD CDD581   1473        CALL NOTYET
81C0 C30C80   1480        JP   WARM
81C3 CD4280   1490 EDIT$  CALL MESSAG
81C6 45444954 1491        DEFM /EDITOR: /
     4F523A20
81CE 00       1492        DEFB NUL
81CF CDD581   1493        CALL NOTYET
81D2 C30C80   1500        JP   WARM
81D5 CD4280   1510 NOTYET CALL MESSAG
81D8 4E6F7420 1520        DEFM /Not yet written!/
     79657420
     77726974
     74656E21
81E8 0D00     1530        DEFB CR,NUL
81EA C9       1540        RET
```

There endeth part one, and those of you with no charity in your souls are right, the rest is yet to be written. The design of the editor, and then the tape routines, are my next task.



PILOT FLOW CHART #1

1.5 Mbaud

How to load programs at 1.5Mbaud

David Parkinson

(or the Gemini EPROM board and paging)

Some time ago I took the route to disks, and after a while I had CP/M 2.2 up and running on two 8" drives. With the reducing costs of dynamic RAM my system memory had grown to 64K - spread over two RAM-B boards. (The main processor board is a Nascom 2 by the way). My disk software resided from F000-F7FF with the screen at F800-FBFF. I did not want to commit the area F000-F7FF to EPROM as it sometimes was required for other software that I was developing. The net result was that I had a variety of tapes that I used to load, depending on what I wanted to do at the time. The procedure for starting CP/M on first switching on was:-

Ensure Power-on-jump switches set for 0.
Ensure Nas-Sys switched in and screen at 800.
Switch on.
Load Cassette tape @2400 baud to F000-F7FF (Disk & I/O routines).
Enter short machine code program (00 76).
Execute it.
Throw switch which dropped Nas-Sys out and moved the screen from 800-BFF to F800-FBFF. (RAM appeared at 0 as it was no longer overlaid, and the screen now overlaid RAM at the other end of memory).
Alter the power-on-jump to F000.
Press Reset.
Bingo CP/M going!

Once in CP/M life wasn't so difficult. New versions of the Disk & I/O software could be loaded over the existing ones and tried immediately, and only in catastrophic cases would the whole start up procedure have to be repeated. Having the routines sitting there in RAM meant that they could be patched easily, or totally changed, and I had no worries about slow EPROMs and whether I should have the Wait state generator switched on. However it was too much of a rigmarole and there must be a better way! Putting everything in EPROM in the sockets on the Nascom 2 would not help as EPROM gobbles up areas of the memory map and you have programs sitting there that you don't want to use. (If you're running Basic you have no interest in Zeap or Nas-Dis have you?). I also have the odd CP/M program that will not run in systems under 56K! Also occasionally there are various system programs that have conflicting execution addresses. This would all have lead to a horrific switching arrangement if I had attempted to use the sockets on the N2.

I could see only one reasonable way out of the problem, and so gave in to the high-pressure salesmanship that I had recently been subjected to from Interface over the Gemini EPROM board. (Mind you I think they'd given me up by then as the first indications I gave that I was about to conceed weren't followed up!).

The reason for choosing the Gemini board is because it supports the page-mode scheme. This effectively allows you to switch the board on or off by setting or resetting certain bits of port FF. When the board is switched off it vanishes from the memory map, leaving the full 64K of RAM free for the wanted system. (Not strictly true as the screen is still cluttering things up at F800-FC00). RAM-A owners can carry on reading as for the application I had in mind the RAM was specifically required NOT to respond to the page-mode switching, as it was necessary that the RAM be present along with the EPROM board. The two can coexist, as, when the EPROM board appears, it asserts its "RAM DISABLE" line and so overlays any RAM that might lie in its address space.

[Note that any RAM on the Nascom 2 cannot be overlayed. The RAMDIS signal only emerges from the N2, and is not connected back in].

The Gemini EPROM Board

The Gemini EPROM board is a standard 8"x8" Nasbus (compatible! -Ed) card, with silk-screen, solder resist, and with plated through holes. It holds the usual addressing logic/buffering for the Nasbus and has 16 empty 24-pin sockets to accommodate the EPROMs and one additional socket which will accept the Nascom Basic Rom. There is also an on-board wait state generator that can be switched in or out as required. This has an advantage over the one on the Nascom 2 in that it only inserts wait states when the EPROM board is being addressed, and not in every cycle irrespective of whether RAM or EPROM is being accessed. As with the Nascom 2 the wait state occurs in both M1 and MREQ cycles. (If you check your Z80 data sheets you will notice that the critical cycle from the point of view of memory access is the M1 cycle, other Read/Write cycles are 1/2 a clock period longer). As mentioned earlier the Board also supports the Nascom Page Mode of working.

The EPROM sockets are subdived into four banks of four sockets each. Each bank can be decoded to a 4K boundary, and each bank has a strap field associated with it that is used to configure the bank's sockets for either 2708s or 2716s. (The 2708/2716 selection being on a bank basis means that the two types may be accommodated simultaneously on the board, but obviously only in separate banks). The 4K decoding is similar to that of the RAM cards, the wanted address being selected by a wired strap on a header plug. (None of the 'Which bank is which?' of the Nascom 2).

Before starting assembly of the card I plugged it into my system which continued to work happily. This is a procedure I would recommend that everyone follows - it can save time later when there are a lot more faults that could be about. This test will reveal any major faults on the board in those lines that are connected to the Nasbus. (Only once have I had a board that stopped everthing. In that case I found the fault was a tiny whisker of copper across pins 1-6 of the edge connector - the board just hadn't been trimmed properly). The board took a little while to assemble - there seemed no end to those 24-pin sockets! Once again the board was plugged in, this time to catch any possible solder splashes. With everything Ok at this point it was time to add the ics.

All the components were there in the kit, the only error being that two 'S' ics were supplied instead of the 'LS' ones specified, but luckily they were common devices and I had some of the LS ones to hand. (Perhaps someone somewhere needs new glasses?) The only other problem I had with the assembly was that the supplied 24-pin header plug refused to go into its 24-pin socket. After a great deal of careful effort (and loud swearing) it partially conceded and now sits about 2/3rd of the way into its socket.

With the TTL ics in and the board plugged into the bus everything still worked, so it was time to add some EPROMs. I set the strap fields for 2716s,the decode address for 8000 on Bank 0, inserted some 2716s and plugged the board back in. Lo and behold everything worked! However I subsequently discovered one small point that does not appear in the manual. If a 2716 bank is decoded as a 4K block and the 4K block is an even one (0,2,4,6,8,A,C,E) then the 2716s must be placed in positions X0 and X1, if it is odd then they must go in X2 and X3. From this it follows that if a 2716 bank is decoded as two non-contiguous 4K blocks, then one must be odd and one must be even.

To sum up. The Gemini EPROM board is a well produced board that does the job it was designed to do without any apparent vices, and has been a welcome addition to my system.

Back to the main topic

To return to the main topic of the article - we now have a working EPROM board, what do we do with it? First of all all programs that we wish to have "on-call" have to be programmed into EPROM and located on the board. The addresses at which they are placed are purely arbitary, and bear no relation to their execution addresses. (I picked 8000 upwards). The only constraints are that they must not overlay any area of memory that is used by the Nascom on-board memory, (as it does not respond to RAMDIS), or overlay RAM that will be used for the Z80's stack while the board is active. Next, two short control programs have to be written for the board, one to deal with power-up, the other to make it easy to use. The remaining things to do before inserting it is to select it to respond to page 1 - the power-on page - and to set the system's power-on-jump address to that of our control program on the EPROM board.

As I intended to work mainly in a disk environment but did not want to burn my bridges behind me, I left the Nas-Sys 3 EPROM on the N2 board and made a modified copy of it. This version supported screen and workspace at F800-FFFF, and included a small addition (I reduced the sign-on message to accommodate it) which we shall see in a moment. By throwing one switch I could still return to a standard Nascom set-up should the occasion arise.

The power-on (or Reset) program is shown below together with parts of the other control program. The purpose of the power-on section is to copy the modified Nas-Sys to RAM at 0, and then to jump to it paging out the EPROM board on the way.(*** Note that we must be off the board before it's paged out otherwise our program will suddenly vanish as the OUT instruction is executed! ***).

```
8000    C3 8003          RESET:  JP      8003H          ;Get PC loaded
8003    01 0800                  LD      BC,800H        ;2K length
8006    11 0000                  LD      DE,0           ;To here
8009    21 8800                  LD      HL,8800H       ;Nas-Sys ROM site
800C    ED B0                    LDIR                   ;Move it down
800E    21 0000                  LD      HL,0           ;Set start addr
8011    D9               GO:     EXX                    ;Save address
8012    01 0006                  LD      BC,6           ;A bit more to move
8015    11 0800                  LD      DE,800H        ;To here
8018    21 8021                  LD      HL,PON         ;This code
801B    ED B0                    LDIR                   ;Move it
801D    D9                       EXX                    ;Reset addr.
801E    C3 0800                  JP      800H           ;Go to it via RAM
                         ; This section pages the EPROM board out and
                         ; starts Nas-Sys.
                         ; (Copied & executed in RAM)
8021    3E 22            PON:    LD      A,22H          ;Any page except 1
8023    D3 FF                    OUT     (OFFH),A       ;Page board out
8025    E9                       JP      (HL)           ;Off to it

                         .   .   .   .   .   .   .
                         .   .   .   .   .   .   .
                         ;********************************
                         ;  This section offers the      *
                         ; multiple choice               *
                         ;********************************
8100    31 1000          OFFER:  LD      SP,1000H       ;Use Nas-Sys user stack
8103    EF                       RST     PRS
8104    0C 57 68 61              DEFB    CS,`What system?',cr
8108    74 20 73 79
810C    73 74 65 6D
8110    3F 0D
```

```
8112    38 20 2D 20                     DEFB    `8 - Boot 8" drives',cr
                         .    .    .    .    .    .    .
                         .    .    .    .    .    .    .
8176    DF                              RST     SCAL            ;get reply
8177    7B                              DEFB    BLINK
8178    FE 38                           CP      `8'
817A    CA 8194                         JP      Z,EIGHT
                         .    .    .    .    .    .    .
                         .    .    .    .    .    .    .
                         ; Eight inch drives

8194    01 0800                  EIGHT: LD      BC,800H         ;2K length
8197    11 F000                         LD      DE,0F000H       ;To here
819A    21 9000                         LD      HL,9000H        ;Where EPROM is
819D    ED B0                           LDIR                    ;Move it
819F    21 F000                         LD      HL,0F000H       ;Load Jump address
81A2    C3 8011                         JP      GO              ;Go to it!
                         .    .    .    .    .    .    .
                         .    .    .    .    .    .    .
```

The code I added to Nas-Sys is shown below. Its purpose is to page the EPROM board back in and jump to the second control routine. This second routine displays a menu on the screen of what is available on the EPROM board. Typing the appropriate number or letter on the keyboard results in the contents of an on-board ROM(s) being copied to its execution address in RAM, the transfer being done in an identical manner to the down loading of Nas-Sys shown above. Note that the EPROM board is Read only, and that any overlaid RAM is not disabled if a write cycle is executed. This means that the EPROM can be copied to anywhere in RAM, even to the same address as the EPROM!

```
                                 ;  Nas-Sys Additions
                                 ;  "D" command altered to arrive here
07F9    3E 01                    BOARD: LD      A,1             ;Select page 1
07FA    D3 FF                           OUT     (0FFH),A        ;(Turnss EPROM on)
07FC    C3 8100                         JP      OFFER           ;Go to Menu program
```

With this set up it only takes me three key depressions (D <enter> 8) and my 8" `EPROM' is loaded (at a rate equivalent to about 1.5Mbaud!) and CP/M boots in in under a second with no hassle at all! Alternatively D <enter> H for the Henelec disk system, or D <Enter> D for Debug & Nas-Dis, or........

Using this technique you can also hold RAM based routines that you frequently use that will not run in EPROM, or small utilities that you use frequently (but not together) can all execute at the same address... It's up to you what you do. If 32K is not enough you can always buy another EPROM board for page 2...... (Anyone seen what's on page 3? - ED).

DODO OF THE WEEK

"... I also found the 9 column listing offputting when trying to check through for errors in entry as, of course, my Nascom tabs in eight columns."

Name and address withheld to protect the guilty. (Hint - read the back issues!!)

Z80 Guide

THE KIDDIES GUIDE TO Z80 ASSEMBLER PROGRAMMING

D. R. Hunt

Part: The Fourth.

Learning to live with a computer
(whilst avoiding grounds for divorce)

INMC80-3 has just plumped through the letter box. Yes, I do have one sent through the post just like any other member; that way I can check on how long it takes from going to the printers to arriving on the members' doorsteps. Heavens, I haven't continued the saga, and Paul will be nagging me for copy for the next issue any minute. So here goes.

After reading the previous episode to remind myself of the story so far, I must at this stage give a very severe warning to those married members who haven't yet been made aware of the light.

Women (and here I'm presuming that I am writing for a predominately male readership), are strange beings. In general, wives (and girl-friends) will, under normal circumstances, take a considerable amount of abuse without turning a hair. They provide us with food after a weary days work, they clean the house and do the shopping. They will tolerate, nay dote upon noisy kids, they will be polite to their mother-in-law (whilst you hate yours). They will accept that us fellas have mates, and going off for a drink on Thursday nights is but the natural course of life. They create miracles when important people unexpectedly announce they are coming to dinner (even the noisey kids are somehow tamed and locked out of sight). In general we fellas assume this to be the normal domestic role of the women in our lives and is so much second nature that we are lulled into the security of believing that nothing could disturb the smooth running of the household.

BUT BE WARNED. Wives and computers go together like peaches and creosote !!!! The aquiring of a computer produces much the same reaction as bringing home half a dozen of the most doubtful birds imaginable, and then announcing that hence forth these are going to receive your undivided attention until at least three o'clock each morning. To say the least, there is some female instinct which instantly eminates hostility towards your latest possession, and this will seriously affect your wallet. I fully endorse the comments by Len Ford (Issue 3 page 39), and feel that his is a masterly understatement of the facts. Wives, who for years have consented to being taken out to dinner annually on the wedding anniversary (if you happen to remember) now assume it as a right that they will be treated to a regal nosh-up at least once a month. The occasional Mars bar will now have to become frequent 2 pound boxes of the sort of stuff that some loony jumps off moving trains, climbs mountains without ropes with his bare hands, and blows up arab castles to deliver. Nights out with the lads will now take on a sheepish and guilty feeling, which of course that uncanny feminine `sixth sense' will immediately detect and amplify by such monosylabic utterings as, "Hmpff", which of course is to be translated as, "If I don't get taken out to dinner this weekend, we'll see how well you survive on cold dinners and dirty shirts for the next week." So dear naive reader, if you've just bought a Nascom, don't think the capital outlay will finish there. There are the consumables (which I mean literally) to consider, and cultivating a sincere friendship with the manager of the local French Bistro should be your next priority.

There are of course exceptions. I know one couple where the reverse is true. `Mrs' plays with the computer whilst hubby sits and knits (or watches Coronation Street on the box, or something equally constructive). ------ So where were we, we've looked round the innards of the Z80, got the idea that HEX is only a way of expressing binary information, learned some of the stupid mistakes that I made after getting my hands on a Nascom and generally not learned much more than could have been gleaned from any good book about Z80 programming in as many days as it has taken months to read this. This time we'll go on to some of the things that the books seem to assume you should know. It all easy stuff, but if you're not in the know in the first place, how the heck are you supposed to find out ?

Shortly after putting our newly aquired Nascom 1 out in the shop, there was `this 'ere fella' who used to wander in, dressed in motor cycle leathers, and spend a lot of time making the Nascom do things that I had never dreamed possible. And so I got to know Howard (he used to be on the committee, but resigned about a year ago). Soon Howard came a regular visitor to the shop, and could be found on a Saturday explaining the niceties of Z80 machine code programming to customers with far more informed authority than I. What's more, he was so dedicated to Nascoms that we didn't even have to pay him. It was Howard who first put me on the right lines as far as programming goes. Up 'til then, it had been a case of `figure it out for yourself'. Who taught Howard I do not know (perhaps I'll ask him before this article is put to bed), but it was under Howard's guidance that some of the mysteries of machine code programming were revealed.

So far I don't think I have drawn any distinction between machine code programming and assembler programming. To me they are one and the same. Purists will argue that machine code is the result of assembler programming. The machine code itself being the sequence of HEX codes that is generated by the act of assembler programming. That sounds like the same thing to me, so there we are. At the time, I would scribble the machine codes directly on to a piece of paper (Woolies Jumbo A4 work pad for 44p, incredible value) and then type them in to the Nascom. Howard pointed out that this was bad practice on two counts. Firstly, it was difficult to remember what more than about half a dozen codes meant, and secondly, if I practised hard I would learn the syntax of the assembler mnenonics and by so doing increase my vocabulary of instructions because I would be thinking of the descriptive action of the instruction I wanted instead of fumbling to remember the code for the few instructions I knew. After a while it became second nature to write a mnemonic for the instruction I wanted, and then look up in the book. Low and behold, there it was. Sometimes I would take this approach too far. Occasionally I would write assembler mnemonics for instructions that don't even exist. But in the main, it works. Think of a way round the problem, write mnemonics to decribe the action to be taken, then translate the mnemonics into machine codes.

I have already mentioned that there comes a time when the light begins to dawn. I think I was past that stage, for as I remember, I felt confident enough to start writing `HANGMAN'. Now that is an important part of learning to program. Simple games are the very best thing to cut the teeth on, as the rules for the game are already laid out, and you don't have to draw up the specification of what it is you are about to try to achieve. Games like `HANGMAN' have very simple rules, and these in turn are very simply broken down into their component parts. So deciding what each individual part of a program is going to do is relatively simple. Be that as it may, it was Howard who introduced me to such things as labels, flags and text strings.

So let's re-examine the asterisks program from part 3. First of all lets lay down the ground rules of what is to happen. In other words, let's draw up the specification.

1) Put an asterisk on the middle of the screen
2) Hang about a bit
3) Replace the asterisk with a blank space
4) Hang about a bit
5) Go back to (1)

We don't need to draw a flow chart of that, it's too simple. A flow chart in fact would probably confuse the issue. Now let's take it stage by stage and write the mnemonics for it.

```
        LD A,2AH          ; Load A with the code for an asterisk
        LD (09E2H),A      ; Put the contents of A at memory location 09E2H, which just
                          ; happens to be on the screen (see last part to see how that
                          ; was decided.)
```

Now notice that I've use a semicolon to separate the mnemonic instructions from the comments. In other words, the program is already being written in two fields, the instruction (or mnemonic) field and the comment field. Separating them like this makes for easy reading and understandability.

Also, I've used one variable, 2AH (the H denotes HEX, remember), and one fixed location, 09E2H. Now the 2AH is easy to remember, but the 09E2H is not so manageable. Let's change the 09E2H to a name of our own devising which will be meaningful. This is known as a label. Now if you are going to use a label to denote a variable, or a location, you should declare it so that you know what it will be. For this we must introduce a new field into the assembler listing known as the label field, and also for convention sake use what is known as an assembler directive. In this instance an equate, abrieviated to `EQU'. A brief word about assembler directives. These are instructions in the instruction field which aren't part of the Z80 instruction set, but are instructions to you or a nice `hairy' piece of machine code known as an assembler (incidently, the semicolon between the instruction and comment fields is also an assembler directive). We won't persue what an assembler is now, but sticking to conventions is a good idea for later, you'll see. Another convention used for assemblers is that label names should not exceed six letters, as assemblers won't look at labels more than six letters long. Another convention is that labels may be alpha-numeric, but should always start with an alphabetic character. A strange effect of using labels starting with an alphabetic character is that some assemblers get confused between labels and addresses starting with an alphabetic character (A to F), so that the address F080H will be mistaken for a label. Some assemblers won't accept any addresses (or data) starting with a letter, so the above address would have to be written 0F080H for the assembler to know what you were on about.

So now our first part of the mnemonic assembly looks like this.

```
SCREEN  EQU 09E2H

        LD A,2AH         ; Load A with an asterisk
        LD (SCREEN),A    ; Load SCREEN with the contents of A
```

Lets carry on, we're going to call the delay subroutine which is further on in the program at address 0D13H, now an address like that is a location, just as the screen location, and just cries out for a label, and an obvious label name presents itself. This time we don't need an `EQU' as the location will be defined by the position of the routine itself, viz: just after the main routine. Another location defined by its position in the program is the start, so when we come to the `jump back to the beginning' bit, we can again use a label as well.

```
SCREEN  EQU 09E2H

START   LD A,2AH         ; Load A with an asterisk
        LD (SCREEN),A    ; Load SCREEN with the contents of A
        CALL DELAY       ; Hang about a bit
        LD A,20H         ; Load A with a space
        LD (SCREEN),A    ; Load SCREEN with the contents of A
        CALL DELAY       ; Hang about a bit
        JP START         ; Go back to beginning
```

Now at this point the main routine should have become quite understandable in its own right, even without comments. The labels make the purpose of the routine quite clear. Notice it also follows the order of the specification quite closely. Mind you, labels are a two edged tool. You can make the meaning of a program totally obscure by using obscure label names. I still haven't figured out how the label name `TBCD3' in NASBUG and NAS-SYS was arrived at (perhaps I'm just dense). I know what the routine does because I made it my business to understand what the mnemonics do, but I challenge anyone to decide the purpose of the label name `TBCD3' without the accompanying mnemonics. Now let's finish the whole thing off, and add the delay subroutine.

```
        DELAY   LD HL,0000H      ; Load HL with 0
        LOOP    INC HL           ; Increase by 1
                LD A,L           ; Copy L to A
                OR H             ; OR A with L
                JP NZ,LOOP       ; If both not 0 then loop again
                RET              ; Return from subroutine
```

One thing is quite clear when writing it out in mnemonic form, and that is I got it wrong in part 3. The JP NZ is wrong, you look. You see, it's so easy to make mistakes when writing code and not using labels. Go on, have a look. Point proven? Well we all make mistooks. I would have spotted that had I used labels.

Got all that, good, now as one final convenience, let's number the lines, because there's a fair amount of discussion to follow now, and I'm getting fed up typing it out each time. From now on I'll refer to labels and line numbers.

```
10              ORG 0D00H
20      SCREEN  EQU 09E2H

30      START   LD A,2AH         ; Load A with an asterisk
40              LD (SCREEN),A    ; Load SCREEN with the contents of A
50              CALL DELAY       ; Hang about a bit
60              LD A,20H         ; Load A with a space
70              LD (SCREEN),A    ; Load SCREEN with the contents of A
80              CALL DELAY       ; Hang about a bit
90              JP START         ; Go back to beginning

100      DELAY   LD HL,0000H      ; Load HL with 0
110     LOOP    INC HL           ; Increase by 1
120             LD A,L           ; Copy L to A
130             OR H             ; OR A with L
140             JP NZ,LOOP       ; If both not 0 then loop again
150             RET              ; Return from subroutine
```

First, notice line 10, I've introduced a new assembler directive, 'ORG'. Now up till now, we haven't supplied any information as to where the program is going to reside in memory. The origin of the program has not been spelled out. In part 3, the program started at 0D00H, and that's as good a place as any, so lets declare the origin of the program, ORG 0D00H. (Notice the '0' in front, that's so the address doesn't get confused with the label). Notice also that I've put an extra carriage return between the declarations at the front and the main routine, and between the main routine and the subroutine. (I was going to say I'd split up into 'byte size' chunks, but such appalling puns are beneath even my meagre wit.) Anyway, by separating the program thus, it's easy to see what each bit is about.

Now to tidy the program up a bit. One one the major omissions is the use of absolute jumps instead of the quicker and simpler relative jumps. Ok, so what is a relative jump you ask (except those adept at the art of Granny Stomping, and brother, have you got it wrong)? A relative jump is a jump that does not go to an absolute location, but is relative to the current contents of the program counter. This is one of the improvements the Z80 has over the 8080, and is very useful. It means that if a routine is written entirely in relative code (meaning relative jumps are used throughout), then it's not tied to any specific location in memory, and therefore can run anywhere (I know it's a glaring simplification, before the know-alls start jumping up and down, I'm not writing this for you). Secondly, relative jumps require fewer bytes than absolutes. Anyway a relative jump is easy to understand, the important thing to remember is that it's relative to the CURRENT contents of the program counter. Remember, the PC increments by one each time it fetches an op-code or operand byte, so let us suppose we put a relative jump at memory location 1000H, the code for this would be, say:

```
1000    18 05       JR xxxx     ; Jump to label xxxx
```

Then when the Z80 encountered this, the PC would be pointing to 1000H. The Z80 would get the first byte and increment the PC. The Z80 then interprets the first byte fetched (the op-code), note that the PC is already pointing at 1001H. Having decided that the instruction is a two byte instruction, it would fetch the second byte (the operand), and increment the PC to 1002 by so doing. At this stage the Z80 has all the information it requires, and procedes to add the second byte (the operand of the jump relative instruction) to the low byte of the CURRENT contents of the PC. Now the PC had already been incremented by the action of fetching the op-code and the operand, and is now pointing to 1002H, so 1002H + 05H makes the address in the PC 1007H, ie: +5 relative to the CURRENT PC position, and not +5 relative to the start of the instruction as is commonly assumed. It works exactly the same backwards, except that the operand supplied is a negative number. The Z80 still adds this to the CURRENT contents of the PC, and so the PC ends up pointing back, but two bytes short of where you might expect.

Now, out of pure cowardice, part 1 didn't contain any information about 'signed binary arithmetic' (that's positive and negative numbers to you). I don't intend to rectify that omission now, but refer you to any '0 level' maths text book. (Go on, own up to the kids that you don't know it all, and borrow one of theirs; either that, or say you want to find out just how ignorant kids are these days, and to check that the text books contain information on binary arithmetic in this computer age.) Which ever way you go about it, you will find that signed binary arithmetic is a bit of a pain, and the designer of NAS-SYS and Nasbug T4, Richard Beal (God bless his little cotton socks) has included the 'A' command to ease this problem. Now I'm not going to waste paper explaining it here, go and read the NAS-SYS or Nasbug manual and in the light of what I said above, all should become clear. The only thing I will add is that because the operand of a relative jump instruction is restricted to a single byte, the maximum range can only be FFH (256 decimal). Now because of the offset of two bytes, this means that the effective range of a relative jump is 80H (equivalent to 127 decimal steps backwards) to 7FH (equivalent to 129 decimal steps forward). So don't get too clever and try jumping all round the memory in relative jumps, 'cos it just won't work.

Well, all that means is that lines 90 and 140 can become relative jumps, thus saving two bytes of code.

```
80              JR START          ; Go back to beginning
140             JR NZ,LOOP        ; If not both 0 then loop again.
```

Another thing we can have a go at is the delay routine. Now I wrote that because at the time I couldn't think of anything better. (As I admitted in part 3, what I actually wrote was worse, and I'm ashamed to show you exactly what I did.) How many of you realised that NAS-SYS and Nasbug both contain a very nice delay routine as part of the keyboard scanning routine, whats more, its location was deliberately chosen so that users could have ready access to it. Its label is RDEL, and it starts at location 0038H. Now the nice thing about that location is that it is one of the Z80 'restart' points. To be compatible with the 8080, the Z80 included the eight restart points from the 8080. Now these are 'implied CALL' locations and are a fixed part of the Z80 structure. One specific single byte instruction will cause the Z80 to call that location. So here we have a delay routine (its actual delay time varies from 7.5mS to 2.5mS depending upon it being NAS-SYS or Nasbug, or wether the Nascom is running at 2 of 4MHz), accessible by a single byte call. All we have to do is to call that a number of times, and there is our delay.

So now to learn about another very useful Z80 instruction, the 'DJNZ loop'. The 'DJNZ' instruction is translated as 'Decrement, Jump (relative) if Not Zero', and what it does is decrement the B register, test it, and if it's not zero, do a relative jump to the location specified by the operand. If B was zero, then it simply 'drops through' to the next instruction. To use the DJNZ loop we must decide how long the delay is going to be, and because the delay varies between versions of the monitor and the Nascom speed, we'll choose 5mS as a good compromise. Two hundred times 5mS is one

second, so two hundred times round the delay loop will be appropriate. So let's draw up the spec. of the delay routine:
1) Set B counter to 200 (decimal)
2) Delay for 5mS
3) Do the loop until B is 0
Now before we go ahead and use the routine in NAS-SYS, we ought to check that it's not going to muck up anything else. So have a look at it. Go on, see if you can figure out how it works before reading on. Got it, good. It works almost like the counting routine I wrote originally, except that it counts in the A register, and throws in a couple of PUSHes and POPs for no better reason than to waste time. The important point is that the only thing that gets changed is the A register, as, for the routine to return, the A register has to contain 0, so, on the return the A register will contain 0. Now that doesn't matter at all, as the next thing we do with the A register is to fill it with the next character to be put on the screen. But watch out, NAS-SYS and Nasbug usually (but not always) ensure that things are unchanged except where neccessary, but don't assume this to always true. CP/M for instance can be relied upon to change almost everthing in sight when an internal routine is called.

So the delay routine suddenly looks like this:

```
100     DELAY   LD B,0C8H       ; Load the loop counter with 200 (decimal)
110     LOOP    RST RDEL        ; Call RDEL (delay for 5mS)
120             DJNZ LOOP       ; Done 200 loops, no, go round again
130             RET             ; Yes, return from delay subroutine
```

Now there is something I've (deliberately) forgotten. Yup, you got it, there's a location there which I fixed as a label, and I didn't declare it. I know what it is, and you should know (because I told you earlier. But, as my junior school English teacher used to say, "What about the man from the moon, he doesn't know what you're on about because you haven't told the complete story." Mind you, if there are men in the moon, they must be very wise, as they had the good sense to stay out of sight when the Yanks arrived. Anyway, a new line, 15:

```
15      RDEL    EQU 0038H
```

Now we have shortened the delay routine considerably, introduced the concept of nested subroutines, and ended up making the program monitor dependent. The first one was a good idea, short routines are always easier to understand. Nested subroutines are also a good idea as they lead to simple, easy to understand modules, and although they take longer to execute, the trade off between speed and simplicity must be in favour of simplicity for the purposes of this demonstration, after all, the nested subroutine is an arbitory delay, so speed in accessing it can hardly be important. The last point about making it monitor dependent is very arguable. The program now depends on NAS-SYS or Nasbug being present, so it's dedicated to a Nascom using one or other of the monitors (I don't know about NAS-MON, I've never tried it). The program is now monitor dependent, unlike my first attempt, where the only requirement was RAM at the point I chose for the program to run, and a screen RAM at the point where I put the characters. On a little example like this, it's not very relevant, but when you start using large chunks of the monitor, routines like 'KBD' and 'CRT' then programs written will be several hundred bytes shorter, but will be very definitely monitor dependent.

Now for the last part of this lesson. (Thank goodness for the little space counter on Diskpen, it tells how close to the end I am before I get there.) The HL register has now been freed by the use of RDEL in the monitor. Now HL is a very useful register, it is the main 'pointer' register in the Z80, and may be used for 'pointing at' a memory location whilst the byte 'pointed at' is manipulated. Lets rewrite the whole shooting match, using HL as a pointer, and not get too thorough about the explanations, to see how you get on.

```
BLINKING ASTERISKS              MACRO-80 3.35                   Page    1

                                Title BLINKING ASTERISKS

                                .Comment \

                                Author          D. R. Hunt
                                Date            01/05/81

                                Purpose         To blink an asterisk on  and  off  the
                                                screen  at  a  rate  determined by the
                                                contents of the B register.          \

                                        ORG 0D00H

  0038                          RDEL    EQU 0038H
  09E2                          SCREEN  EQU 09E2H

  0D00    21 09E2                       LD HL,SCREEN    ; Set HL to point to screen
  0D03    36 2A                 LOOP1:  LD (HL),"*"     ; Put an `*' at (HL)
  0D05    CD 0D0F                       CALL DELAY
  0D08    36 20                         LD (HL)," "     ; Put a `space' at (HL)
  0D0A    CD 0D0F                       CALL DELAY
  0D0D    18 F4                         JR LOOP1

  0D0F    06 C8                 DELAY:  LD B,200        ; Prime the delay counter
  0D11    FF                    LOOP2:  RST RDEL        ; Call the delay in monitor
  0D12    10 FD                         DJNZ LOOP2
  0D14    C9                            RET

                                        END

No  Fatal error(s)
```

Now, I've cheated, I don't intend to repeat the mistake I made in part 3, so I've used an assembler. Now it's not one of the ones you can buy for a normal Nascom, it's disk based, and has some special features. The only reason I've used it instead of my ZEAP, is that my Nascom is all set up for disks, and unlike Mr. Bowden (issue 3 page 27), my Nascom is not multi-mapped, and it's one hell of a hassle to set it all up for ordinary NAS-SYS working. So I must explain some of the differences as well as give a brief run down on assemblers in general.

So, an assembler is a program, it takes the mnemonics you feed it, and turns them into the appropriate machine codes. Same as looking the codes up in the book, but faster. Assemblers can be very very fast, (some are just fast), and will beat writing it all down on your Woolies Jumbo pad any time. They have one major advantage over doing it by hand, (apart from speed). Lets suppose we wanted to add one extra line right in the middle of a program, all the absolute jumps and all the calls after the insertion would have to be changed, because all the absolute addresses would have been moved up by the insertion. In a large program these might run into a hundred or more. It's a mind bending chore doing it by hand (although all of us who had Nascoms before the assemblers came along, managed it). Now, see the advantage of labels for calls and jumps, you see, until it's time to actually assemble the program these aren't absolute addresses, they are just locations -- well -- er, labeled. Do you see what I am getting at? Inserting the odd line here and there doesn't change the label, it simply changes its final location at assembly time. Everything is referenced to the labels, not absolutes, so insertions (or deletions) don't matter.

Two more fields have appeared, the `address' field and the `object' field. The address field contains the address of the first byte of the instruction. The object field contains the byte or bytes which make up the particular instruction starting at the the address given, and for subsequent addresses to the start of the next instruction

Now to the assembled program itself, I've changed some of the label names. The main program loop has been called LOOP1, START would have been inappropriate, as the loop no longer goes back to the beginning of the program. Having called the main loop LOOP1, then it makes sense to change the label for the delay loop to LOOP2.

A `quirk' of this particular assembler is that it requires a colon at the end of the label (to tell it where the label ends), except when the label is followed by an assembler directive (EQU or ORG). Some assemblers follow this convention, some don't!! This assembler allows an assembler directive called `.Comment', and this allows me to write a chunk of descriptive text about the program without having to put a semicolon at the start of each comment line. Notice that semicolons are still used to separate the mnemonics from comments though, it's only provided for convenience. This assembler likes another assembler directive, an `END' at the end of the program, some assemblers will chuck this out as an error.

See that I defined the characters I want to place on the screen as actual characters enclosed in quotes. Most assemblers will allow this, and it a nice practice to do this when you actually mean `Load A with a character', as opposed to `Load A with a variable'. (To the casual glance, LD A,2AH could mean either). Assemblers will usually allow decimal numbers to be entered, and they will automatically convert them to HEX for you. Look where I load the B register in the DELAY subroutine. I put LD B,200, and the assembler took it as a decimal number because I left off the `H' suffix. Another thing this particular assembler does (and many others don't) is to print any two byte operands the `right way round', this makes for easy reading of the assembled object code because it prints it the way you `think', but remember, two byte operands are always entered low byte first, so the listed code this particular assembler prints is not exactly as it would appear when placed (or automatically assembled) in memory.

So that's an assembler. The little counter at the top of the screen is down to 600 odd, so this is where it's got to finish. It's quite likely that there will be plenty of assembly listings in this issue. See if you can read them, by looking at the labels. You don't have to think too hard about what the code is doing (will that be the next part of this never ending saga, I wonder), but given that the programmer has thought carefully about the structure of the program and chosen his labels wisely, it should make some sort of sense. The moral of this part of the story has been, think of your specifications, and think carefully about your choice of labels. You never know, you might not be the only one who will read your programs one day, so why make life difficult for the other fella?

VIDEO

On the following page is a chart showing the complete Nascom video memory map giving the Hex addresses. This drawing must have taken a great deal of time to prepare and many thanks to the originator. Unfortunately his name has been mislaid, but we'll find it by the next issue.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	
16	0B CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	16
1	08 0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	1
2	08 4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77	78	79	2
3	08 8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	3
4	08 CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	4
5	09 0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	5
6	09 4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77	78	79	6
7	09 8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	7
8	09 CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	8
9	0A 0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	9
10	0A 4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77	78	79	10
11	0A 8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	11
12	0A CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	12
13	0B 0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	13
14	0B 4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77	78	79	14
15	0B 8A	8B	BC	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	15
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	

Sound & Music

Programmable Sound Generator

a review by Kevin Smith

Source: Easicomp Ltd., 57 Parana Court, Sprowston, Norwich, NR7 8BH. COST 43.00 + VAT.

This unit uses the now ubiquitous AY-3-8910 chip, which can make all manner of weird noises, and comes complete with software on tape to do just that! Also provided are programs to examine and modify the internal registers of the AY-3-8910, and to test that it is fully functional. For those unfamiliar with this chip, it contains 3 square wave tone generators, a noise generator, mixers and amplitude controls, an envelope generator, and two TTL- compatible I/O ports, all of which are controlled via sixteen 8-bit registers. These are accessed via ports 2 and 3.

The unit is supplied built on a 8" x 3" double-sided fibreglass pcb, which plugs into the Nasbus. Unfortunately, no edge connector socket is supplied, and there are no daisy chain connections between lines 16 and 17, 19 and 20.

An onboard amplifier, based on a LM386, powers a 1/4W. speaker, which is more than adequate, but a 5-pin DIN socket is also provided for output to your hi-fi music centre.

Several pages of documentation are supplied, consisting mainly of photocopies of the I.C. manufacturer's information, which is pretty comprehensive, although most of the examples given are for a 1-79MHz clock. This encouraged some fruitful experimentation. No circuit diagram is supplied.

In conclusion, I found the unit easy to use, although perhaps over-priced. Certainly, the addition of sound adds another dimension to your programs.

Music Board

a review by Ian Henderson

Source: BBF Engineering, 28 New Road, Melbourn, Royston, Herts.

Price: 21.65 built and tested including VAT.

The Music Board is a small board (5" x 3") built on a single sided fibreglass PCB. Connection to the Nascom PIO is made via a 14 pin socket in the centre of the board. Output is via a 5 pin DIN socket that can be fed to an audio amp, or through, for example, a radio/cassette or your hi-fi.

The Music Board is basically a tone generator giving 8 octaves of 12 notes. A variable resistor must be set to tune the master oscillator against a reference such as a tuning fork - alternatively a scope may be used. Once set the board is simplicity itself to operate. Seven bits of the PIO are used, giving 128 possible output codes, 00 to 7F. The first nibble selects one of the eight octaves, the second, one of the twelve notes. The remaining four codes in each sixteen bytes are `silent'. As the PIO latches the data written to it, any note set will continue until the PIO contents are changed.

As only one PIO port is used, two cards could be connected to the Nascom. In this case one of the oscillators is used to drive both cards. The effect of two cards together should be `interesting'.

Documentation is thorough, with good descriptions and examples. To sum it up, effective and economical.

Add ons

This is the first in a series of articles where we will be asking manufacturers of Nascom compatible "goodies" to outline their product range. In this issue we have details from Gemini Microcomputers Ltd.

Gemini is a very young company, started only some 7 or 8 months ago and yet they already have a very comprehensive range of products available, with several more exciting items about to be introduced. MD of Gemini is John Marshall, founder and ex-MD of Nascom, and Technical Manager is Paul Greenhalgh, an ex-Nascom Engineer. Perhaps this explains the company's strong interest and capability in producing Nascom compatible product!

GEMINI MICROCOMPUTERS

Existing Products

EPROM/ROM board

With an increasing range of firmware becoming available for the Nascom, users were rapidly running out of EPROM sockets and Nascom 1 owners were eager to find an easy way to add the BASIC ROM to their systems without recourse to surgery, so an EPROM/ROM board seemed an obvious omission from the Nascom product range that had to be put right!

The Gemini G803 EPROM/ROM board has 4 banks of 4 sockets for EPROMS. Each bank can take either multi-rail 2708s (1K x 8) or single rail 2716 (2K x 8) EPROMS, although each bank must contain only one type. Flexible decoding allows each bank to be located on any 4K address boundary. There is also a further 24 pin socket to take the MK36271 8K BASIC ROM. Any memory on the card produces a RAMDIS signal on the bus when it is addressed. This gives the EPROM or ROM priority over any RAM at the same address.

The card also contains a wait-state generator which, when enabled (a link option), is only activated whilst the card is being accessed. This means that a Nascom 2 owner can move all of his EPROMS and BASIC ROM onto the card and switch off the wait states on the N2. (I know many N2's will run without waits anyway, but not according to the BASIC ROM spec!)

Finally (yes there is more) the card supports the Page Mode scheme. What's that? Well, without going into great detail, (perhaps someone will write INMC80 an article?) it allows you to have more than 64K of memory connected to your system (up to 256K in fact) although only 64K can be accessed by the Z80 at any time - sorry, no 200K Startrek!

Disk System

After the EPROM board we started to think about disk systems, but we weren't the only ones! One day Dave Hunt said "we've got a disk controller going, do you want to see it?" After seeing the card (of course we accepted his offer) we asked how it would be sold and were told "as a kit". Having previously considered all the implications of a disk controller kit, drives, PSUs, interconnects, software etc. etc. we shuddered, and, as regular INMC80 readers already know, decided to produce a complete unit based on the Henelec controller card. A power supply and case were designed, a CP/M licence obtained from Digital Research, Pertec double sided drives selected and, with the addition of the Henelec controller card, the whole lot built and tested.

The Gemini G805 disk system connects to the Nascom PIO and is available with one or two drives and CP/M or D-DOS, a simple operating system. The popularity of the system has lead to Business & Leisure Microcomputers introducing DCS-DOS, a much more sophisticated version of D-DOS with file handling for BASIC, ZEAP, Nas-Pen and machine code files.

Supermum

Our attention now turned to the Nascom 1. Expansion of this had always been a messy business as, when Nascom 1 was originally designed, it wasn't really expected that anyone would wish to add anything! All owners of expanded Nascom 1s will no doubt painfully remember soldering 43 way ribbon cables, and the dreaded memory plague.

The Gemini G806 Supermum is a 12" x 8" board that sits back to back to the Nascom 1. It provides a 5A PSU, 5 slot motherboard, and a buffer section. The buffer section provides reset jump facilities and gates Reset with M1 - something the Nascom buffer board should have done but didn't. Connection between the Nascom 1 and Supermum is via a mini 43-way motherboard.

64K RAM Board

With the advent of disk systems peoples greed (and need) for more and more memory was rapidly growing. The obvious solution - a single board giving a full 64K of RAM.

The Gemini G802 `RAM64' is a RAM board kit, available in 16, 32, 48 or 64K sizes. The bus `RAMDIS' signal, giving priority to EPROM/ROM in the system, means that a `RAM64' board of any size can be used without any data clashes. Additionally a page mode option is available (see EPROM/ROM board) allowing four cards to be added to the system. Although the Page Mode System is relatively unexplored as yet, there is a lot of potential for, for example, phantom disk drives with instant access!

`RAM64' is a full 4MHz board with flexible address selection to 4K boundaries. The board is also available built and tested fully populated (64K) and complete with page mode, at a very silly price. (Cheaper than some of the kits!)

Miscellaneous

Other existing Gemini products are the Gemini G807 3 amp power supply, Gemini G601 Reset jump kit (fitted to the Nascom buffer board to give `reset jump' to any 4K boundary) and Gemini G808 EPROM programmer. This latter item, produced in conjunction with Bits & P.C.'s., connects to the Nascom PIO and can program multi-rail 2708 EPROMs or single rail 2716 EPROMs. Two low insertion force sockets are provided for `Donor' and `Recipient' EPROMS. Software is supplied on tape (N2 format) and can program from the `Donor' or RAM, and read and verify EPROMs.

NEW PRODUCTS

About to be launched by Gemini are three new 8" x 8" boards, two of which can be used with Nascoms.

Video Card

There has long been a requirement for an 80 x 25 video card for professional applications and for use with disk systems with CP/M, where many software packages expect a screen width of at least 72 characters. The trouble with a screen of this size is that it requires a 2K block of memory to be located somewhere and, if the screen routines are going to support a wide variety of functions, getting on for another 2K of memory to hold the software. The solution then, with this card, was to make it I/O mapped and intelligent!

The Gemini G812 Intelligent Video Controller (IVC) card is Z80A microprocessor controlled. Two screen formats are supported by the card - 80 x 25 crystal controlled, plus an adjustable dot clock to give a second format (set to 48 x 25 as supplied.) Output from the card is a 1V peak to peak composite video signal for driving a monitor - with 80 character wide displays, TV sets are not recommended.

The IVC occupies three I/O ports of the host system. Port B1 is for bidirectional data transfer, B2 provides handshake signals, and port B3 is used to reset the card independantly of the system reset. The character set on the card gives

128 characters in EPROM, and 128 in RAM. Single commands to the IVC card can define the 128 RAM characters as the inverse of the main set or as block graphics characters, giving a resolution of 160 x 75 blocks. Furthermore, byte patterns may be sent to the card to define your own characters.

The IVC card supports a wide variety of commands (sent as control codes or ESCAPE sequences.) These include insert/delete character, insert/delete line, clear to end of line/screen, lock a portion of the screen from scrolling, invert/blank the screen, set, test or reset a block graphics cell, return a line from the screen (yes, there is two way communication!), plus lots more.

Software to drive the card from the host computer is very simple. For owners of the Gemini G805 disk system, Richard Beal has already written a new version of SYS that supports this card (and also supports screen editing within CP/M!!). Nascom owners only need enter a short 'U' routine - however it must be pointed out that any program that is expecting the screen to be 48 x 16 and memory mapped at 0800-0BFF will not automatically adjust itself to the new screens' format and position!

Disk card

Although the Henelec controller card used in the Gemini G805 disk system is very good, it does have several limitations - it is only single density, only supports up to 3 drives, and runs via the PIO.

The Gemini G809 FDC card is an 8" x 8" bus card. It supports Pertec FD250 (5.25",48TPI,DS), Micropolis 1015 (5.25",96TPI,DS) and Pertec FD514 (8") disk drives and can control up to four drives of the same type. Up to 8 drives (2 boards) can be used in the same system. Density is either single or double, selected under software control.

CP/M 2.2 will be available to support up to 4 Pertec FD 250 drives with the Nascom. Using double density and 512 byte sectors this gives 350K per drive (formatted)! The software also supports SD Systems format to allow transfer of data to/from the Gemini G805 disk system.

The Other Card

With all of the above cards, Gemini have the complete range for a very powerful and flexible system, bar a minor item - a CPU card! Hence we have produced an 8" x 8" card which is just that. Although not of immediate interest to Nascom owners, this brief description completes the details of our product range.

The Gemini G811 CPU card utilises (yes, you've guessed it) the Z80A processor at 4MHz. There are optional wait states, and reset jump to any 4K boundary. Serial I/O is via a WD8250 UART. This gives programmable band rates, stop bits etc. Input/Output to the UART can be switched between the RS232 and cassette interfaces under software control. The RS232 interface supports modem control and handshake signals; the cassette interface is 1200 band CUTS. A Z80A PIO is provided for parallel I/O.

Provided on the board are four 'bytewyde' 28 pin sockets - these sockets will accept a wide variety of RAMs/EPROMs/ROMs from the 1K x 8 static RAMs up to the latest 32K x 8 ROMs. (32K x 8? Yes!) The on-board memory block can be switched out of the memory map under software control, allowing a 64K RAM system to be fully realised.

The monitor is something special! Living at the top of memory (F000), to programs it pretends it is CP/M, plus it has a range of useful commands. You don't have to have disks, but if you decide to upgrade to them later you can take all your software with you - and the monitor already contains the necessary bootstrap routines to run the Gemini G809 FDC card. This is real expansion without redundancy!

Availability

All of the items in the first part of this article are real products and available now. The last three items are finding their way into production as I write (they'll only be available built and tested) and may well be on demonstration (or even available) at your distributor by the time you read this.

Gemini Microcomputers does no direct selling, and the above items are ONLY available (in the UK) from the following companies: Bits & P.C's., Business & Leisure Microcomputers, Electrovalue, Henry's Radio, Interface Components, Target Electronics.

IMPRINT

The IMPRINT: a review.

Rory O'Farrell

The IMPRINT is a new operating system for the IMP printer. It comes in the form of a new EPROM, which is substituted for that provided as standard in the IMP. With the addition of this EPROM, your IMP has a number of important new facilities.

Firstly it has the facility, under software control, to print either normal sized characters, or double sized characters, which it prints 40 to the line. All characters can now be printed unidirectionally or bidirectionally, by sending the correct control code to the IMP. The IMP can also interpret Control I as a tab character, and steps across the page to the next multiple of 8 columns, thus becoming compatible with the Tab function under the CP/M disk operating system. Control L (code 0C) is recognised as a form feed command, and causes the IMP to advance the paper by six lines.

Connection of the IMP to your system remains unchanged, the Busy line behaving as previously. The new system immediately makes the IMP sound different. The reason is not hard to find. Examination of the manual discloses that the IMP is effectively printing at twice the speed (unless you have been changing the preset power up options). Gone is the change to unidirectional printing when there are more than forty characters on a line. Why doesn't the head overheat? The manual explains that the head is rated for continuous printing, and the only character which caused occasional difficulty was the rubout character - the white block. This character has now been changed to an inverted ? , and in consequence 80 characters per line can now be printed in bidirectional mode. This option can be disabled under software control, so that when best alignment is required in the type it can be obtained by selecting the unidirectional mode.

If the Line Feed button is held down as Reset is pressed on the IMP, it will immediately set to, and print its character set four times, both in standard and double width. When you have installed the EPROM, this is the easiest way to check that all is as it should be.

What happens if you have an application which demands heavy lines across the page, as it might be, lines of the old 7F white Block? All is not lost! The IMP has one surprising addition:

A HI RES GRAPHICS FACILITY!!!

This facility, in addition to the previously mentioned facilities, make the IMP unsurpassed value for money! It now acquires the facility to plot hi res graphics all across the page, on receipt of a 1F code (control _). When it receives this character, it prints out what is left of the buffer, goes to the lefthand side of the print area, and feeds to a new line. It then proceeds to gather the next 760 Bytes, and interpret the 7 least significant bits of each byte as instructions for each print needle (bit 6=top dot, bit 0=bottom dot). When it has these 760 bytes (they take a fair while to send - over 6 secs at 1200 baud) it prints the line. In printing the line, it takes account of the head specification, which says that a dot can not be printed in two columns in succession, and if it finds such an illegal condition, it resets a dot or dots. To indicate that it has interfered with your data, it turns on the Red Error LED, and turns it off at the end of the line. This allows you to plot 380 dots across the width of the page at any one time, and you have 760 columns at your disposal. When the graphics line is finished, the IMP feeds by the amount of the line only, so that another graphics line can be printed below and immediately contiguous. In this way it will be possible to plot quite fine resolution graphs, displays and pictures. I have in mind using the IMP to plot soil resistivity data from Archaeological surveys, which I had been asked to process. The plot from the IMP will very quickly give an indication of the result, without having to go to the trouble of doing a detailed manual plot.

It would not be out of place to list here the various control codes and their functions:

Code 02 (CTRL B) sets bidirectional print. This mode gives maximum throughput - say 65 chars per second!

Code 03 (CTRL C) sets unidirectional print. This gives best alignment for columns of figures. The command is obeyed immediately on receipt.

Code 04 (CTRL D) sets double width printing. This character is queued in the buffer, and only takes effect in its appropriate place. It is automatically reset at the end of a line.

Code 05 (CTRL E) sets single width characters. All lines start out in this condition, so you must reindicate by a CTRL D if you want double width after each line feed.

Code 08 (CTRL H) Backspace. This will only backspace within a line. It will not pass a Line Feed (0A), a Carriage Return (0D) or a Form Feed (0C).

Code 09 (CTRL I) Horizontal Tab. This will cause the print head to print spaces until it is at the next column which is a multiple of 8.

Code 0A (CTRL J) Line feed. This prints any preceeding characters in the buffer, and advances the paper by one line.

Code 0C (CTRL L) Form Feed. This causes 6 line feeds in succession.

Code 0D (CTRL M) Carriage return. This causes any preceeding characters in the buffer to be printed, and if the strap option is set in the IMP, causes a line feed.

Code 1F (CTRL _) Graphics Mode. Prints out the Buffer, and puts head at lefthand side on a fresh line. Then it interprets the next 760 Bytes as a dot pattern, and prints them across the page. At the end of the line, it feeds enough for another graphics line to join below the last one, and the IMP returns to the normal type mode. It is necessary to reinvoke the graphics option at the start of each line of graphics.

These are the options, with the addition of the self test on reset. Without any doubt, the IMPRINT is a very worth while addition to any IMP. The alteration to the rub out character to allow continuous bi-directional printing alone allows the throughput of the IMP to be nearly doubled (38 chars per sec to 65 chars per sec from my measurements!) and may put off the evil hour when a faster printer becomes essential. Those who have had to wait two hours for the latest edition of a printout will know what I mean (and sympathise?).

The IMPRINT comes with a well commented and well printed set of documentation. There is one error in it that I found. The reference on page 3 to code 04 should be to code 02. With my copy, there was no mention of how to put the EPROM in position. I hope that this will be rectified shortly. To open the IMP, you will need a 3/32" Allen hex key. Unplug the IMP from the mains (dying is something you usually only get one chance at!) and open the two little black studs on each side with the Allen Key, which can be got in any good toolshop or engineers providers. Very carefully, slip the case off, and lay it on its back beside the machine. It is still connected to the IMP by a loom of wire, so try not to put too much tension on these wires. Then, with a long handled fine screwdriver, working from the front of the machine, insert the screw driver under the EPROM, which is about 1/3 of the way across the machine, pointing front to back. Very gently, you can twist the blade to free this EPROM, until it can be lifted out. It should be held only by the ends, not by the pins. The new EPROM can be placed in position, holding in the same manner, with the orientation dot or dimple on the end facing the print mechanism, and pushed down gently into position. The old EPROM should be placed into a piece of antistatic foam, or foil wrapped airfoam, and kept until a use arises for it. It could be reprogrammed with a new character set for an N2, or an N1 with Econographics. Having inserted the EPROM, now is the time to lubricate the innards of the IMP. A drop of oil on the polished guide rail, the frame edge, the helical cam, and the ribbon driver are all that is required. Now put the top back, screw in the screws, thread the paper, and plug in. Turn on, hit reset, while holding down the Line feed button. The machine will enter self test mode, and print away busily for about 25 seconds. The rest is up to you!

The IMPRINT is available from Interface Components (and possibly other Nascom distributors ?), costing #30 plus VAT. The author, David Parkinson, is to be congratulated on having got so much into so little.

Game Idea

PIRANHA FISH

Have you ever come home from a hard day at the 'Grunge Foundry', had dinner, thrown the Radio and TV Times on the floor in disgust, walked into the other room, switched on the computer, and then like the organist in the 'Lost Chord' typed idly at the keys. Then had this brilliant idea for a program to write, only, within a twinkling the inspiration has vanished. No? Oh well, forget it. How about those of you who are quite capable programmers, but can never think of anything to program? May be it's Sunday afternoon, and having mowed the kittens, you've nothing better to do. Either way, I was looking through a manual for 'Tiny-C' when I found the following demonstration example, and thought, "There's a marvellous program for the INMC80, given someone with the time to write it, and a sufficiently warped imagination." Well for all of you with warped minds and nothing better to fill the time, here are the rules, they make hilarious reading in themselves.

You are on safari, and your party consists of the following unlikely people.

2 Cannibals
2 Big-game hunters
1 Doctor
1 Nurse and
3 Missionaries

You have arrived at a river 100 yards wide, filled with hungry piranha fish. There is a leaky canoe by the shore, and as you must cross the river, that is the only transport available. Worse, it will hold at the most only four people. The canibals paddle the best, followed by the hunters, the doctor, the nurse, and lastly, the missionaries who are notoriously weak. You have to decide who will take the canoe each trip back and forth, thereby getting the whole party across with the minimum of carnage.

The doctor can attend major and minor wounds, unless he himself is wounded. The nurse can attend minor wounds, or if the doctor is wounded but on the same shore as the nurse, the nurse can attend to major wounds under the guidance of the doctor.

The speed of the canoe is the average of the paddling strengths of the people in the canoe. A speed of 100 units is required to paddle to opposite shore just as the canoe sinks. The initial paddling strengths are:

Cannibals	120
Hunters	90
Doctor	70
Nurse	50
Missionaries	40

Strengths should be multiplied by the following factors for unhealthy paddlers:

Minor wound, attended	0.9
Major wound, attended	0.8
Minor wound, unattended	0.8
Major wound, unattended	0.7
Dead	0.0

During the trip across the river, certain events will happen with the following probabilites:

The Canoe fills with water at a predetermined rate.
During each yard (speed divided by 4) of the trip a single piranha fish will jump into the boat with a probability of 0.25. The fish will always select a random toe. Cannibals will always spear the fish, but half the time they will make a hole in the canoe by doing so. Hunters always panic and capsize the boat. The Doctor is quick half the time avoiding

the piranha, and panics half the time. The Nurse always panics, but half the time she is calmed down, and half the time she jumps (alone) out of the boat and must swim ashore. When the boat capsizes, everybody must swim. Dead people always float to the correct shore, and incredibly, the canoe gets there too.

When swimming the events that follow may occur to each person in the water, individually:

Dead people always float ashore
Live people make it ashore unscathed half the time. The other half, they aquire minor wounds (probabilty 0.67) or major wounds (probabilty 0.33). In no case do they come out of the river healthier than they went in. (Note, these probabilities could be made worse the further there is to swim.)

On shore the victims health may become worse with the following probability:
Healthy people never get worse.
Attended people get worse with a probabilty of 0.11.
Unattended people get worse with a probability of 0.33.
Dead people never get worse.
To get worse means that a minor wound becomes a major wound or a person with a major wound dies.
When a minor attended wound gets worse, it become a major unattended wound.
The 'Worse Health' events should be computed for each person once per canoe trip, whether or not the person participated in the latest trip. So people who were wounded early in the game have more chance of getting worse that people wounded later.

Scoring:

1000	starting maximum points
-100	for each dead person
-30	for each major unattended wound
-15	for each major attended wound
-10	for each minor unattended wound
-5	for each minor attended wound

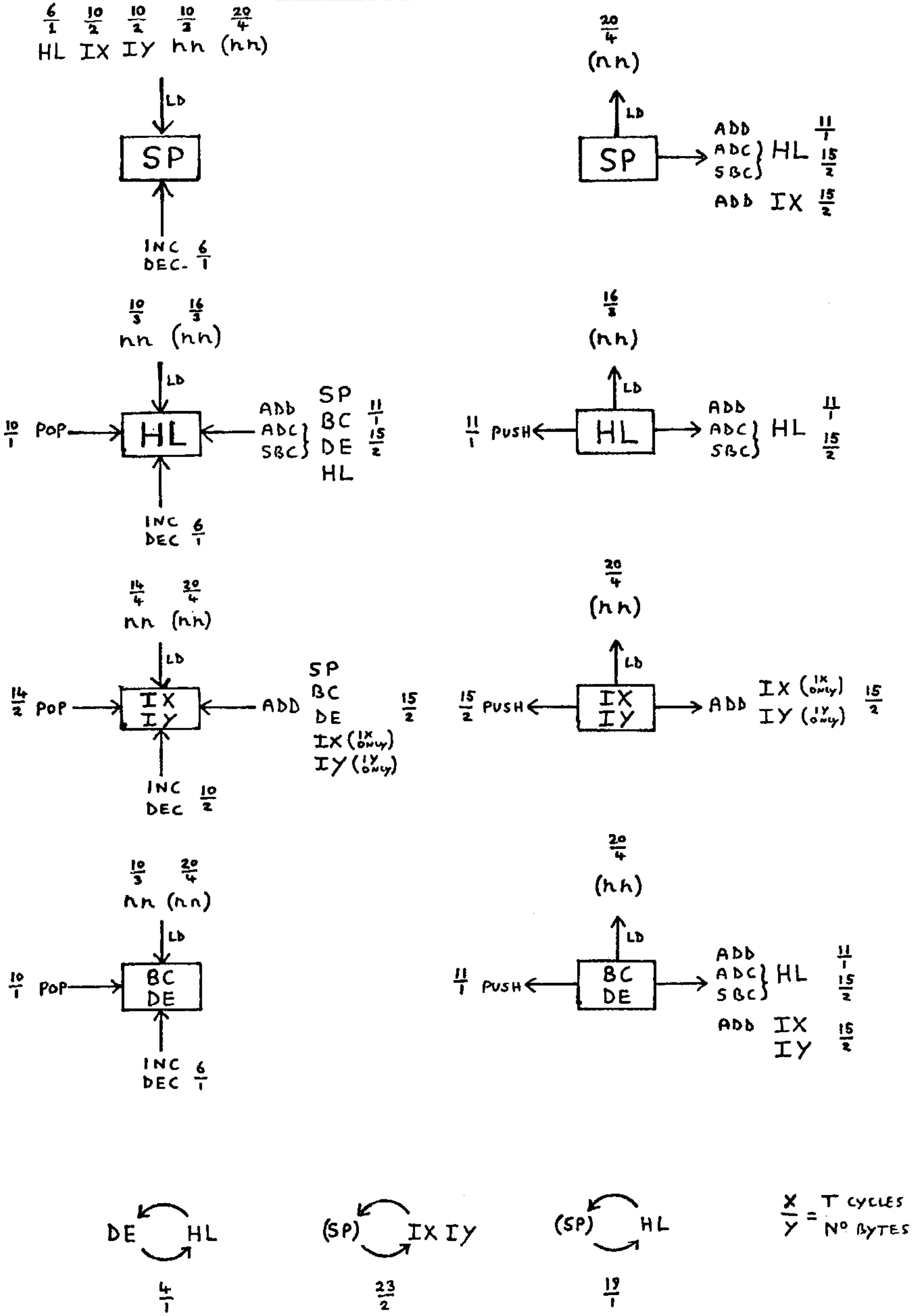
Certain programmers with particularly warped minds might like to modify the program for a 'Maximum carnage game' (minimum score). To do this you need a new rule: On each successive trip you must leave at least one more person on the shore than the previous round trip.

Happy paddling.

Z80 Ops

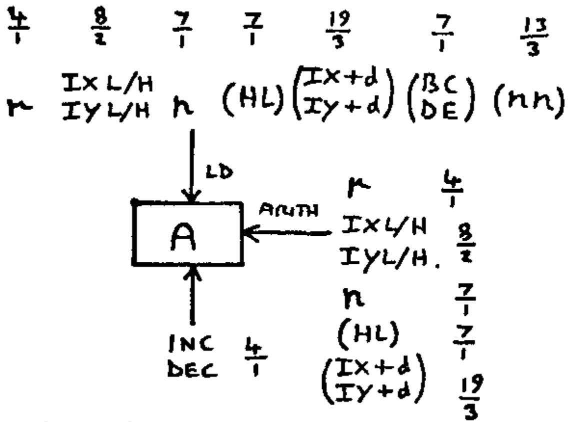
On the following pages are charts submitted by Mr. J. Rollason, to whom many thanks. They show all the 8 and 16 bit operations that can be performed on the Z80, along with details of the number of T states and number of bytes.

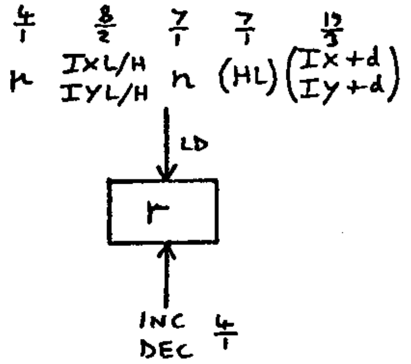
16 BIT OPERATIONS

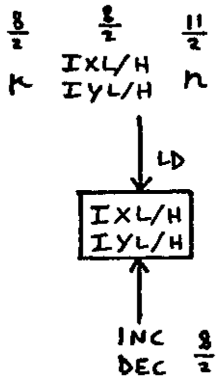


J. ROLLASON

8 BIT OPERATIONS



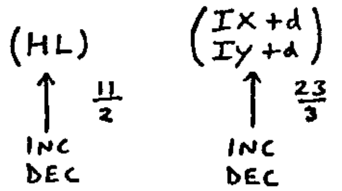


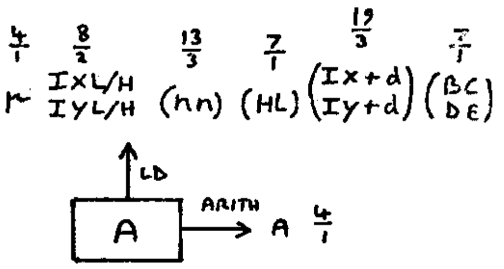


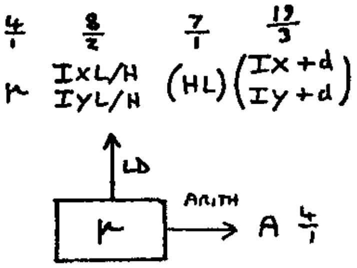
1 BIT OPERATIONS

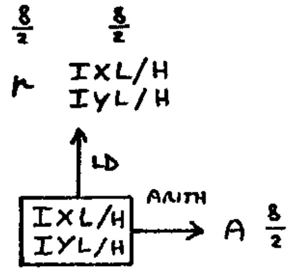
	r	(HL)	(IX+d / IY+d)
BIT	$\frac{8}{2}$	$\frac{12}{2}$	$\frac{20}{4}$
SET	$\frac{8}{2}$	$\frac{15}{2}$	$\frac{23}{4}$
RES	$\frac{8}{2}$	$\frac{15}{2}$	$\frac{23}{4}$
[9 SHIFTS	$\frac{8}{2}$	$\frac{15}{2}$	$\frac{23}{4}$
4 SHIFTS	$\frac{4}{1}$ ON 'A' ONLY]		

NON REGISTER ARITHMETIC









r INCLUDES A, H.L, B.C, D & E

IXL/H REFERS TO IXL & IXH
IYL/H " " IYL & IYH

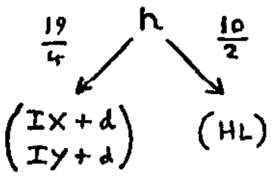
NOTE: IXL/H & IYL/H CANNOT BE USED WITH H & L.
I.E LD H IXL WOULD BE LD IXH IXL.

ALSO: IXL/H CANNOT BE USED WITH IYL/H

THE IXL/H PREFIX IS &DD
THE IYL/H " " &FD

E.G &DD
LD A H ≡ LD A IXH

NON REGISTER TRANSFERS



Strings

STRINGS IN BASIC (AGAIN)

G.T.Klement

After reading INMC NEWS 80/3 , I think you will possibly be interested in a routine which saves and loads string arrays in BASIC.

Since I am a beginner in machine code programming, I am sure there are better solutions so take it for a first step to solve this problem.

Here is a ZEAP file with the source code, so you may easier test this routine. After the ZEAP file I have added a short BASIC program for demonstration.

ZEAP Z80 Assembler - Source Listing

```
                0010 ;  ===== STRINGSAVE V4 ======
                0020 ;              17.3.81
                0030 ;
                0040 ;
                0050 ;        DATA FORMAT  BASIC
                0060 ;
                0070 ; Nam   Dis   Sub DIM+1 DIM+1      L / CHP
                0080 ;
                0090 ;|1|2|  |3|4|  |5|  |a|b|  |a|b|     |A|B|C|D|
                0100 ;|_|_|  |_|_|  |_|  |_|_|  |_|_| ..  |_|_|_|_|
                0110 ;
22FF 0008       0120 RIN     EQU  #8
22FF 000D       0130 CRLF    EQU  #0D
22FF 0028       0140 PRS     EQU  #28
22FF 0030       0150 ROUT    EQU  #30
22FF 0038       0160 RDEL    EQU  #38
22FF 005B       0170 MRET    EQU  #5B
22FF 005D       0180 TDEL    EQU  #5D
22FF 005F       0190 MFLIP   EQU  #5F
22FF 0066       0200 TBCD3   EQU  #66
22FF 0068       0210 B2HEX   EQU  #68
22FF 006B       0220 ERRM    EQU  #6B
22FF 006D       0230 SOUT    EQU  #6D
22FF 006F       0240 SRLX    EQU  #6F
22FF 0070       0250 SRLIN   EQU  #70
22FF 0071       0260 NOM     EQU  #71
22FF 0072       0270 NIM     EQU  #72
22FF 0077       0280 NNOM    EQU  #77
22FF 0078       0290 NNIM    EQU  #78
22FF 0686       0300 LODEND  EQU  #0686
22FF 068A       0310 SAVEND  EQU  #068A
22FF 080A       0320 SCREEN  EQU  #080A
22FF 0C1E       0330 ARG10   EQU  #0C1E
22FF 105A       0340 BEGCLR  EQU  #105A
22FF 10C3       0350 CLRTOP  EQU  #10C3
22FF 10DA       0360 BEGFRE  EQU  #10DA
22FF 10D8       0370 BEGARR  EQU  #10D8
                0380 ;
0CC0            0390         ORG  #CC0
                0400 ;
                0410 ; ******* DRIVER SAVE *********
                0420 ;
0CC0 DF77       0430 DRSAV   SCAL NNOM
0CC2 E5         0440         PUSH HL
0CC3 E5         0450         PUSH HL
0CC4 CDDD0C     0460         CALL FIN
0CC7 C1         0470         POP  BC
0CC8 CD060D     0480         CALL MATRIX
```

```
               1690 ; ******* DRIVER LOAD *******
               1700 ;
0D66 DF77      1710 DRLOD   SCAL NNOM
0D68 E5        1720         PUSH HL
0D69 DF78      1730         SCAL NNIM
0D6B E5        1740         PUSH HL
0D6C CDDD0C    1750         CALL FIN
0D6F DF5F      1760         SCAL MFLIP
0D71 CD060D    1770         CALL MATRIX
0D74 CD7A0D    1780         CALL LOAD
0D77 C38606    1790 EXITL   JP   LODEND
               1800 ;
               1810 ; ==== SBR LOAD ====
0D7A E5        1820 LOAD    PUSH HL
0D7B ED5B1E0C  1830         LD   DE,(ARG10)
0D7F 2AC310    1840 L0      LD   HL,(CLRTOP)
               1850 ; --- In Header ---
0D82 CF        1860 L1      RST  RIN
0D83 FEFF      1870         CP   #FF
0D85 20FB      1880         JR   NZ,L1
0D87 0603      1890         LD   B,3
0D89 CF        1900 L2      RST  RIN
0D8A FEFF      1910         CP   #FF
0D8C 20F4      1920         JR   NZ,L1
0D8E 10F9      1930         DJNZ L2
0D90 CF        1940         RST  RIN
0D91 4F        1950         LD   C,A
0D92 CF        1960         RST  RIN
               1970 ; --- Check Header & Range ---
0D93 13        1980         INC  DE
0D94 12        1990         LD   (DE),A
0D95 13        2000         INC  DE
0D96 13        2010         INC  DE
0D97 B9        2020         CP   C
0D98 202B      2030         JR   NZ,L5
0D9A AF        2040         XOR  A
0D9B B9        2050         CP   C
0D9C 2827      2060         JR   Z,L5
               2070 ; --- Calc. Chr.pointer ---
0D9E 47        2080         LD   B,A
0D9F ED42      2090         SBC  HL,BC
0DA1 22C310    2100         LD   (CLRTOP),HL
               2110 ; --- Test for CLEAR Range ---
0DA4 D5        2120         PUSH DE
0DA5 E5        2130         PUSH HL
0DA6 ED5B5A10  2140         LD   DE,(BEGCLR)
0DAA ED52      2150         SBC  HL,DE
0DAC E1        2160         POP  HL
0DAD D1        2170         POP  DE
0DAE 3826      2180         JR   C,ER2
0DB0 7D        2190         LD   A,L
0DB1 12        2200         LD   (DE),A
0DB2 7C        2210         LD   A,H
0DB3 13        2220         INC  DE
0DB4 12        2230         LD   (DE),A
0DB5 1B        2240         DEC  DE
               2250 ; --- Data in ---
0DB6 41        2260         LD   B,C
0DB7 0E00      2270         LD   C,0
```

```
0DB9 CF        2280 L3      RST  RIN
0DBA 77        2290         LD   (HL),A
0DBB 81        2300         ADD  A,C
0DBC 4F        2310         LD   C,A
0DBD 23        2320         INC  HL
0DBE 10F9      2330         DJNZ L3
0DC0 CF        2340         RST  RIN
0DC1 B9        2350         CP   C
0DC2 C2D10D    2360         JP   NZ,ER3
0DC5 13        2370 L5      INC  DE
0DC6 C1        2380         POP  BC
0DC7 0B        2390         DEC  BC
0DC8 AF        2400         XOR  A
0DC9 B9        2410         CP   C
0DCA 2002      2420         JR   NZ,L7
0DCC B8        2430         CP   B
0DCD C8        2440         RET  Z
0DCE C5        2450 L7      PUSH BC
0DCF 18AE      2460         JR   L0
               2470 ;
               2480 ; ==== LOAD ERROR ====
0DD1 EF        2490 ER3     RST  PRS
0DD2 3F        2500         DEFM /?/
0DD3 00        2510         DEFB 0
0DD4 18EF      2520         JR   L5
               2530 ; ==== OM ERROR ====
0DD6 EF        2540 ER2     RST  PRS
0DD7 4F4D      2550         DEFM /OM/
0DD9 00        2560         DEFB 0
0DDA DF6B      2570         SCAL ERRM
0DDC E1        2580         POP  HL
0DDD E1        2590         POP  HL
0DDE 1897      2600         JR   EXITL
```

ZEAP Z80 Assembler - Symbol Table

```
0C1EH 0330 ARG10      0068H 0210 B2HEX
10D8H 0370 BEGARR     105AH 0340 BEGCLR
10DAH 0360 BEGFRE     10C3H 0350 CLRTOP
000DH 0130 CRLF       0D66H 1710 DRLOD
0CC0H 0430 DRSAV      0CFEH 0850 ER1
0DD6H 2540 ER2        0DD1H 2490 ER3
006BH 0220 ERRM       0D77H 1790 EXITL
0CDAH 0600 EXITS      0CE0H 0640 F1
0CF5H 0750 F2         0CDDH 0630 FIN
0CD2H 0550 H1         0D7FH 1840 L0
0D82H 1860 L1         0D89H 1900 L2
0DB9H 2280 L3         0DC5H 2370 L5
0DCEH 2450 L7         0D7AH 1820 LOAD
0686H 0300 LODEND     0D08H 0950 M1
0D15H 1040 M2         0D1BH 1100 M3
0D24H 1170 M8         0D06H 0930 MATRIX
005FH 0190 MFLIP      005BH 0170 MRET
0072H 0270 NIM        0078H 0290 NNIM
0077H 0280 NNOM       0071H 0260 NOM
0028H 0140 PRS        0038H 0160 RDEL
0008H 0120 RIN        0030H 0150 ROUT
0D30H 1260 S1         0D40H 1420 S2
0D56H 1550 S3         0D5BH 1590 S4
0D28H 1220 SAVE       068AH 0310 SAVEND
080AH 0320 SCREEN     006DH 0230 SOUT
0070H 0250 SRLIN      006FH 0240 SRLX
0066H 0200 TBCD3      005DH 0180 TDEL
```

```
               0490 ; --- Header out ---
0CCB C5        0500        PUSH BC
0CCC DF5F      0510        SCAL MFLIP
0CCE DF5D      0520        SCAL TDEL
0CD0 AF        0530        XOR  A
0CD1 47        0540        LD   B,A
0CD2 DF6F      0550 H1     SCAL SRLX
0CD4 10FC      0560        DJNZ H1
0CD6 C1        0570        POP  BC
               0580 ; --- Save Data ---
0CD7 CD280D    0590        CALL SAVE
0CDA C38A06    0600 EXITS  JP   SAVEND
               0610 ;
               0620 ; ==== SBR FIND ====
0CDD 2AD810    0630 FIN    LD   HL,(BEGARR)
0CE0 ED5BDA10  0640 F1     LD   DE,(BEGFRE)
0CE4 E5        0650        PUSH HL
0CE5 ED52      0660        SBC  HL,DE
0CE7 E1        0670        POP  HL
0CE8 F2FE0C    0680        JP   P,ER1
0CEB 3EB0      0690        LD   A,"0+#80
0CED BE        0700        CP   (HL)
0CEE 23        0710        INC  HL
0CEF C2F50C    0720        JP   NZ,F2
0CF2 3E44      0730        LD   A,"D
0CF4 BE        0740        CP   (HL)
0CF5 23        0750 F2     INC  HL
0CF6 5E        0760        LD   E,(HL)
0CF7 23        0770        INC  HL
0CF8 56        0780        LD   D,(HL)
0CF9 23        0790        INC  HL
0CFA C8        0800        RET  Z
0CFB 19        0810        ADD  HL,DE
0CFC 18E2      0820        JR   F1
               0830 ;
               0840 ; ==== NO VARS ====
0CFE E1        0850 ER1    POP  HL
0CFF E1        0860        POP  HL
0D00 E1        0870        POP  HL
0D01 DF6B      0880        SCAL ERRM
0D03 DF71      0890        SCAL NOM
0D05 C9        0900        RET
               0910 ;
               0920 ; ==== SBR MATRIX ====
0D06 46        0930 MATRIX LD   B,(HL)
0D07 78        0940        LD   A,B
0D08 23        0950 M1     INC  HL
0D09 5E        0960        LD   E,(HL)
0D0A 23        0970        INC  HL
0D0B 56        0980        LD   D,(HL)
0D0C D5        0990        PUSH DE
0D0D 10F9      1000        DJNZ M1
0D0F 221E0C    1010        LD   (ARG10),HL
0D12 E1        1020        POP  HL
0D13 180F      1030        JR   M8
0D15 44        1040 M2     LD   B,H
0D16 4D        1050        LD   C,L
0D17 D1        1060        POP  DE
0D18 F5        1070        PUSH AF
0D19 AF        1080        XOR  A
0D1A 1B        1090        DEC  DE
```

```
0D1B 09        1100 M3     ADD  HL,BC
0D1C 1B        1110        DEC  DE
0D1D BB        1120        CP   E
0D1E 20FB      1130        JR   NZ,M3
0D20 BA        1140        CP   D
0D21 20F8      1150        JR   NZ,M3
0D23 F1        1160        POP  AF
0D24 3D        1170 M8     DEC  A
0D25 20EE      1180        JR   NZ,M2
0D27 C9        1190        RET
               1200 ;
               1210 ; ==== SBR SAVE ====
0D28 3A1E0C    1220 SAVE   LD   A,(ARG10)
0D2B 4F        1230        LD   C,A
0D2C 3A1F0C    1240        LD   A,(ARG10)+1
0D2F 47        1250        LD   B,A
0D30 03        1260 S1     INC  BC
0D31 0A        1270        LD   A,(BC)
0D32 5F        1280        LD   E,A
0D33 03        1290        INC  BC
0D34 03        1300        INC  BC
0D35 E5        1310        PUSH HL
0D36 0A        1320        LD   A,(BC)
0D37 6F        1330        LD   L,A
0D38 03        1340        INC  BC
0D39 0A        1350        LD   A,(BC)
0D3A 67        1360        LD   H,A
0D3B C5        1370        PUSH BC
               1380 ; --- Data out ---
0D3C AF        1390        XOR  A
0D3D FF        1400        RST  RDEL
0D3E 0605      1410        LD   B,5
0D40 DF6F      1420 S2     SCAL SRLX
0D42 3EFF      1430        LD   A,#FF
0D44 10FA      1440        DJNZ S2
0D46 43        1450        LD   B,E
0D47 AF        1460        XOR  A
0D48 BB        1470        CP   E
0D49 7B        1480        LD   A,E
0D4A DF6F      1490        SCAL SRLX
0D4C DF6F      1500        SCAL SRLX
0D4E CA5B0D    1510        JP   Z,S4
0D51 DF6D      1520        SCAL SOUT
0D53 060B      1530        LD   B,11
0D55 79        1540        LD   A,C
0D56 DF6F      1550 S3     SCAL SRLX
0D58 AF        1560        XOR  A
0D59 10FB      1570        DJNZ S3
               1580 ; --- Check Pointers ---
0D5B C1        1590 S4     POP  BC
0D5C E1        1600        POP  HL
0D5D 2B        1610        DEC  HL
0D5E AF        1620        XOR  A
0D5F BD        1630        CP   L
0D60 20CE      1640        JR   NZ,S1
0D62 BC        1650        CP   H
0D63 20CB      1660        JR   NZ,S1
0D65 C9        1670        RET
               1680 ;
```

And now the BASIC program. There is one important rule to load the array:

The array is stored in the free space below the last valid string in the string garbage area. So one has to force the garbage collect routine with "PRINT FRE(0)" after the "USR(0)" command for loading the array.

```
1 REM --- STRINGCSAVE  ---
2 REM  G.T.KLEMENT          17.3.81
9 REM
10 REM ERROR MESSAGES:
11 REM SAVE    Error  Variable not defined
12 REM                Program returns to
13 REM                BASIC.
15 REM LOAD    ?      Read error.
16 REM                a)LEN :Block not
17 REM                       loaded
18 REM                b)DAT :Block
19 REM                       loaded
20 REM       OMError CLEAR Range to small
21 REM                Read sequence interrupt
22 REM       no MSG  Variable was improper
23 REM                dimensioned.
24 REM
100 REM****************************************
110 REM* POKE machine code                    *
120 REM*                                      *
130 REM*                                      *
140 REM* RAM    from 3264 to 3552 = CCO-DEO   *
150 REM* "SAVE" :  DOKE 4100,3264    <<<<<<<< *
160 REM* "LOAD" :  DOKE 4100,3430    <<<<<<<< *
170 REM* Call routine with USR(0)             *
180 REM* Variable (default) : DO$             *
190 REM* Define Variable :                    *
200 REM* POKE 3315,ASC("1.CHR")      <<<<<<<< *
210 REM* POKE 3305,ASC("2.CHR")+128  <<<<<<<< *
220 REM****************************************
230 REM
240 REM --- Poke to memory ---
250 RESTORE300
260 FOR I=3264 TO 3552 STEP 2
270 READ J: DOKE I,J: NEXT
280 RETURN
290 REM ******* DATA ***********************
300 DATA30687,-6683,-8755,-16116,1741,-15091
310 DATA24543,24031,18351,28639,-1008,-12863
320 DATA3368,-30013,10758,4312,23533,4314
330 DATA-4635,-7854,-270,15884,-16720,-15837
340 DATA3317,17470,9150,9054,9046,6600
350 DATA-7656,-7711,-8223,-8341,-13967,30790
360 DATA24099,22051,4309,8953,3102,6369
370 DATA17423,-11955,-20491,2331,-17637,-1248
380 DATA8378,-3592,8253,-13842,7738,20236
390 DATA7994,18188,2563,863,-6909,28426
400 DATA2563,-15001,-81,1286,28639,-194
410 DATA-1520,-20669,31675,28639,28639,23498
420 DATA-8435,1645,30987,28639,4271,-15877
430 DATA11233,-16977,-12768,8380,-13877,30687
440 DATA-8219,-6792,-8755,-8436,-12961,3334
450 DATA31437,-15603,1670,-4635,7771,10764
460 DATA4291,-305,8447,1787,-12541,-2
```

```
470 DATA-3040,-1776,20431,5071,4882,-18157
480 DATA11040,-18001,10024,-4793,8770,4291
490 DATA-6699,23533,4186,21229,-11807,9784
500 DATA4733,4988,6930,3649,-12544,-32393
510 DATA9039,-1776,-17969,-11838,4877,3009
520 DATA-18001,544,-14152,6341,-4178,63
530 DATA-4328,20463,77,27615,-7711,-26856
540 DATA16873
```

G.T.Klement
Troststrasse 100
1100 WIEN
AUSTRIA

IMPERSONAL

Scurrilous Musings

by Guy Klueless

It was recently announced that Nascom has been bought by Lucas Logic (pity no-one remembered to invite me to the celebration), and this started my mind thinking about all the benefits that such a giant could offer Nascom. For instance, have you ever had trouble connecting things to all those tiny little pins scattered all over the N2 board? Perhaps we'll now see these changed for nice sensible quarter inch Lucar connectors!

From the advance copy I've seen of this issue of the newsletter, it seems that Gemini gets more than it's fair share of space. Well, we all know from our basic astrology that Gemini is (are?) the 'twins'. But do you know the twin's names? Yes of course, it's Castor and Pollox. So, from now on you know that Gemini is Castor and Pollox. Please make sure you've got the right teeth in if you're going to say that quickly. (Reminds me of all the permutations of the Accles and Pollock ads we used to have on the London tube.) Any (printable) suggestions please?

On the subject of Gemini, did you know that a soon to be announced Hewlett Packard computer (using twin 16 bit processors) has been dubbed with the 'in-house' name of 'The Gemini Computer'. Reports suggest that Gemini aren't pleased, and will be even less pleased if the advertising copywriters latch onto the name. Me? I didn't say a word, did I?

We hear reports of Nascom power supplies starting to get embarrassed by the ever increasing loads being imposed by the endless goodies users will insist on fitting to their toys. There are red faces in the Henry's camp, because the genius who designed their interpretation of the theme included a spurious 0.68R resistor in the unregulated 5V rail (to limit regulator dissipation when lightly loaded, they say), which means that their PSU exhibits spectacular hum bars on the display when pushed. Answer, take it out. Anyway, to overcome the shortage of juice, there's always Nascom's 8 amp PSU, assuming you can afford a second mortgage ..... or Aztec do a very small neat switch mode 6 amp PSU at the 100 off price of 52.00. Now that price is interesting when you consider Tangerine (you know, those bright orange things) are offering a switch mode 6 amp PSU with Aztec written all over it, for the one off retail price of 49.00. Makes you wonder what sort of deal they came too.

Overheard in a shop recently, the remark made by a cash paying customer buying a disk system. In reply to his wife's comment that 700 quid was a lot to shell out for such a small box, he said, "I know dear, but it'll make a lovely bookend!".

And in conclusion, the postscript to a letter from a member suggests that if your Nascom is poorly, then you should try Lucas-aid!!

How Lucas Logic, dashing young scion of a powerful British family, perceived the true excellence of the splendidly developed Nascom, rescued her from penury, bestowed on her the family name and, by so doing, vouchsafed her shelter and a future rich with prospects

nascom Microcomputers is now a Division of Lucas Logic Limited.

Lucas Logic

Lucas Logic Limited, Welton Road, Wedgnock Industrial Estate,
Warwick CV34 5PZ. Telephone: Warwick 497733.

NASBUS COMPATIBLE NEWS

More great deals from 6 Nascom Dealers

A NASCOM-2 BASED SYSTEM FOR £1549 + VAT

The proven Nascom-2 microcomputer can now be bought as a complete system from £1549 + VAT. For this price you get the Nascom-2 kit, 16K RAM board kit, Kenilworth case with 2 card frame, Centronics 737 printer, 10 inch monitor, and the Gemini Dual Drive Floppy Disk System. The CPU and RAM boards are also available built – the additional cost is available on application.

GEMINI G805 FLOPPY DISK SYSTEM FOR NASCOM-1 & 2

It's here at last. A floppy disk system and CP/M.

CP/M SYSTEM. The disk unit comes fully assembled complete with one or two 5¼" drives (FD250 double sided, single density) giving 160K per drive, controller card, power supply, interconnects from Nascom-1 or 2 to the FDC card and a second interconnect from the FDC card to two drives, CP/M 1.4 on diskette plus manual, a BIOS EPROM and new N2MD PROM. All in a stylish enclosure.

Nascom-2 Single drive system.	£450 + VAT
Nascom-2 Double drive system	£640 + VAT
Nascom-1 Single drive system.	£460 + VAT
Nascom-1 Double drive system	£650 + VAT
Additional FD250 drives	£205 + VAT

D-DOS SYSTEM. The disk unit is also available without CP/M to enable existing Nas-Sys software to be used. Simple read, write routines are supplied in EPROM. The unit plugs straight into the Nascom PIO.

Single drive system	£395 + VAT

(please state which Nascom the unit is for)
Certain parts of the CP/M and D-DOS disk systems are available in kit form.
Details available on request.

KENILWORTH CASE FOR NASCOM-2

The Kenilworth case is a professional case designed specifically for the Nascom-2 and up to four additional 8" x 8" cards. It has tulipwood side panels and a plastic coated steel base and cover. A fully cut back panel will accept a fan, UHF and video connectors and up to 8 D-type connectors. The basic case accepts the N2 board, PSU and keyboard. Optional support kits are available for 2 and 5 card expansion.

Kenilworth case	£49.50 + VAT
2-card support kit	£7.50 + VAT
5-card support kit	£19.50 + VAT

GEMINI EPROM BOARD

This Nasbus compatible EPROM board accepts up to 16, 2716 or 2708 EPROMs it has a separate socket for the MK36271 8K BASIC ROM for the benefit of Nascom-1 users. And for Nascom-2 users, a wait state for slower EPROMs. The board also supports the Nascom Page Mode Scheme.

EPROM Board (kit)	£55 + VAT
EPROM Board (built & tested)	£70 + VAT

CASSETTE ENHANCING UNIT

The Castle interface is a built and tested add-on unit which lifts the Nascom-2 into the class of the fully professional computer. It mutes spurious output from cassette recorder switching, adds motor control facilities, automatically switches output between cassette and printer, simplifies 2400 baud cassette operating and provides true RS232 handshake.

Castle Interface Unit	£17.50 + VAT

A-D CONVERTER

For really interesting and useful interactions with the 'outside world' the Milham analogue to digital converter is a must. This 8-bit converter is multiplexed between four channels – all software selectable. Sampling rate is 4KHz. Sensitivity is adjustable. Typical applications include temperature measurement, voice analysis, joystick tracking and voltage measurement. It is supplied built and tested with extensive software and easy connection to the Nascom PIO.

Milham A-D Converter (built and tested) ...	£49.50 + VAT

PROGRAMMER'S AID

For Nascom ROM BASIC running under Nas-Sys. Supplied in 2 x 2708 EPROMs. Features include: auto line numbering; intelligent renumbering; program appending; line deletion; hexadecimal conversion; recompression of reserved words; auto repeat; and printer handshake routines. When ordering please state whether this is to be used with Nas-Sys 1 or 3. **Price £28 + VAT.**

DUAL MONITOR BOARD

A piggy-back board that allows N1 users to switch rapidly between two separate operating systems. **Price (kit) £6.50 + VAT.**

NASCOM-2 Microcomputer Kit	£225 + VAT
NASCOM-1 Microcomputer Kit	£125 + VAT
Built & tested	£140 + VAT
IMP Printer. Built & tested	£325 + VAT

CENTRONICS 737 MICRO PRINTER

A high performance, low price, dot-matrix printer that runs at 80cps (proportional) and 50cps (monospaced). This new printer gives text processing quality print. And can print subscripts and superscripts. It has 3-way paper handling and parallel interface as standard. Serial interface is optional. **Price £425 + VAT.** Fanfold paper (2000 sheets) **£18 + VAT.**

GEMINI 'SUPERMUM'

12 x 8 piggy-back board for Nascom-1 offering five-slot motherboard, quality 5A power supply and reliable buffering with reset jump facility. **Kit Price £85 + VAT.**

BITS & PC's PCG

5 x 4 board which plugs straight into Nascom-2. Operates on cell structure of 128 dots, producing 64 different cells. Once defined, each cell may be placed anywhere, any number of times on screen simultaneously. Max screen capacity 768 cells. Dot resolution 384 x 256 = 98304. Many other features including intermixing of alpha-numeric characters and pixels. **Price (kit) £60 + VAT.**

GEMINI 64K RAM BOARD

Newly developed Nasbus compatible board that can accommodate up to 64K of RAM including Page Mode facility. **Kit Prices: £110 (16K), £130 (32K), £150 (48K), £170 (64K). Add VAT to all prices.**

All prices are correct at time of going to press and are effective 1st June 1981

DISKPEN

The powerful text editor written for the Nascom is now available on a 5¼ inch floppy disk with a number of new features. **Price £43.25 + VAT.**

PORT PROBE

Allows monitoring of input and output of Nascom PIO. This board can generate interrupts and simulate handshake control. **Price (kit) £17.50 + VAT.**

HEX & CONTROL KEYPADS

Hexadecimal scratchpad keyboard kit for N1/2. **Price £34 + VAT.**
As above but including (on the same board) a control keypad kit to add N2 control keys to N1. **Price £46.50 + VAT.**

BASIC PROGRAMMER'S AID

Supplied on tape for N1/2 running Nas-Sys and Nascom ROM BASIC. Features include auto line number, full cross-reference listing, delete lines, find, compacting command, plus a comprehensive line re-numbering facility. **Price £13 + VAT.**

GEMINI EPROM-PROG.

2708 (multi-rail) and 2716 (single-rail) EPROM programmer kit controlled by N1/2 PIO. Supplied with comprehensive software for use with Nas-Sys.
Kit Price £29.50 + VAT.

WE STOCK Nasbus COMPATIBLE COMPUTER PRODUCTS

All the products are available while stocks last from the Nascom dealers below.
(Mail order enquiries should telephone for delivery dates and post and packing costs.) Access & Barclaycard welcome.

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

BUSINESS & LEISURE MICROCOMPUTERS
16 The Square, Kenilworth, Warks.
Tel: (0926) 512127.

ELECTROVALUE LTD.
680 Burnage Lane, Burnage, Manchester M19 1NA.
Tel: (061) 432 4945.
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).