INMC NEWS 80

Issue: 3

February-April 1981

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

CONTENTS

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 your NASCOM distributor for this.

Chairman's bit

Well here we are again, and you may notice, with a nice number of hardware articles, and diagrams. My thanks to those who answered my appeal for typists, even to the unsuspecting wife of one member, who may not know yet that her services have been volunteered. I have had a number of people contact me regarding the technical drawing bit, and the drawings in this issue have been done by Brian Hollins of Slough who committed the tactical mistake of wandering into the shop and volunteering, not knowing that I had all the articles which required illustration in my case at the time. Thanks Brian, I'll bet you didn't expect such prompt acceptance of your offer!! Still, my thanks to all of you, we now have a nice little pool of typists and artists we can call on, and we'll be in touch when we need you. This way the load can be spread and the services offered can be distributed without any one person being landed with all the work.

Hopefully, this issue will be produced in record time, we've been indulging in a bit of "Digitus Extractus", and as there was a lot of material in hand, much of it already submitted on tape or disk, we've had an easy time of it. (Speak for yourself! - ED.) BUT !!! We've got little material for the next issue. So if you've got something you'd like to see in print, and think that it may be of value to others, then let us have it. Only please read the little note buried somewhere in this newsletter about articles on, `How I built my Nascom', and the like.

The clouds are still over Nascom (at the time of writing), and if anything look blacker as the days pass. Stock is still flowing from Nascom, but there are shortages of most product lines, with little apparent likelyhood of the supply situation easing in the near future. The receiver says that he has a number of possible purchasers looking into the company, but apart from purchasing the name of the company there can be little incentive on the part of any purchaser to carry on with expansion of the existing product lines as so much valuable development time and effort has been lost. It would take months to recruit engineering staff, and longer to weld them into a team capable of producing any worthwhile products. I doubt that any purchaser would be willing to put up the necessary capital for that unless the name and goodwill of Nascom were purchased at far below the receivers' current asking price.

The ball seems to be firmly in the court of the independent suppliers who continue to have faith in the product (if not necessarily in the people who currently control the company). New Nascom compatible products are planned and some are already available, to the extent, that if you were thinking of buying a Nascom (if you don't own one already, then how come you are reading this newsletter?), then I suggest you get one quick whilst there are still some around.

The six largest Nascom dealers have publicly announced their intention to support existing Nascom products and introduce new Nascom compatible products where ever possible. It doesn't require much crystal ball gazing to see a situation developing, where Nascom will be producing so little that all the effective support will come from independent suppliers. So that if you already own a Nascom you'll be sitting pretty. If you don't own one, but want one, you'll find it very hard to get.

In the meantime, the receiver is making life a little difficult for these very same dealers by having tried to raise a high court injunction against one of them (and sending strong warning letters to the rest), in part, regarding the use of the prefix `NAS' being attached to both existing Nascom products, and to products compatible with Nascoms. A somewhat unproductive action I would have thought, and calculated to alienate the goodwill there is between Nascom and their dealers. I can see Nascoms' point, they want to protect their good name against any old rubbish that anyone wants to market. But their action seems extremely heavy handed. Particularly as it is those very dealers who are keeping the name of Nascom alive in the light of the receivers' singular lack of action in that direction. I know of at least two dealers who have taken exception to this behaviour.

Last issue I mentioned the possiblity of making this newsletter more 'Nasbus' orientated, as more goodies are available for Nascom from outside suppliers than from Nascom themselves. The few letters we have had (you apathetic lot !!) all support this idea. However, in view of the above, perhaps 'Nasbus compatible' might be a better idea than just 'Nasbus', before the committee get high court injunctions slapped on them as well. How does the name 'Nasbus Compatible News' grab you ? Bit silly isn't it. Still we will continue our policy of supporting Nascom owners everywhere and being as impartial to all products that come our way for review, comment, etc.

This issue we've tried to make a bit more hardware concious, as there are a lot of owners out there who have soldering irons and are just itching to turn them on and tag lots of little goodies on to their Nascoms. The article about music synthesis is only a start. We've heard of one project to get a Nascom controlling a large electronic organ. So may be we'll end up with lots of musical Nascoms singing around the country. Perhaps, if the idea caught on, we could organise a concert for n-part harmony Nascoms or something.

D. R. Hunt.

Editorial

Too Expensive ?

In the editorial of the December issue of the LSG, old Brucie implies that all microcomputer manufacturers are busy making exhorbitant profits. While this MAY be true of the American and Japanese companies with their vast production capacities, and consequent low overhead per unit, I believe the situation at the low-cost end of the UK market to be somewhat different. I do, of course, have one particular UK company in mind when making the above statement, but let's just consider the others briefly.

The lowest cost microcomputer (if one can truly call it that) available in Britain has been developed, and is now being produced, with more than a helping hand from the tax-payer. Moving up the price scale a little we find one company that is having its product manufactured in the Far East, and an other that says it is "backed by Britain's largest private company." Then, momentarily passing by one certain product, comes an S100 orientated system that really does cost an awful lot more to turn into a sensible machine than the adverts at first imply. And continuing up the price scale somewhat further we venture into "black box and 'research machine'" territory where, undoubtedly, margins are healthy. I apologise if I've passed anybody by.

So, what about the company I passed by, Nascom ? This company started with no large backing, either public or private, and tried to offer value for money to the customer and reasonable margins to the dealer to gain their assistance in achieving a good position in the market. And this they did. But Nascom's own margin was small, too small, and having to offer dynamic RAM boards in place of static devices did nothing to help the situation. The net result of this low margin is now well known.

So Brucie, I have to disagree with your statement, and will even go so far as making a contradictory one. That is: should Nascom be purchased, and the Nascom 2 taken into production by the new owner, there is little doubt in my mind that this product will have to go UP in price.

P.A.Greenhalgh.

Letters

Keep the Humour

Firstly, in reply to Dr. G R Kelman, may I say that I appreciate things like Lawrence, Scurrilous Musings, spoof letters, and most things with humour in. Life is too boring to be taken seriously, especially all the time, keep them coming!

Secondly, will you please credit the person who wrote the Space Invaders program published in the last issue. As I am a beginner at machine code (and not much better at BASIC?) I have put off trying to write a Space Invaders game, having given up trying in BASIC because of the speed (or lack of it!). I would like to thank him, albeit in his absence, but I cannot do this if I do not know who he is! [Ed - Sorry, we don't know either!].

Come to that, I would like to thank everyone who contributes, and makes this THE most interesting computer publication around.

There you are, the unsolicited testamonial that you promised me 10.00 for!

Please, keep up the good work, especially things like the Kiddies Guide to Z80 Programming, etc.

In fact, just keep up the good work.

K. Brown, B.F.P.O.

Modifying STARTREK

Many thanks for the library Programs that I purchased recently. Space Invasion, 3D Noughts and Crosses, Magic Labyrinth and Startrek. All brilliant work but I have had trouble with the last mentioned two programs. Far be it for me to criticise (I am only a novice) but may I make the following comments.

Magic Labyrinth: There is a syntax error in Line 2440. The decimal points should both be upward arrows. (See elsewhere in this issue to see why that happened - Ed.)

Startrek: This is written for Nascom 2 with graphics but it does not say anything about that on the Program. This drove me nuts until that fact dawned upon me at 3 a.m. in the morning! Also I cannot get the machine code routine to work properly. I have a Nascom 1 with "Bits & P.C.s" graphics and the result was a mightmare of chaos. For owners of Nascom 1 with or without graphics, may I suggest the following alterations. Rewrite the machine code using Lines 90 to 130 inclusive with this:

```
DOKE  3152, 27085
DOKE  3154, 14336
DOKE  3156, -20735
DOKE  3158, -20665
DOKE  3160, 3370
DOKE  3162, -5664
DOKE  4100, 3152
```

Change Line 5340 to: IF I8<>31 GOTO 5360 (13 is not Nascom 1 C/R)
In Line 5760 and 5770 change CHR$(129) to CHR$(127)
In Line 5710 change all three CHR$( ) to suit your own particular graphics capability and imagination. The symbols will be used as Enterprise, Klingons and Starbases respectively.

Well that's it. Now it works. Anyway, it has been great fun sorting it all out and I have learned a great deal from so doing.

Ray Ridgwell, Gt. Torrington, Devon

NM Documentation OK !

As a new member may I first of all congratulate the Committee on an excellent publication which is both highly informative and yet very readable.

I would like to make a couple of comments. The first is with regard to the criticism, by some of your correspondents, of the NASCOM-2 documentation. Having had experience of the documentation supplied with other micros, I was absolutely delighted when I received the NASCOM manual. Obviously it contains the odd error, or is brief in places, but compared with other manufacturers' manuals I think it is to be commended not condemned. It cannot be expected to be "all things to all users".

My second comment concerns possible future articles. I consider myself a reasonably competent BASIC programmer and am slowly beginning to feel my way around using machine code. However, my knowedge at the more technical side of the NASCOM is very limited. Would it therefore be possible for someone to prepare a few `noddy' articles on , say, general electronics (how does a decoupling capacitor work, tips on using multimeters etc) and perhaps on certain aspects of the NASCOM design.

Finally, I read with a sinking heart the debate in your pages on running BASIC without WAIT. I have difficulty running BASIC with WAIT! My system appears to have an undocumented - `Trample-on and change string' function. In large programs strings are corrupted - particularly when involved in routines printing to the top line of the screen. Result - the program eventually fails or, occasionally, BASIC crashes completely. I do not think this is anything to do with the dreaded `Plague' but I do not have any idea where to start looking for a solution - has anyone else had this problem?

Ian Irving - Thatcham, Berkshire.

ED - see elsewhere in this issue re. your "trample-on and change strings" problem.

LOAD Nasbug tapes into Nas-Sys

Other members who, like me, have upgraded from NASBUG T2 to NAS-SYS may find the following program useful. It solves the problem of what to do with all those programs that were saved on cassette using NASBUG, but won't load under NAS-SYS.

It is an adaptation of the NASBUG cassette Load sub-routine, running under NAS-SYS. One difference between it and the original `L' Command is that it stores the loaded programs at the header address + 1000H in order to avoid overlapping monitor workspace and stack (this assumes some memory expansion, of course).

Additionally, the displacement of 1000H makes it easier to modify the absolute jumps, etc. that are necessary after re-location.

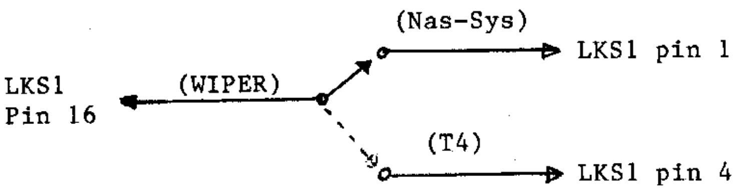
To use the load program, execute at 0C80, then load from cassette. The VDU format and error-checking are as under NASBUG.

```
0C80  21 8A 0B 22 29 0C CF FE
0C88  1D 28 FB FE 1F 28 03 F7
0C90  20 F4 11 8A 0B 06 08 1A
0C98  FE 2E 20 02 DF 5B DF 64
0CA0  2A 21 0C 7D 84 4F 3E 10
0CA8  84 67 E5 21 00 08 E5 E5
0CB0  DF 64 23 7E E1 77 23 81
0CB8  4F 10 F4 DF 64 23 7E B9
0CC0  E1 D1 20 07 01 08 00 ED
0CC8  B0 18 B5 DF 6A 18 B1 00
```

Paul de Bak, SWEDEN

Running T4 on an N2

If anyone wants to run T4 on an N2 then place T4 1 and 2 in the first two sockets of block A i.e. sockets and wire the associated linkblocks for 2708 eproms. Remove the link between LKS1 pins 1 and 16. Connect LKS1 pin 4 (Block A select) to pin 7 (XROM). Connect a 1 pole 2 way switch as follows:



This I have used to load my old programs under T4, then either use them (STARTREK STRIKES AGAIN!!), or readjust them. Put the processor in a halt and switch over to Nas-sys.

Has anyone noticed the NMI lead on pin 4 of the Keyboard port. It would seem possible to connect a switch between this pin and 0v to generate an NMI instead of reset (as long as workspace has not been corrupted). This may give any info. such as being locked in an "eternal loop".

David Roberts, Kendal, Cumbria.

Sorting out my N1

It may interest fellow members of INMC80 to hear of a problem I have been having with my Nascom 1 and the cure that I have effected. The problem at first sight was a form of memory plaque, DD.., FD..., and ED... instructions were corrupted and Basic crashed frequently. Naturally I tried all the cures and even tied the data and address bus lines to +5 volts through 10K resistors but nothing really worked well.

I then added the Interface Components excellent EPROM board and measured the +5 volts rail, + 4.65 volts. Well, 32K and quite a few EPROMs made me start wondering how I could afford an 8A power supply. Then I put the meter across the earth terminals, earth on the power supply and on the N1 board, and low and behold there was a drop of 320 mV. I added thick wires to the power supply earth, running to Nascom 1 earth (100 mV drop), commoned the four rails on the motherboard, (screen from coax cable tacked in place) (20mV),and ran a further earth lead (thick) from motherboard to 43 way connector (5 mv).

Operationally the N1 is now immune to interference, washing machines, fridges, vacuum cleaners and the like, even a Weller soldering gun firing on the same ring circuit dosen't affect it.

I hope you can see fit to publish this letter as I feel there must be others suffering from the same problem and also considering the same solution I was, selling up and buying transatlantic plastic.

Doug Taylor, London

Audio-Video, Adventure, and a Tape mod.

Having recently joined INMC80, and receiving INMC80-2 as my first newsletter, I was delighted to see how well my money has been spent.

One of the projects I have in hand is an audio to video display system. By this I mean a type of programmable sound-to-light consisting of several audio filters each controlling one bit of a port. The port is read and used as the basis for a display on the monitor. The filters are proving the main difficulty and the idea is still on paper only. Has anyone else tried anything similar?

You may have read the article on "Adventure" in the August edition of Practical Computing and mentioned by Dr. Dark in the last newsletter. One of the great features of this program is its portability of databases. While it is easy enough for anyone to implement the program in machine code, unless certain standards are conformed to, the whole object of the program is defeated. I have been trying to define these standards with Chris Southern (he wrote a letter to PC about the program) but I realise others must be brought in to contribute to this process otherwise our efforts are largely wasted. I invite everyone who is even vaguely interested to write to me so that a proper "net" can be established.

Lastly, a modification to the tape interface which lets you play music on the cassette without writing rubbish on the screen. A relay controlled by TP10 is one way of tackling this, another is to connect pin 10 of IC30b (ENABLE) to the bit controlling TP10 and the LED. IC30 must be removed and pin 10 bent out. One end of a short piece of insulated wire (approx. 1.5") should be soldered very carefully to pin 10. The other end of the wire should be inserted into pin 12 of the IC socket of IC24 (Q4). The operation is very simple. When Q4 of IC24 goes high it turns on the LED but now it also enables IC30b activating the receive data clock for the UART. When low, IC30b is disabled, the UART does not get the "receive clock", and anything on RDATA is ignored.

Luan Thompson, OXFORD

Remarkable Graphics

In issue 7 of INMC News you gave the Nascom Graphics characters. In all of these characters the bottom two lines of the 8 x 16 cell are blank! However, the characters still run into each other (apparently) from one cell to another. This is remarkable!

D. Smith, Stockton-on-Tees

Ed. - Not all that remarkable: Nascom 1 has 16 TV lines per character, Nascom 2 has only 14. The diagram was for Nascom 2.

Thanks

Thanks for everything you are doing. Don't forget us Nascom 1 types will you?

R.C.Ridgwell, Gt. Torrington

Regular fixes

INMC80-2 arrived today, just in time to stop the withdrawal symptoms from driving me to the funny farm. Please keep the fixes coming (even put up the subs. if necessary). Thank you for your excellent work.

F.A,Whitby, Nottingham

Ed - We do say that Nascoms are addictive, but this is ridiculous.

CRT Speed Error

For those readers who have not been able to get my CRT display speed control working (see last issue), please curse no more, there is a misprint or two. Corrections as follows: In BASIC line 20, the third data item should be 3187 NOT 31187. Also, under "To EXECUTE FROM NAS-SYS:" `TABLE Address' should be 077E NOT OC7E.

Otherwise all is in order and I expect most have solved the difficulties anyway (!)

No hard feelings,

Dave Lorde, Pontyclun

PS Issue 2 was excellent. Can this go on?

More Thanks

This is a letter of thanks for your efforts over the past year.

In my opinion, the INMC80 News is indispensible to a Nascom owner. As it seems to be almost impossible to get any technical information from Nascom, INMC80 is the only source left to us.

Thanks to my membership of INMC80 last year I now have a Nascom 2 which runs at 4MHz without wait states and will also run cassette in/out at 2400 baud with no errors! I also have made use of the INKEY, REPEAT and `PRINT USING' routines and have several very interesting M/C code program on tape.

All the above I would never of heard of without INMC80s: once again - many thanks and long may you prosper! How do you do it for the price?

One last point - I have just finished entering the Space Invaders prog. in INMC80-2 and whilst I find it very good indeed I do not like the use of the `A' and `X' Keys for Base Movement - also it runs too fast for me even at Novice level! Please, can you publish in the next mag. details of how to: (1) slow it down, and (2) change the left and right movement keys as, in spite of David Hunt's excellent articles on Z80 Assembler programming I doubt if I shall ever understand enough to attempt such alterations on my own

W. Squires (yet another Dodo), Cambridge.

Ed - I'm sorry, but we don't have a source listing of the Space Invaders, so cannot advise on any mods - anybody disassembled it yet and can answer?

CHASE Mods.

Please find enclosed a patch to one of your free progs.

```
SCORING PATCH FOR "CHASE"

ADDS SCORING OF GAMES FOR EACH
SIDE IN H.BIRKETTS GAME OF "CHASE"
(SEE INMC ISSUE 4.)

MD33,34,35 TO C3 C8 ØD

THEN ADD:
```

```
ØDC8 21 CC ØB AF BE 21 D6 ØB 3C
ØDDØ 2Ø Ø3 21 FØ ØB CD DE ØD D4
ØDD8 CD 3E ØØ C3 5B ØC 7E FE 96
ØDEØ 2Ø 2Ø Ø2 36 3Ø FE 39 2Ø EC
ØDE8 Ø7 36 3Ø 2B CD DE ØD C9 ØE
ØDFØ 34 2B C9 ØØ ØØ ØØ ØØ ØØ 25
```

G.M. Clarke, Edinburgh.

Keyboard Help

I am trying to understand the KBD routine in the T4 Monitor and can get so far and then come to a halt! I'll persevere and get there but it strikes me that a useful inclusion in the INMC80 issues would be a section on Monitor Software; a nuts and bolts break down of the routine. One section could be treated each issue or even spread over two issues. I should be pleased if this suggestion were to be adopted and would hope that others would benefit also.

Dave Pyke, Preston, Lancs.

NAS-SYS CAN CHECK SUMS!

Many thanks for the excellent newsletters I have received since becoming an 'INMC80' member. All the work put in by committee members is reflected in the interesting and helpful articles and tips. For those of you with NASCOM 2s who have been wondering how to print out check sums using the "T" command (re Mr. D. Tucker's program for Nasbug monitors in issue 5), try the following:

```
FE08  UOUT : CP   BS   ;note BS is 08 here (not 1D as for Nasbug)
37           SCF       ;mustn't set carry...
3F           CCF       ;...or Nas-Sys doesn't jump to CRT routine.
C0           RET  NZ   ;no BS found so print the character
AF           XOR  A    ;zero accumulator
C9           RET       ;CRT ignores NULLs.
```

Then activate the routine by storing its address for the jump vector $UOUT, which is at address 0C77:

```
0C77  C3 80 0C    $UOUT :  JP UOUT
```

Now, to "switch on" the facility of ignoring backspaces (as produced by the Tabulate command to delete check soms), type U. To revert to normal CRT output, type N.

If you are using Richard Beal's "keyboard Repeater", then type in the routine UOUT at address 0C80, and modify address 0CA7 onwards as shown below:

```
0CA7  21 80 0C    LD   HL,UOUT
0CAA  22 78 0C    LD   ($UOUT+1),HL
0CAD  DF 5B       SCAL ZMRET
```

Now the vector at $UOUT, together with the keyboard repeat data, is loaded for you when you type E0C90.

Yours Nastaligically,
Rob Sheratt, Taynton.

What's Wrong With It?

It was interesting to read the reviews of the Henry's Basic Toolkit in the INMC80 issue 2. Being the Author of the package it was nice to get some feedback, even if not totally complimentary!

The program arose out of some work I did on modifying several Pet Basic programs to run on a Nascom 2. It was while I was doing this that I felt the need for

(a) an intelligent renumber program, and
(b) a cross-reference listing.

Both were originally written as stand-alone programs. It then seemed appropriate to put both together, toss in a few other items, and link the lot into Basic somehow.

It was at this stage I decided to take a RAM based approach. By making the program self-relocating it solved the problems of where to locate the program, and where to locate its workspace. (Use C80 or D00 etc and somebody is bound to have a special routine sitting there!). Also RAM is cheap - now around 20.00 per 16K (EPROM around 64.00 per 16K) - and re-usable when not running Basic.

In response to Mr. James Weatherson-Roberts comments:-
As explained above my requirements were
1) Renumber and
2) Cross-reference.
The rest came later as add-ons. I could have continued adding features, (one suggestion I received was for an on-screen editor that would cope with Basic lines over 48 characters long), but I had to draw the line somewhere, otherwise the program could have ended up larger than the Basic interpreter itself!

The one comment in the comparative review that worried me was the one that the documentation was "lousy". In fact it was that comment that prompted this letter. I do not claim to be a great writer, but I would expect that most people would at least describe the documentation as "adequate". It describes clearly and concisely all the commands available and how to use them. I admit there is no blow-by-blow account of how each command executes, but I believe that to be beyond the scope of a manual. A well commented source listing would be a step towards doing this, but has been omitted in this case to keep the cost of the package down.

Perhaps INMC80 readers would like to write in and comment on this? I don't promise to reply (perhaps an INMC dogsbody might? - I hope that you are not talking about ME - ED.), but hopefully INMC80 could publish the result.

The questions are:-
(1) What are your views on current documentation of Nascom Hardware/Software (be as specific as you like)?
(2) What more would you like to see? Worked examples? Source listings?
(3) What premium would you be prepared to pay for this? (Remember source listings are long - The Basic Toolkit is 31 pages, Nas-Dis is 41 pages - this increases production costs and distribution costs).

David Parkinson, Ipswich

Letters to the Ed.

We are more than happy to publish any letter written to the INMC80 committee that is of general interest to the readers. If the letter contains a query then we will try and answer it within the pages of the newsletter. We regret, however, that we just cannot cope with personal replies to queries and so, if it is important to you, we suggest that you try Nascom (joke!) or your friendly local distributor.

The Committee.

MASTERMIND MODS AGAIN

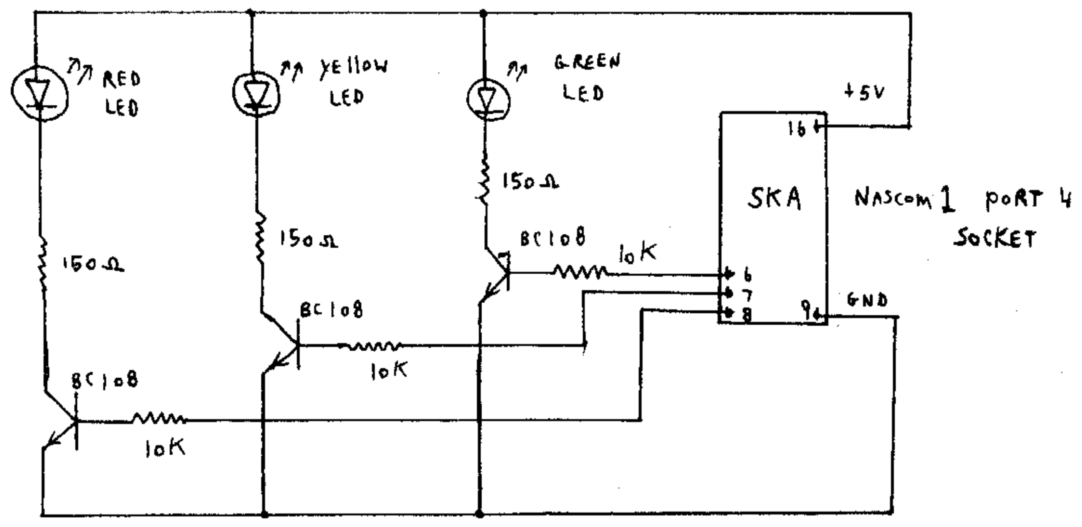
Dear INMC

Thanks to G. Benson for the suggested mods to my mastermind program in INMC80 1. The program was written before B-BUG was avalible, and the random number routine I used was rather poor, but was the best I had at the time. For those without B-BUG here's a modified version of its random number routine, which fits in place of the old RAND routine in Mastermind:

```
0C9A 21 69 0F   RAND: LD   HL,KEY
0C9D 01 07 04         LD   BC,0407 ;4 octal digits
0CA0 ED 5F      RN2:  LD   A,R
0CA2 86               ADD  A,(HL)
0CA3 38 01            JR   C,R2-$
0CA5 3D               DEC  A
0CA6 77         R2:   LD   (HL),A
0CA7 91         SUB2: SUB  C
0CA8 30 FD            JR   NC,SUB2-$
0CAA 81               ADD  A,C
0CAB 3C               INC  A
0CAC 77               LD   (HL),A
0CAD 23               INC  HL
0CAE 10 F0            DJNZ RN2-$   ; repeat for 4 digits.
0CAF C9               RET
0CB0 00               NOP  ; spare
```

TRAFFIC LIGHTS

Here is a simple hardware software combination suitable for an LED traffic lights display, as requested in INMC80 1 by P. Nurse. First the hardware, three transistors driving LEDs from the PIO socket SKA, on bits 5 to 7 of port 4:



Now the software, a table driven loop with nothing specific to a traffic light application. Each byte of the table is used as follows:

```
BIT   7 6 5 4 3 2 1 0
USE   R Y G <-delay->
```

Where R Y and G are the outputs to the LED drivers (1=on), and the lower 5 bits hold the number of seconds (approx) that the output is maintained for. The special code 00 denotes the end of list and causes the table pointer to be reset to the start. The program is relocatable and will run on NAS-SYS or T1 to T4 monitors, execute at E00:

```
0E00 3E 0F     GO:    LD   A,0FH  ;output mode
0E02 D3 06            OUT  (06),A ;for port 4
0E04 21 21 0E RESET:LD   HL,TAB ;table
0E07 7E        LOOP:  LD   A,(HL) ;get entry
0E08 B7               OR   A      ;test
0E09 28 F9            JR   Z,RESET-$
0E0B E6 E0            AND  0E0H   ;get top 3
0E0D D3 04            OUT  (04),A
0E0F 7E               LD   A,(HL)
0E10 23               INC  HL     ;onto next
0E11 E6 1F            AND  1FH    ;get lower 5
0E13 28 F2            JR   Z,LOOP-$
0E15 4F               LD   C,A    ;delay count
0E16 06 80     DL2:   LD   B,80H
0E18 AF        DL1:   XOR  A      ;KDEL max
0E19 FF               RST  38H    ;monitor delay
0E1A 10 FC            DJNZ DL1-$
0E1C 0D               DEC  C
0E1D 20 F7            JR   NZ,DL2-$
0E1F 18 E6            JR   LOOP-$ ; next entry
                      ;
0E21 90        TAB:   DEFB 90H ;red 16 secs
0E22 C2               DEFB C2H ;red,amber 2 secs
0E23 2A               DEFB 2AH ;green 10 secs
0E24 42               DEFB 42H ;amber 2 secs
0E25 00               DEFB 00  ;end of cycle
```

KEYBOARD BITS

I was surprised to see in INMC80 1 that Martin Dyer was using the PIO and push button switches for the controls on a space invaders game. The NASCOM keyboard is suitable for this job, providing the correct software is used to read it. The KBD routine does give information about keys that are held pressed, as anyone with a repeat keyboard will confirm. Whenever the routine is called it updates the bit map, KMAP from C01 to C09, where each bit in the map represents a key on the keyboard (1=pressed). Here's a short program to tell you the byte and bit in KMAP for any one key as it is pressed. The program should run on NAS-SYS or any of T1 to T4 monitors, and is relocatable, execute at F00:

```
>T F00 F4A
    0F00 11 D0 0B 21 3D 0F 01 0C
    0F08 00 ED B0 3A 69 00 FE C5
    0F10 28 03 DF 61 37 D4 69 00
    0F18 11 D5 0B 0E 08 21 09 0C
    0F20 06 08 7E 17 38 0B 10 FB
    0F28 2B 0D 20 F4 3E 20 12 18
    0F30 DA 3E 2F 80 12 1E DC 3E
    0F38 31 81 12 18 CE 20 42 69
    0F40 74 20 20 20 6F 66 20 43
    0F48 30 20 00 00 00 00 00 00
```

Meanwhile back at the space invaders game, I wrote my version about a year ago after I got my programmable graphics board working. As I had spent a lot of time getting arcade quality graphics on the game, I wanted to make sure it responded correctly to the keyboard move and fire buttons (hold to move, fire while moving etc). I have included below a simplified version of the main loop of the program at the point where the keyboard is checked. Controls are Z=left, X=right, NEWLINE=fire:

```
MOVE: CALL KBD      ; monitor keyboard routine
      JR   NC,BITS-$
      CP   1FH      ; newline on NASCOM 1
      CALL Z,FIRE   ; send off missile
BITS: LD   HL,0C03H; KMAP
      LD   A,(HL)
      DEC  HL
      RLCA          ; bit 4 (Z) to bit 5
      OR   (HL)     ; bit 4 (X)
      CP   10H
      CALL Z,RIGHT  ; move gun right
      CP   20H
      CALL Z,LEFT   ; move gun left
      DEC  (IY+1)   ; bomb delay
      CALL Z,MOVBOM; move bombs
      DEC  (IY+2)   ; missile delay
      CALL Z,MOVMSL; move missile
      etc
```

I hope this bit of code will be of use to anyone working on shooting games, like invaders.

Yours faithfully,

D. Ritchie.
Montrose.

D.J.Software

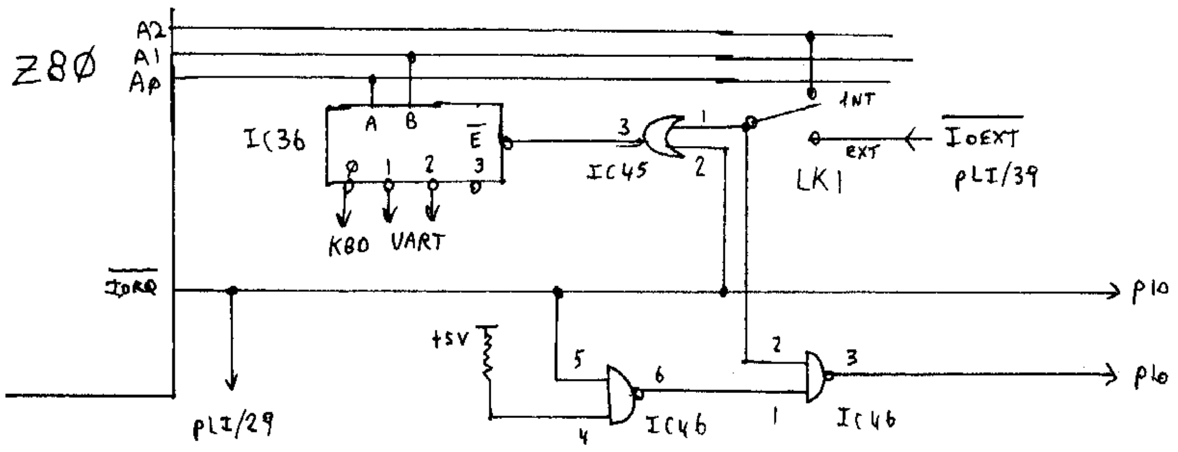
Following Mr J.B.Hawkes letter in INMC80-2, it has been brought to our attention that D.J.Software are no longer trading, and consequently the ZEAP patch tape is no longer available.

ODD IO

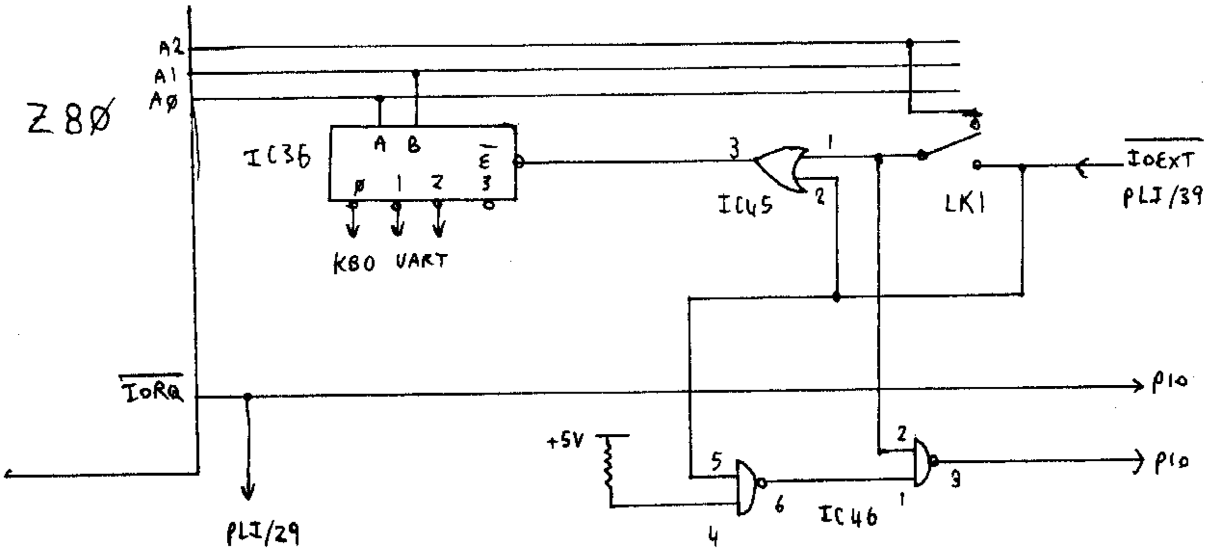
Dear INMC

I recently completed a new IO extension board for my NASCOM 1, this included a CUTS cassette interface, a sound effects generator chip and some TTL ports for a joystick, keyboard and an IBM printer (my PIO is used to provide real time interupts from the video blanking logic). On testing the board by reading and writing to the new ports I got conflicts with more than one device attempting to put data on the bus at the same time. Naturally I suspected that I had either designed or wired up the port decoding wrongly, but I could find no errors on my board.

I decided to look at the NASCOM 1 circuit again to check that I was providing the correct IOEXT signal, and here I discovered the cause of the problem, below is a simplified version of the NASCOM 1 IO decoding:

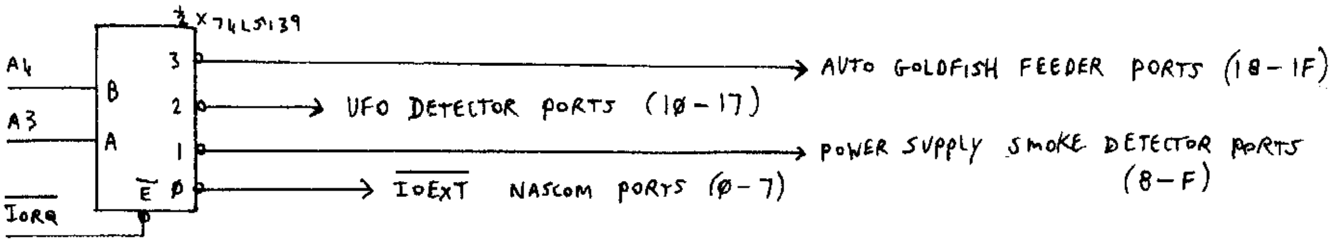


As can be seen on internal port decoding, if A2 is lo then the keyboard and UART ports are selected while A2 hi selects the PIO. Unfortunatly on external IO decoding IOEXT has the same effect as A2 had, so that there is no way of turning off the NASCOM 1 ports. Fortunatly the circuit can be corrected with the existing gates, to the following:



The link LK1 must be left in the internal position (to select internal decoding IOEXT must now be connected to IORQ). Now with IOEXT lo A2 selects between the internal ports, while IOEXT hi disables the on board ports as it should.

To make this modification disconect pin 2 of IC45 and pin 5 of IC46 by cutting the tracks on the component side, then join these two points together and to the IOEXT line at LK1, on the solder side. You can then build your IO board with the following decoding:

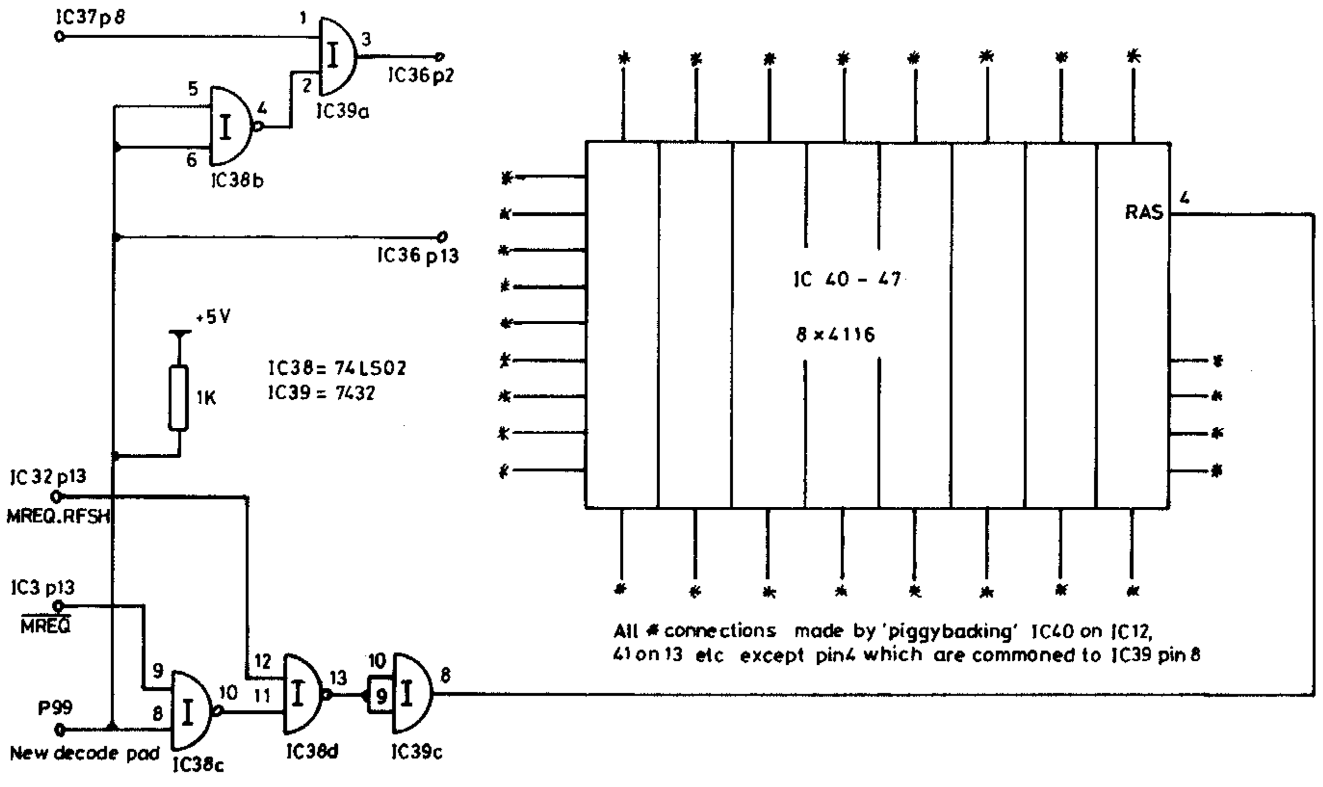

½ x 74LS139
A4
A3
B
A
$\overline{E}$
$\overline{IORQ}$
3
2
1
ø
AUTO GOLDFISH FEEDER PORTS (18-1F)
UFO DETECTOR PORTS (1ø-17)
POWER SUPPLY SMOKE DETECTOR PORTS (8-F)
$\overline{IOEXT}$ NASCOM PORTS (ø-7)

Yours faithfully,

D. Ritchie.

Circuit Diagram for modifying RAM A
===================================

See `32K - 48K Expansion' article.



32K→48K Expansion

We've received the following from John Fisher of Ipswich who very kindly sent us a tape which we took to be a Naspen tape. It wasn't! We presume the tape was made on a text handler of his own, and the tape format was ASCII strings, more like a Basic LIST. Still about 5 minutes of thinking had tape loader software prepared, and the following into a Naspen file for editing.

HOMEBREW HARDWARE

by John Fisher.

This is the first of a (possible) series of notes for enhancing the performance of the NASCOM-2. The author's interests are to do with pictures and signal processing and to this end I have tried to increase the speed and memory capacity of the processor, to increase the speed of the cassette interface and to develop a cheap and flexible way of drawing pictures. These requirements happen to co-incide with those of 'dynamic' games.

First of all the memory.

This expands the memory of the '32k' RAM (A) board from 32k (which it has already got sockets for) up to 48k. This is accomplished by connecting a further eight dynamic RAM memory chips onto the multiplexed address bus internal to the board. This means that only 2 additional ICs apart from the 4116s are needed and very little extra wiring need be done, such that it can be done in an afternoon.

Here is a step-by-step list of the modifications needed.

1) Connect the 8 extra 4116s piggy-backed on top of ICs 12-19, (see diagram) with all pins paralleled except the pin 4s.(Row Address Strobe). I used sockets soldered to the backs of the existing 4116s with the pin 4s bent up.

2) Connect all 8 of the new pin 4s together. This is the new RAS3 point.

3) Wire up a new RAS3 driving circuit as follows:

Create a new address select pad P99. connect this to +5v via a 1k resistor ( )

Mount 2 extra ICs (38 & 39). These are 74LS02 and 7432 respectively. This can be done by a small strip of Veroboard and some doublesided Sellotape. I put the chips on top of positions IC 29 & IC 30. ( )

CUT the following tracks on the p.c.

IC	Pin	to	IC	Pin	
36	14	+5v	36	13	( )
37	8		36	2	( )

CONNECT the following:

IC	Pin	to	IC	Pin	( )
-	P99		36	13	( )
37	8		39	1	( )
39	3		36	2	( )
-	P99		38	5	( )

-	P99		38	6	( )
38	4		39	2	( )
-	P99		38	8	( )
3	13	MREQ	38	9	( )
38	10		38	11	( )
32	13	MREQ. RFSH	38	12	( )
38	13		39	9	( )
38	13		39	10	( )
39	8		RAS3		( )

4) Connect the new address select pad P99 to the appropriate decode pin(s). ( )

5) Perform the clock modification as described in INMC newsletter No 7. (This delays the column address strobe to the 4116s by delaying the clock waveform from NASBUS Pin 5 to IC 31 Pin 3 through 2 inverters).

6) Perform the 'gridding' operation described in the NASCOM construction article.

7) These may be necessary but I have managed to do without them:
Increase the power supply decoupling in the areas of ICs 40-47 and at the address multiplexers ICs 20 & 21.
Increase the values of the series resistors R7-14 to allow for the extra capacitance of the added RAM chips.

The resulting memory should be fast enough to run with the processor at 4MHz without a 'WAIT' state. It was in my case. Unfortunately the author (and the INMC) does not hold himself responsible for any damage caused by inaccuracies in these notes, and the constructor should check that what he is attempting is safe.

==ooOoo==

Following on from the above, we can't recommend soldering sockets to the tops of very expensive MOS chips. So only attempt this mod. if you are prepared to accept the (remote) possibility of a handful of dead 4116s.

Having warned you off, now on to how to do it with the least risk:
There are two dangers, first static, and the usual precautions should be observed. Secondly, soldering so close to the package allows heat to travel through the lead frame to the gold wire bonding that connects the lead frame to the chip itself, and damage to the bond can follow.

So: cover a piece of polystyrene tile with cooking foil, and plug the IC into it. The foil connects all pins together reducing risks from static, and also conducts the heat away quickly. Plug the socket through another piece of foil, and fold it up (like wrapping a toffee). This keeps heat away from the socket. Solder two diagonally opposed pins to line everthing up, and if in line, solder the remaining pins. Then carefully remove all traces of the cooking foil between the chips with tweezers.

This method has been used very successfully for piggy-backing 2708s for use as selectable NASBUG/NAS-SYS monitors on Nascom 1s, and Tandy suggest it as a method of mounting the lower case character generator chip on the TRS 80.

Opinion

WHO'S AFRAID OF THE MICRO CHIP ? It's only a paper wolf, after all.

by RORY JOHNSTON. (Public Affairs Editor, 'Computer Weekly'.)

(This article first appeared in the 'OPINION' column of the Sunday Times on June 22, 1980, and is reproduced here by kind permission of the author, Mr Johnston, and the Sunday Times.)

Politicans, journalists and through them, the public at large, have become aware quite suddenly that something important has been going on in micro electronics laboratories. Engineering has never been a subject to grip the public's imagination, but in this case our opinion leaders have been struck by the feeling that these microchips could have economic and social effects far greater and more widespread than other technical developments in the past. But how much do people outside the laboratories understand what these chips actually are, and what their significance is? Has the public been properly informed about these important matters by the media ? I regret to say, no.

The invention of the integrated circuit has made the incorporating of elaborate electronic control into a wide range of industrial processes much more economically attractive than before. The ideas involved are not new; they are, however, very strange to most of the population. Technical developments up to now may have been awe-inspiring but at least everyone could understand what they were for. It is easy to see what a jet engine does but what on earth does a computer do ?

As long as there are only a few computers about it does not really matter if only experts understand them, but when silicon chips have invaded every corner of our lives it is vital that both the average politician and the average citizen have an adequate grasp of what is going on, so that they can pursue sensible policies and not be bamboozled by the technologists.

There are already signs of panic among some people, for instance in trade unions and civil liberties groups, leading to ill-considered action that could cause serious harm to our economy and the rule of law.

Clearly the media have an important duty to educate the public about microelectronics and they have taken up the task with plenty of enthusiasm. From the heavyweight television programmes to the magazines that are given away free at tube stations, hundreds of features have set out to explain the mysteries of the chip to an awestruck populace.

And how have they done the job ? Disastrously. The public has been left with a widespread impression that computers are capable of far more than they actually are, that unemployment for the majority of the population is inevitable, that computers when introduced assume an authority that mere humans are powerless to resist, and that these machines provide an easy means for the establishment of a political tyranny on a scale undreamt of by Orwell.

Certainly the task of stripping away the mysteries is not easy. For a start, how do you explain the technicalities without boring your audience ? What, for instance, is a 64K RAM, a product which many electronics firms are now racing to develop ? Random Access Memory is the normal sort of data store used by a computer, and 64 kilobits is a measure of how much information this particular device can hold. The Wirral Globe, however, explains that it is a chip with "up to 64,000 different uses." The Daily Mail states that "such a powerful memory would have been called a computer, only a few years ago." From the distant shores of Papua New Guinea, the Post-Courier tells us it can hold "64,000 kilograms."

Fact-hunting with a blunderbuss is rife. Vogue magazine states that "The National Enterprise Board has two non-government funded schemes, INMOS and EXMOS, for high technology." INMOS indeed exists but EXMOS is a figment of the imagination. Confusion has been rife between "silicon," the chemical element that forms the basis of microcircuits, and "silicone," an organic compound best known for its use in enlarging breasts.

More serious is the problem of understanding just what computers can do. Woman's Weekly told its readers: "You could soon find yourself sitting beside your home computer telling it to have the lawn mowed, windows cleaned, carpets vacuumed, freezer defrosted and dinner cooked." I hate to disappoint readers, but no, you could not. Physical tasks such as washing windows are far too complex for any general purpose mechanism that will be built in the short or medium term future.

We read in The Daily Telegraph: "This article was written on a machine that makes my trusty manual typewriter as out of date as a quill pen. When I made a mistake it corrected it ..." No, it did not. It made it easy for the writer to correct it, by pushing buttons rather than struggling with a rubber.

The spectre of technologically-produced unemployment is closely tied to this overstating of the computer's capabilites. Clement Freud MP said on "Any Questions?" "One of these chips equals, ooh I don't know how many hundred people." The technical staff's union ASTMS was more definite about it: "One chip can replace 800 white collars." In reality chips do not equal people, but in the office-automation field that ASTMS is dealing with, a machine incorporating a chip can increase a worker's productivity by at a very rough average 100 per cent - so for "hundreds" read "one."

It is when casting the computer in the role of Big Brother that the media indulge in the most bizarre flights of fancy. The Daily Mail tells us: "By 1984, it is not inconceivable that every computer in the country could be linked together in one gigantic data retrieval operation.... When one computer is allowed to talk to another, who knows how much damaging gossip can come out about every one of us ?" In fact it is totally inconceivable, being both technically impractical and pointless. And computers don't gossip.

Dozens of similarly mindless opinions have appeared over the last year. But more serious are the "1984" stories that are factually incorrect. For instance, in reports in February about telephone tapping, it was stated that a computer was being used to convert telephone conversations automatically into printed text. This is nonsense. The technology of "Voice recognition" is a very long way from being able to do that, and yet the assertion was repeated unquestioned by most of the national newspapers, showing alarming degrees of both gullibility and ignorance.

Sadly, the image of the tyrannical computer is a very engaging one and it is difficult for the truth to elbow out the emotion. Consequently, in addition to the public worry, ill-conceived schemes are put forward to control computers and the uses to which they are put. These controls are neither practicable nor necessary; technology changes but human rights and the principles that protect them stay the same.

Our major problem now is that it is the image of the computer and its high priests that has become tyrannical, not the machine itself. Beneath the fear there is a great deal of awe of a technology that is really not worthy of it. A social researcher has only to say that his data has been analysed by computer for his conclusions to be taken as irrefutable. Still, there is hope for the future in our schools, where many children are learning to use computers from the inside out. In the media as well, there are signs that quite a few journalists and politicians are beginning to find out what a 64K RAM is.

Doctor Dark's Diary

ONGOING HARDWARE SITUATION

Everyone will, I hope, be glad to hear that Marvin is now fully recovered from his recent illness. The problem was finally traced to two places; even after the 8 amp p.s.u. had been replaced (it was producing spikes at about 10 MHz) there was still the occaisional glitch. An oscilloscope revealed that there was noise on the mains, even when the scope itself was the only thing switched on in the house. If you find you have this problem, (and be very careful how you look for it, please, as I don't want to lose any readers!) there is a simple cure, that is very cheap. Wire into the plug that goes into the back of the 8 amp unit a "delta capacitor". The 3 amp unit never needed this, and I am not sure why the huge capacitor in the 8 amp unit doesn't seem to stop all the noise. Delta caps are available from Maplin, and if you haven't got their catalogue, I don't know how you manage. Perhaps you all live near component shops?

THE "ASKING FOR IT" SECTION.

There are three reasons for this section:
1. To help me to find more things to write about, thus preventing us all from becoming bored to death by unknown opcodes (however, see really interesting section, further on).
2. To make life easier for the committee by enabling those of you who want to write to me about this column, or any other vaguely Nascom related subject, to write direct. (Far be it from me to mention the fact that this would mean I got your letters a little sooner, on account of how busy the committee are!)
3. There really were three reasons, but I am not sure what the final one was. Perhaps I just want to get lots of post!

Anyway, my name and address are as follows:

Chris Blackmore,
31 Herne Rise,
Ilminster,
Somerset,
TA19 0HH

I don't promise to answer all, or even any letters, in case there is a great flood of them; I will try to, however. I've just remembered reason number three. The idea was to speed up the appearance of feed-back generated by these articles, which, in combination with the hoped for (I have this in writing from a member of the committee!) reduction in the gaps between issues of INMC80, should make it easier to understand whatever is that I am rambling on about..... Dare I mention that I have just thought of a fourth reason? Yes. Anyone near here want to start a sort of informal Nascom fan club?

VARIETY, THE SPICE OF LIFE, OR SOMETHING

I think it would be a good thing to let the users of the mystery system that is Nasbus compatible share our magazine. Mind you, heaven knows what the users of the Gemini (or is that the wrong name?) will make of this column!

NASPEN.

Once upon a time, yours truly used to write several draft versions of each of these episodes. Then they would be typed out, and sent to INMC80, who re-typed them before printing them. This was time consuming, as well as unkind to trees. (Some of my best friends are trees). This episode was prepared, edited, re-written and generally messed around using Naspen. When I finish, I will send it in on tape, which (this is where it may all go horribly wrong!) the editor will load, format, and print out with the fabulous Qume Sprint. He can even use his computer to edit in some of those REMarks of his! (e.g. What does this mean?-Ed)

OK, not the most detailed of reviews, I know. For better detail, see the review in Practical Computing. Or get some-one to demonstrate it to you and see what you think. "I'm convinced it's a major contribution to ....."

SOUND BOARDS THE SNEAKY WAY.

Seen all those adverts for sound boards for Nascoms, but can't afford them? Winchester Technology, who make and sell one that uses the AY-3-8910 chip, will sell you the documentation for a very reasonable amount. So what? says Disgusted of Tunbridge Wells. The circuit diagram is included in the documentation, which is of a very high standard in all ways. So you could buy a Winchester Technology prototyping board to make you feel better about pinching their circuit, and make your own up.

This is more or less what I am up to, except that I have made several changes to the circuit, to protect the guilty and make it even better, I hope. It will serve me right if it doesn't work: more news next episode.

OWNING UP TIME!

Some of the people who wrote to me about mystery opcodes will have had letters from me in which I suggested that ED 00 does something amazing. In fact, Pierre Molinaro has kindly written to explain how I made this error. When writing routines to test such opcodes, it is of course vital to ensure that only the opcode itself caused any changes you find after it has been executed. Since the monitor single steps the first instruction when told to 'Execute' something, the task will be affected even if the first instruction is a LD SP,nnnn intended to move the stack to a place where the memory is known to be clear. I hope that is clear! So start with a NOP when writing code to test things you think may affect the stack, and you won't end up with a red face.

On the other hand, I must assure you that I really did know about ED 6B and ED 7B. I was just seeing if you were all concentrating.....

How many of you noticed that Personal Computer world recently wrote "Dave Barrow is intrigued by the missing instruction codes CB 30 to CB 37..." (February 81, page 136). PCW wrote about those codes in their second issue, back in 1978 which is where I found out about them. Can it be that they have memory plague?

PASCAL vs PILOT CONTROVERSY SHOCK PROBE

Actually, no contest, because the two languages are for different purposes. A long time ago I wrote a sort of Pilot interpreter, but wasn't very satisfied with it. It has since been converted to run under Nas-Sys by one of the very few people who expressed an interest in it. As soon as he is happy with his work, no doubt it will be sent in for the library. Meanwhile, I have read P.J.Brown's book, "Writing Interactive Compilers and Interpreters", and seen the error(s) of my ways. The whole thing will have to be done all over again, starting from scratch, in my view. You shouldn't hold your breath.... If you can't wait, there are various Pilot interpreters already available from magazines, although these are mostly written in BASIC.

What this boils down to is that I have a fair idea how much work there is in implementing something as complex as Pascal, and hope my "Is Pascal Necessary?" heading last issue has not caused offence. The paragraph was actually an attack against the authors of articles that have been padded out to earn more money, and was written because I was jealous of the fact that they get paid for their articles. (You want money for this ?! - Ed.)

Clubs

Local Clubs and Groups

Below is a list of names and addresses of people who would like to contact others in their area with a view to self-help, starting a local club, and probably a good excuse for a drink or three! If any other people would like to start something along these lines, then just send us a note of your name, address and/or telephone number. (Preferably on the back of a fiver!!)

"Would any Nascomaniacs or other interested parties in the West Kent area contact one of us with a view to forming an informal local group."	Chris Wallwork Tunbridge Wells 37682 Ray Szatkowski Tonbridge 355960
"I am willing to start a local club in the Maidstone area."	C. Affleck-Stewart 17 Kennington Close Maidstone, Kent.
"I will be happy to give any help I can to Irish Nascom users - I know how much comfort it was to me to know that the various complaints I ran into baffled someone else as well as myself!"	Rory O'Farrell Tinode Blessington Co. Wicklow, Ireland. 01-582285
"A computer club has been formed around the Southend College of Technology. If there are any interested members in the Southend-on-Sea area, please contact me."	R.Knight 128 LT Wakering LT Wakering Southend-on-Sea. 0702-218456
"I would like to get in touch with any Nascom owners in my area."	Jean Michel Guiter Lycee Denis Poisson 45300 Pithiviers, France.
"At the Anglia Computer User Group we would like to start a Nascom group. Would any interested parties please contact us."	Anglia Computer Users Group 88 St. Benedicts Street Norwich, Norfolk.
"I feel that Nascom users in Scotland should get together to help each other out with our various problems, and with this purpose in mind I would suggest that any Nascom users in Scotland who would like to meet or form a club please contact me to discuss this matter further."	J. Brown 17 Albert Ave Glasgow G42 8RB.

SPELLING

Someone has written in complaining about our spelling (although also admitting the articles are 'generally'(?) accurate). I aks yuo?

Disks again

GEMINI DISK SYSTEM

In the last issue we devoted some space to disk systems, in particular the Gemini G805. At the time we said we didn't intend to review it in that issue, as the people supplying the hints and tips had too much of a vested interest in the system and therefore any review forthcoming from that source could hardly be objective. Having said that, a few days later, a letter arrived from the Merseyside Nascom Users Group, containing a review on the Gemini system using D-DOS (the simple 'floppy tape recorder') software. Well although their review arrived before the last issue went out, it was too late to include it, so we have had to hold it over until this issue. Before we start on the review, we'll give a little of the history of the Gemini system.

Designing a Disk System
or How To Go Loony in Six Easy Lessons by Dave Hunt (a suitable case for treatment)

About March last year three people decided that it was about time that Nascom pulled its finger out and got the long awaited disk system on the road, failing that, they would have to do something themselves. Well it seemed that although the Nascom disk system had been seen in the flesh, it was still no nearer completion, so Peter, Richard C (not Beal) and I started to look around to see what made disks work, and why Nascom were taking so long. Bolstered by a bit of spare loot from Naspen royalties (Oh! What a give away.), and the fortuitous publication of a design discussion about a simple disk drive system in one of the more obscure industrial magazines, Peter and Richard beavered away for about a month, and finally phoned me, saying "Come on over, we've got something that works", and it did!! There it was, a huge 10 amp power supply, running a hand knitted 4" square vero card, surrounded by a rats nest of wires attached to an N2 at one end, and a Shugart drive at the other. It was a rather different approach than the American designed disk controllers available, in that instead of giving the controller card a list of instructions and then letting the controller card take over the bus and get on with it, it made the Z80 do most of the work. This meant that the Z80 couldn't do something else whilst disk access was taking place (as the American designs envisaged), nor could system interrupts occur during disk access. But, what the heck, it brought the chip count down to 8, and made it cheap compared along side its imported rivals. 99.99% of applications would not require the Z80 to be doing anything else whilst disk access took place, and anyone giving priority to system interrupts would just have to put up with it (or buy an American one).

One problem caused considerable debate amongst us. We chose the Western Digital 1771 FDC chip to do all the work, and Western Digital recommend the use of an external data/sync separator. Now the trouble with data/sync separators is that they usually use monostables and these usually need 'twiddle pots'. 'Twiddle pots' aren't a clever idea when it comes to kits, because constuctors usually can't manage to set them correctly without sophisticated test gear. A lot of effort went in to designing a consistent data/sync separtor which didn't need adjustment, and after a lot of brain scratching Richard C came up with one. We wrote special test routines which could count (and then recover from) 'soft errors' and set it to work. In theory we should get on average about one error an hour, yet after churning back and forth all day, still no errors. Fantastic. Then someone muttered darkly that Tandy use a 1771 and get away without any data/sync separator at all. So we took it all off and repeated the test. 72 hours later, still no errors (except by picking up the drive and shaking it). Moral: who needs data/sync separators anyway!!! Just to hedge our bets, we left pads on the pcb to connect a data/sync separator, just in case.

So it worked, what next, we three could all take benefit from the disk system, but surely, we thought, some other people might like it as well. We offered it to Henry's Radio, for them to produce it as a kit on a royalty basis, so they could have all the agro, whilst we sat back and collected the money. Alright in theory, but it was still only one prototype and a rats nest at that. Peter had a prototype pcb laid out, and as with these things, whilst the rats nest worked perfectly, the pcb was noisy and unreliable. A lot of work went into getting that pcb reliable. Then I set to, to turn the whole thing into a production job, and write the words. By that time, a lot of other people had got to hear what we were up to. Here were people who might actually pay money (or offer services) to get their hands on a prototype, and we needed services, and prototype testers. In all, six production prototypes were made, and shunted off in different directions, whilst we gaily accepted offers of help in return. Dear old Richard Beal offered to write the CBIOS for a CP/M system, and ended up with an incredibly neat piece of software called SYS. Other offers were to investigate the feasibility of adapting the system to 8" drives, testing with various software houses' CP/M software, etc. I decided that a simple 'floppy tape recorder' might be a nice idea, and as I always need practice at software I sat down and wrote D-DOS (D for Dave, by the way. Modest aren't I). Peter decided that ED (part of CP/M) left a lot to be desired, so DISKPEN started to see the light of day.

So it was nearly altogether, Henry's advertised the kits, nearly all the components for the initial batch of 50 were purchased, and our bit was almost done. Henry's called the thing the Henelec Disk System, and already a comforting number of orders were in the bag. Then Gemini moved into the scene. "Why not," they suggested, "Put the whole thing in a box, built, with double sided drives, and sell it through other dealers as well. Disk systems are serious and expensive toys, and not all potential customers want (or can build) kits.". 'Sound thinking, Batman', we'll do just that. Gemini designed a power supply, to drive two disk drives and the controller card, and an attractive case, made to match the rather stylish Kenilworth Nascom case. The initial production order went up from 50 to 250, and the original launch date was put back two months to allow time for the software to be rewritten for double sided drives and five times the number of pcb cards to be made. The Gemini G805 was born.

It was decided that the Gemini would only be available built and tested, whilst the Henelec would only be availble as a kit. The software is compatible with both, and the Gemini has double sided drives with a total capacity of 160K each whilst the Henelec is available with both single and double sided drives with capacities of 80K or 160K, the single sided system remaining compatible and cheaper.

Gemini/Henelec Disk System with D-DOS

by the Merseyside Nascom Users Group

One of the group members has been fortunate enough to secure one of the first 5 Henelec Disk Systems (it wasn't, it was a built Gemini, Ed.), and in the short time he has had it, has managed to disassemble both the unit and the software. Over a few pints in a rather plush pub in the wilds of Cheshire we have written this short article for the benefit of anyone contemplating buying the system. The system that we are using consists of a single drive, the disk controller and a D-DOS operating software. We would have liked CP/M as well, but financial constraints prevented this at the time. Having only D-DOS means that this article is limited in that it can not discuss the use of CP/M on a Nascom, however, as CP/M is such an accepted standard, there are plenty of books and articles regarding this. Implementing CP/M on a Nascom 2 appears to be quite straight forward by replacing the MD PROM, but on a Nascom 1, it does require some board modifications so that the memory map can be altered to suit.

The Hardware

The system we have consisted of a single drive made by Pertec, fitted in a case (designed to take two such drives), of two part steel construction. The back plate carries the mains on/off switch, fuse, power regulator transistors, mains lead

and a 26 way ribbon cable terminated in a plug to fit a Nascom 2 PIO socket. The case and finish are of a very high standard. Along with the disk drive is a power supply which gives +5V at about 1 amp, and +12V at about 2 amps. These supplies are adequate for two drives and the controller card. We noted that the controller card also requires a -5V supply, but this is derived from a voltage doubler on the controller card fed from the clock. Again the power supply and controller card are constructed to a very high standard. The controller board uses the Western Digital 1771 Floppy Disk Controller (FDC) and is configured to give double sided single density operation on up to three drives if required. It is possible to get double density, however, a number of problems exist which make this unfeasible. The only reservation we have about the controller board is that it uses the internal data/sync separator of the 1771. Western Digital themselves recommend an external chip for this function. However, we have had no problems in this area, and the board has been made so that one could be fitted at a later date if necessary. We also received the complete circuit diagrams of the controller card (because we asked for it). We understand the circuits are not normally supplied. The Pertec 250 drive, as we have already said, is used in double sided, single density mode in this system, and is configured as follows:
35 tracks per side
18 sectors per track
128 bytes per sector
all to IBM standrds
161.28K maximum formatted capacity
Cable is supplied to connect two drives. We don't know what you are supposed to do if you wish to run three drives. Worth clarifying with the dealers if you have that number in mind.

The Software

D-DOS is normally supplied in two EPROMs assembled to run at B000H, but we understand is also available, to special order, in either 2716, or on tape assembled to run at 8800H if you already have a Bits & PCs Toolkit sitting at B000H. The first 1K (B000-B3FFH) consists of control software for the FDC. This includes a number of low level routines to control the PIO. It also initializes the FDC controller, drives, and positions the heads on track 0. Re-try software is also included which in the event of a read error makes five attempts at reading then moves the head out, repositions it, and has five more tries. If this fails, then it returns to the command level with an error message. No listing is supplied of this 1K but it is well worth disassembling because a number of routines exist which are not otherwise mentioned; for example, a power on boot which will load track 0 sector 1 into RAM at 1000H and then execute it. Assuming you have the ability to program EPROMs, alter the jump at B000H to B385H to make use of this feature. We hope our mentioning this doesn't upset the supplier too much. With this in mind, you will have to find the other routines yourselves.

The second EPROM (B400-B7FFH) consists of the main DOS and gives you four simple commands to control the disk, as follows:

N	Return to NAS-SYS
R	Read from disk
W	Write to disk
F	Format disk

The middle two need five parameters to be specified as follows:
R aaaa nn tt ss dd
Which translates thus

aaaa	Initial address to read data to
nn	Number of sectors to be read (128 bytes per sector)
tt	First track to read from
ss	First sector to read from
dd	Drive to read from

For example
R 1000 12 2 1 0
Read starting at 1000H, the next 12H sectors, starting at track 2H sector 1H from drive 0. The write command is similar:
W aaaa eeee+1 tt ss dd

aaaa	Initial address to write data from
eeee	Last address
tt	First track number
ss	First sector number
dd	Drive to write to

It becomes obvious from the above that a great deal of personal housekeeping is required of the operator, to keep a note of what programs sit at what track/sector on the disk. It is not by any means intended as a file handling system in its present form and does require a working knowledge of memory storage in general to operate. Thats what CP/M is about anyway.

Does it work?

Obviously this section will interest prospective purchasers of the system, and we have tried to be as honest as possible in our experience. The documentation is easily read and we did as instructed. We can see no problems in that area. The system was connected up and an attempt was made to format a disk. The drive went round but nothing else happened. A quick phone call, and the answer came back that the drive was running fast. Apparently the speed of the drives had not been set correctly at the Pertec factory and unfortunately when the controller tries to format a disk it also reads back the number of bytes on that disk (not quite what it does, but near enough for purposes of argument. Ed.). If the drive is running fast, then obviously the number of bytes written will be less than the required minimum so an error occurs. The solution was to take the metalwork apart, remove the pcb from the base of the drive - taking care not to strain the delicate head leads - and then set the drive to the correct speed using the strobe, provided for the purpose, printed bottom of the drive flywheel. This problem is known to the suppliers, and we are sure that drives will be set correctly in future. (Ed. We are told that they are now, so please do not fiddle !) The drive now worked well, but we were still unable to read the innermost track. It seems that the heads need a bit of bedding in (or may be the disks. Ed.), so we wrote a loop and left it formatting continuously whilst we went off for a few pints. On our return all worked well and no problems have been encountered since.

Overall impressions are that it is an excellent piece of equipment and gives much improvement over using tape as a storage media. For example, it will load 32K in 13 seconds. The only problem is the amount of housekeeping required to operate the system, but we dare say that in time this situation will change. The system is more than adequate for the job in hand and represents a long awaited add-on to implement on the Nascom.

Cost of the system and how does it compare

A single drive with D-DOS will cost you 395.00. A single drive with CP/M is 55.00 dearer. The CP/M with boot EPROMs and new MD PROM costs 95.00. Extra Pertec 250 drives are 205.00.
Doing a comparision with other systems
Two drives, case, power supply and CP/M

Gemini/Henelec	640.00
Nascom	655.00 (pre-receivership price - product still not available)
Apple	650.00 (single sided 80K per drive - not CP/M)
Sharp MZ80	675.00 + 99.00 for I/O box + 200.00 for CP/M conversion

It is difficult to draw a comparison on price alone, as these other systems have a lot of in-house software available as well, so we think the only conclusion that can be drawn is that disks for any system are not cheap at the moment.

If you have any further comments or have had experience with the Gemini or Henelec system, please let us know at the Merseyside Group. (We'd like to know as well. Ed)

If you want to contact the Merseyside Nascom Users Group please do so through the group secretary:

Graham W. Myers,
5, Beechwood Drive,
Wincham,
Northwich,
Cheshire.

Keyboard/Disk Problem

Whilst on the subject of disks, our thanks to one intrepid gentleman who has fitted his CP/M disk system to a Nascom 1. Now it is well known that some Nascom 1s will run quite happily at 4 MHz, and even if they don't then a bit of RAM swopping and changing the Z80 will usually bring them into line. Even so, there have been recurring reports of unreliability for no apparent reason, and this has usually been put down to 'a slow chip somewhere'. Well Mr. Hodgson has an N1, which until he fitted his disk system, would work perfectly at 2 MHz, and 'so so' at 4 MHz. Odd occasions when things became unreliable, but generally speaking, Ok. Well on went the disk system, and it worked perfectly at 2 MHz but not at all at 4 MHz.

Mr. Hodgson took up the challenge to see what was happening, his investigation led to a most intriguing solution. As Holmes is quoted as saying, "When you have eliminated the possible, then what ever is left no matter how impossible...." (or words to that effect). Mr. Hodgson was left with the keyboard !!!! The keyboard has a monostable which times the keyboard counter reset pulse. He found that the reset pulse was just a 'nth' too long and that it didn't come off until just after the first counter pulse arrived. Reducing the value of the timing resistor not only put the keyboard counter straight, but cured all the other unreliability problems as well. His CP/M burst into life at 4 MHz, the fridge which caused occasional 'crashes' no longer had any effect, spurious noise from other sources no longer corrupted programs, in fact, perfect. We can see how this problem could inhibit correct keyboard function, but as to the rest...well? Still in desparation this one is worth a try.

Multi-mapping a Nascom 2

by C. Bowden

When I recently added a Dual Disk System to my Nascom 2, I wanted to be able to switch between various operating options. On the one hand the CP/M system is superb in its file handling capabilities and there is extensive software support. On the other hand, the standard 'ASM' and 'DDT' utilities are 8080 code handlers, and 'ED' is rather clumsy to use when it comes to editing source code. 'EBASIC' is quite similar to other 8K Basics in its features, apart from some of the file handling commands, and again, it is necessary to construct source files using 'ED'. So until I obtain MACRO80, MBASIC and DISKPEN and other software more suited to the Z80 CPU and Nascom hardware, there is considerable attractiveness in being able to use ZEAP, NAS-SYS, ROM BASIC, NASDIS, NASPEN, as well as commercial software such as Chess, etc, that I have.

Most of the Nascom software will run in RAM (although you must have the tape version of ZEAP); there is a large number of options available to users on how to map their Nascom, depending on how much RAM or ROM is used and its addressing. The system described in this article is only one way of obtaining a high degree of flexibilty, some of the possible variations are mentioned in the text. However, the system described works and offers the following features.

1) Support of NAS-SYS, with RAM available up to DFFFH (Basic ROM selected), or RAM up to EFFFH (D-DOS selected).
2) Support of CP/M, with RAM available to EFFFH.
3) All NAS-SYS software (except ROM Basic) has been placed on disk.

Certain points should be made here:

1) Standard (ROM) D-DOS runs from B000H to B7FFH, which would conflict with RAM as mapped above.
2) ROM ZEAP will not (unless somehow modified), run in RAM, and is sited at D000H to DFFFH. There would again be some conflict with RAM.

In either case the ROM's would work, but would limit the effective system RAM size.

Possible solutions

ZEAP
1) Use a tape version of ZEAP (1000H to 2000H). This would allow up to 52K of RAM to remain free.
2) Find out how to modify ROM ZEAP so that a RAM copy of it will run. This allows 48K of free RAM.
3) Switch off the ROM `Block Select' signal out of the MD PROM or memory board decoder when ZEAP is not in use. (48K with ZEAP selected, 60K max. without ZEAP.)

D-DOS
1) Use the tape version of D-DOS. This is rather unattractive as it means a tape has to be loaded each one wishes to use disk.
2) Disable the D-DOS ROM's by switching the `Block Select' signal off when ROM's are not in use. This is a better solution, but it means in some cases that extra ROM sockets are needed, which are not available, or else that two out of four ROM sockets may not be used.
3) Change D-DOS to run at F800H - FFFFH. (Or try to persuade your Nascom dealer to supply D-DOS to run at this location.) I altered D-DOS without too much difficulty by altering all the jumps etc from BXXXH to FXXXH (where the XXX also changes in the second most significant digit).

Important Construction Notes

1) The Nascom 2 manual makes reference to Block A and Block B on the main board. These locations change depending on several variables, and in order to avoid confusion, where BLKA or BLKB are refered to in the following description, then BLKA = Pin 6 of LKS1, BLKB = Pin 4 of LKS1.
The relevant IC's to be inserted are located by the IC socket numbers and not by the references A1 - 4, B5 - 8 marked on the pcb.
2) The switches S1 and S2 should be mounted as near as possible to LKS1 to reduce the chance of noise pick-up, and to reduce delays.

Circuit operation and positioning of IC's

1) All standard Nascom 2 IC's in normal socket positions.
2) CP/M ROM's 1 and 2 in IC sockets IC39 and IC40.
3) D-DOS ROM's in either separate 4K block on external RAM (A) board or EPROM board, or (if suitably reprogrammed) in IC sockets IC41 and IC42.
4) If available, 4 x 4118 RAMs in IC sockets IC35, IC36, IC37 and IC38. If not, it will be necessary to use a 4K block of RAM on a separate RAM (A) board. If 4118s are not used, it may be possible to site the D-DOS ROM's for area B000H to B7FFH in this location, and to loop a `B' decode from an external RAM board, but this has not been tried. (B000H to BFFFH is not decoded by the new N2/MD/CPM PROM.)
5) If available, a 4K block of RAM may be wired to decode E000H to EFFFH. (When the Basic ROM is selected on S2, it will disable the RAM and work correctly.

Circuit operation

A) Standard NAS-SYS/ROM Basic.
This selection of S1, S2 is shown in the circuit.

- i) Decode 0000H monitor decode selects NAS-SYS.
- ii) Decode 0800H video and workspace decode selects video and workspace.
- iii) Decodes E000H, E800H, F000H and F800H are ORed together and decode ROM Basic.
- iv) When S1d is in the position shown the 'Power On Jump' select 0000H; ie, the four diodes effectively act as if LSW1 1-4 were closed (up).

B) S2 selected to 'DISK'.

- i) The Basic ROM is completely disabled.
- ii) Any block of RAM in the area E000H to EFFFH will now function.
- iii) Decodes F000H and F800H are ORed together to select any IC's in sockets IC39 - IC42 inclusive (CP/M ROM's and possibly D-DOS ROM's).
(If necessary S2 could be changed to a 3 pole type, and the third pole used to select D-DOS in another IC block, if D-DOS is still at B000H to B7FFH.
- iv) With Nascom selected on S1, D-DOS can be used to Read/Write/Format disks.

C) S1 selected to 'CP/M' (S2 at 'DISK').

- i) The monitor ROM select is disabled, NAS-SYS will not operate.
- ii) The 0000H and 0800H decodes are paralleled and select a 4K block of RAM; this RAM can consist of 4 x 4118s in IC sockets IC35, IC36, IC37 and IC38; or it can be 4K of dynamic RAM on an 8K RAM (A) board.
- iii) The video and workspace RAM IC's are selected to the F800H decode.
- iv) The F800H decode is removed from IC's 39 - 42, effectively disconnecting the D-DOS ROM's.
- v) The F000H decode is routed to IC's 39 - 42, so the CP/M BIOS and BOOT will be operative.
- vi) The 'Power on Reset Jump' is changed to F000H, due to S1d allowing the diodes to 'float'; ie, IC2 pins 3, 6, 10 and 13 go 'high', equivalent to LSW1 1 - 4 being down.

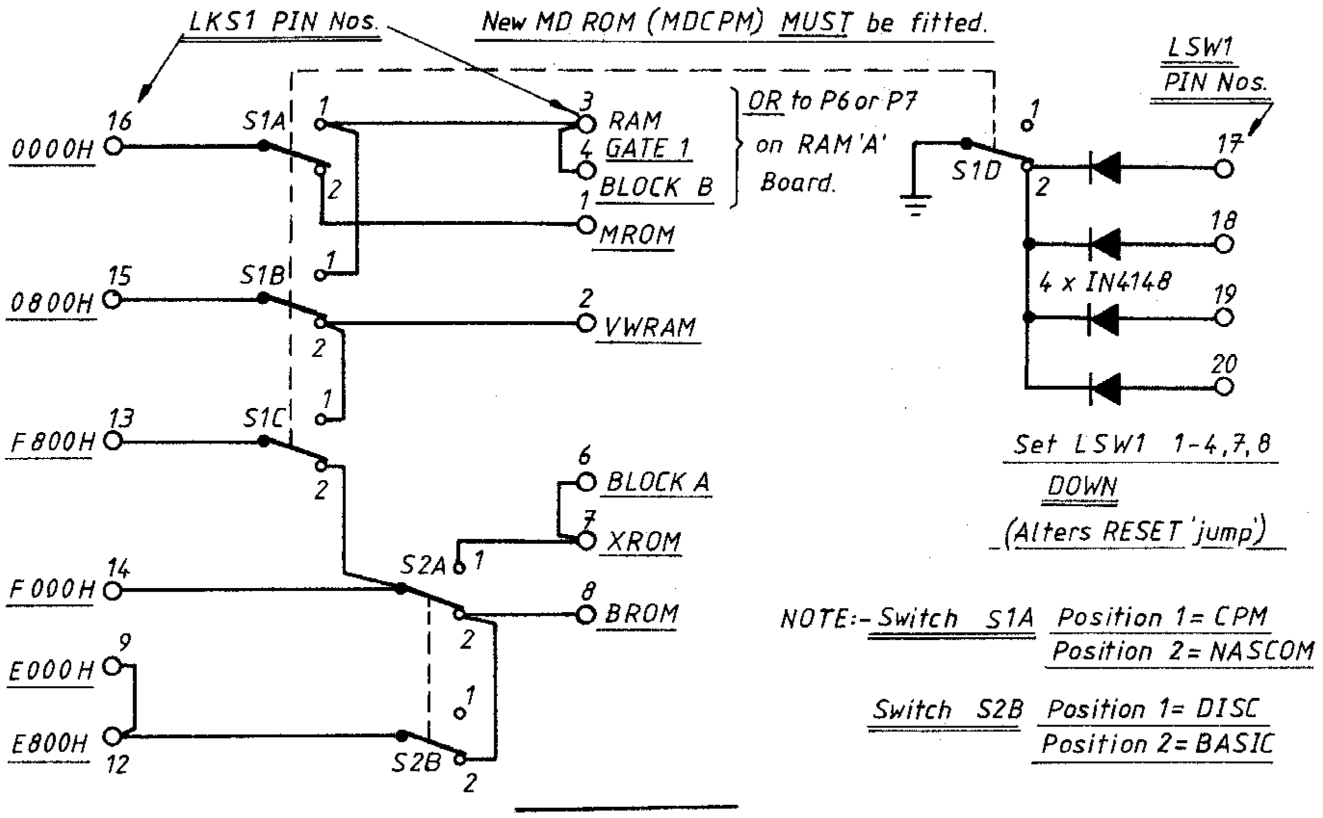
In the author's Nascom 2

IC35 - 38	=	4 x MK4118N-4 static RAM (switched as described).
IC39, 40	=	CP/M 2708 ROM's for F000H - F7FFH (switched as described).
IC42, 42	=	D-DOS 2708 ROM's (altered) F800H - FFFFH (switched as described).
Main RAM	=	48K RAM (B) 1000H - CFFFH (not switched)
Aux. RAM	=	8K RAM (A) D000H - EFFFH (not switched)

Total RAM under CP/M	=	60K (excluding video/workspace)
Total RAM under Nascom/Disk	=	56K (excluding video/workspace)
Total RAM under Nascom/Basic	=	52K (excluding video/workspace)

The system has been working in this fashion for a week or two, and has not exhibited any serious problems. The clock is set at 4MHz with 1 'Wait State'. The RAM (A) was modified when originally used on a Nascom 1 to cure slight 'memory plague', by gridding the power tracks, and putting 6K8 pullups on the outputs of one row of memories. The clock circuitry into the CPU has been modified slightly to improve clock symmetry. IC11 has been changed to a 7414 and a 270R pullup has been added between pins 6 and 14 of IC11. R85 (33R) has been short circuited. Without the last clock mod occasional 'soft' errors occured, particularly when using Basic.

Finally, a couple of points about the use of D-DOS not mentioned in the notes supplied. Firstly, D-DOS uses the area just below 1000H as stack space. If a 'READ' command is used to transfer data off disk into RAM at the stack address, the program will fail. The best solution here is to 'READ' to some other area, and then to use the 'C' or 'I' commands to move the data as required. Secondly, the 'Sector' counter used by D-DOS can only count up to FFH sectors, after which it overflows back to 00H. The ammount of data to fill FFH sectors occupies 32K of RAM (128 x 256 bytes). In fact, since sectors start at 01H rather than 00H, it is slightly less than 32K. This problem can be solved by putting data onto disk (or reading it off) in blocks not exceeding 30K in length. The whole usable RAM under NAS-SYS/Disk can be tranferred in 2 x 28K blocks, and the whole lot can be read or written in less that 30 seconds.



LKS1 PIN Nos.
New MD ROM (MDCPM) MUST be fitted.
LSW1 PIN Nos.
0000H
16
S1A
1
2
3 RAM
4 GATE 1
BLOCK B
1 MROM
OR to P6 or P7 on RAM 'A' Board.
S1D
1
2
17
18
19
20
4 x IN4148
0800H
15
S1B
2 VWRAM
F800H
13
S1C
6 BLOCK A
7 XROM
S2A
F000H
14
8 BROM
E000H
9
E800H
12
S2B
Set LSW1 1-4,7,8 DOWN (Alters RESET 'jump')
NOTE:- Switch S1A Position 1= CPM Position 2= NASCOM
Switch S2B Position 1= DISC Position 2= BASIC

IMPERSONAL

Scurrilous Musings

by Guy Klueless

Ego Departmart

It has come to the attention of your intrepid reporter that the Great British Post Office (sorry, shouldn't that be British Telecom?) indulges in a little bit of flag waving now and then. It seems that when they have custom made chips manufactured, they have a little picture of Buzby printed on the masks. Further, it's rumoured that they use Buzby's legs as a registartion target. I always thought there were more uses for that dratted bird than the adverts proclaimed.

Whilst on the subject of ego boosting, Nascom, under the guise of the Reciever have been flinging High Court injunctions about. One of the affidavits actually quotes part of this very column (from the last issue) as evidence (of what I can't figure out)... but still, it's flattering to mentioned in high places.

Another titbit from the court evidence seems that Nascom want to stop people using the prefix 'NAS' on various non-Nascom products. I wonder how they are going to stop the space shuttle carrying the legend 'NASA' on the sides ?

Unintentional puns

Seen in a TR*S* 80 Fortran Manual:
"If a Fortran program is to be run in ROM, the programmer must be aware of the ramifications." UGH !!!

Hardware review

by Richard Bateman

The I/O Systems Graphics Board.

The I/O Systems Graphics Board appears to be a very attractive product - giving very high resolution at a reasonable price (384 dots by 224 dots by 55.00). It uses bit mapping and uses 10.5K of your memory when all the screen is displayed. The board does not contain any video memory itself, but is connected to a 16K block of the RAM (A) or (B) board.

What do you get ?

An assembled PCB 4.25" x 3.75", double sided but not plated through, with 6 socketed ICs, 3 resistors and 6 100n capacitors. There is also a 33 way low quality edge connector and a 35 page very high quality manual. Unfortunately there are no circuit diagrams and no real explanation of how it works.

Connection

This is the hard part. You have to lift several ICs on your precious Nascom and RAM board and make connections - 35 in all - with not very good instructions. It is quite difficult to know where to mount the board as it does not conveniently fit anywhere. Overall the board is not the easiest thing to connect or mount, but with some imagination it can be done.

Does it work ?

Very well, an extremely stable video with good clean lines. The result is better than with the original video as no accesses are made to write to the screen during the display refresh time. In fact the Z80 is shut down during the display refresh, which can be nearly 75% of the time, so don't expect fast displays with this system as it is equivalent to running with a 1MHz clock (compared with an N2 at 4MHz).

Conclusions

The I/O Graphics Board is a clever implementation of a simple idea, possibly a little overpriced when the penalties are considered. The detractions are, considerable loss of system RAM, reduced operational speed of the system, and difficulty in performing a workmanlike job in connecting the card. I liked the manuals (with regard to programming the card) and the superb graphics that may be implemented. Overall, good, but really only applicable to dedicated graphics freaks.

The graphics card costs 55.00 and is available from:
I/O Systems Ltd.,
11, Laleham Ave,
Mill Hill,
London, NW7 3HL.

Classified Ads.

Philips mini-digital cassette recorder. Brand new, in original wrapping with interface diagram and information. Superceded by disk system before interface was purchased from Currah. Cost 109.00. Will accept 95.00. 01-449-1690

Siemans 40 track, single sided, double density 5.25" drive. Owner upgrading to double sided so 110.00. 01-223-9829

Z80 made simple

THE KIDDIES GUIDE TO Z80 ASSEMBLER PROGRAMMING

D. R. Hunt

Part: The third.

Catching the Nascom Disease
(Mental Health Warning: Computers are addictive)

So far we have learned to count with the aid of eight fingers two thumbs and six assorted toes (unless you are lucky enough to have grown the necessary extra three fingers on each hand), we have looked at what goes into making up machine code instructions by making use of a spare #1,000 train set (or perhaps you tried it in real life by hiring a redundant BR marshalling yard for a weekend). We have also looked at the unpredictable results of giving the processor the wrong codes to work with, and had a look around the innards of the Z80 in general. Now its `crunch time'. We're acutally going to write a program. Well not so much a program, more of a .... er .... well, stupid little thing that can only really count as a program because it actually does something, not much, but something.

This is the story of how I came to be involved with these mind bending machines, and how I wrote my first program. Bear in mind that at time I had only the sketchiest idea of what it was all about, and, although you might swear at the incomprehensibility of the documentation, it was a darned sight worse then than it is now.

First, capture your Nascom

It was, I think, in April in the year of '78 that I was sent forth by my superiors. My quest, to capture, dead or alive, a Nascom 1 from the shrine of the Lynx, situated in the dark and mysterious nether depths of Bucks, some leagues from my abode. I drove to Chesham and parked my trusty (or, in truth, I should say rusty) chariot in the parking place thoughtfully provided for the purpose by the elders of the Council of Chesham. I walked with trepidation up Broad Street, thinking that the name was ill devised, save for the purpose of confusing the finder by its very inappropriateness. Past the forboding gloom of the funeral parlour (I looked for black ravens or other evil birds, but there were none) and for the first time, approached the shrine of the Almighty. Finding a door marked with the sign of the Lynx (and sundry other unlikely names) I entered into a dark and smokey hall which seemed full in equal parts of cardboard boxes, plastic foam, semiconductor components and small gnomes scurrying about, all, so it seemed, answering to the name of Heather. I asked a passing gnome, "May I see G.., er, I mean Mr Marshall please.", and was told to hang on (which I did with difficulty as the cardboard boxes were stacked mighty high, and wobbled a lot). At last He approached. "Mr. Marshall," I said, grovelling on the floor, putting on my best ingraciating smirk, and touching the old forelock (which is now quite grey through the frustration which the Almighty bestowed upon me as I left). A giant (amongst the gnomes) looked on, somewhat bemused I remember. "Mr. Marshall," said I, my voice trembling before this awe inspiring creature, "I'm from XXXXX's and we've been advertising these Nascom 1 things for about three months, I've got a waiting list that has filled up my little note book, and we still haven't had one to play with. Can I please," grovel, grovel, "take one away so I can see what the heck it is I'm selling." The Almighty spoke, "My son, take thy leave of this place carrying thy doom away in this cardboard carton, and may the curse of computing be upon thee from this day forward." I left as quickly as I might, in great fear that the precious cardboard box would be stolen from me ere I reached the door. I had actually escaped with my life from the dread shrine with a treasured Nascom a full week before anyone else got one. And, low, there were not the expected bolts of lightning, nor thunder claps nor plagues of silicon chips persuing me up Chesham High Street. I hadn't even seen a fiery dragon (the bemused one still looked on, perhaps he was pretending to be the dragon that day, but he showed no tendency to burst into flames). I regained my trusty chariot in great haste imagining my fate, should I be mobbed by hordes of waiting Nascom Customers who had perhaps already heard that I had escaped with some of the treasure in the cardboard carton. I went home.

Enough of the flowery stuff (and I haven't touched a drop all night even though it's Christmas. Shows how long ago this was written.) I got out the instructions and didn't understand half the words in the book as it was written in 'foriegn' disguised as English. But I understood the circuits and layout diagrams, so I assembled it. A PSU was hastily designed on the back of a fag packet and the parts procured. Three days after my escape from Chesham I switched it on and it worked (no, no problems at all). Well it worked. Now what do you do with it?

I fumbled through the manuals and discovered that the 'T' and 'M' commands worked and that I could alter locations within the memory and then redisplay what I had done. I tried to understand the big program in the software book, as an example of what to do (it turned out to be a listing of the monitor, but I don't remember that I realised this at the time). But there was no clue here. Everything was utterly incomprehensible. Well this was a fine start. #200.00 worth of computer I couldn't use, when the punters who had paid their deposits saw it, I'd be a likely guest at a lynching with me as main participant. My govenors would be at the front of the crowd making sure the knots were tight, as it was I who had recommended that XXXXX's buy the darned things in the first place. I had to make it do something to prove to the customers, who would be collecting in a couple of weeks, that they hadn't spent a couple of hundred quid on a pile of electronic junk, and that they really didn't want their money back.

Don't try to be too clever, Dave (Very friendly guy, calls himself by his first name !- Ed.), little steps are always the easiest. First define what it is I intend to make it do.

Always know what it is you intend to do. I was told once that the definition of the 'Genius Level Programmer' is someone who writes a program, tests it to find out what it does, and then writes the specification. None of us are geniuses, so keep it simple, and always know what you are about.

So I have about a week to prove that it will do something, but what? How about making an asterisk blink on and off on the screen. That would prove that I had some control over it. So think!!!

First of all it's necessary to get some idea of the syntax of the machine instructions. Read and reread the Z80 technical manual. Machine instructions are written in mnemonics. That means it's supposed to make it easy to remember. This is true of some, like 'LD' stands for 'load'. But others are impossible. Who would know that 'DJNZ' meant 'decrement, jump not zero' unless I had just told you. I think it's done as a convenient shorthand and saves cramped wrists when hand assembling programs. The first thing about the instruction codes is that they are split into groups, 8 bit, 16 bit, load, logical, arithmetic, etc. Another thing is that when dealing with a 'from - to' type instruction, like a load, the destination is always first, the source is always last. So LD A,C means load (remember from the last episode this actually means copy) A with C. Or literally, fill A with a copy of C. That load instruction is known as a 'load register direct' instruction. Think about it, it's sensible. Another one is to 'load register immediate', that is don't copy the data from anywhere, just put it there. These Zilog people seem to use 'n' to represent the data, so LD A,n means load A (destination first again, notice) with 'n', whatever 'n' might be. An interesting group of load instructions is the 'load register indirect' (a less sensible definition I think), this is where one of the main 16 bit register pairs, HL, DE or BC is loaded with a 16 bit address, and then a register is loaded with the data 'pointed at' by the 16 bit register pair. LD A,(HL) is one of this group. The brackets round the HL indicate that it is indirect, and it is the contents of the location pointed to by HL which we are interested in. Yet another of these instruction types is the 'load ext addr' where the idea is to either load an address location with the contents of a register, or vice versa. So LD A,(nn), two 'n's indicating a 16 bit address, means fill A with a copy of the contents of address 'nn'. Note the brackets again indicate that it is the contents of the address we are interested in, and not the address itself. There are two other types, 'indexed' and 'implied'. I'll let you work those out for yourself.

Actually, all this `groups', `immediate', `indirect', etc, lark is not all that relevant. It's only when it comes to trying to describe the functions of the instructions to someone else that it really matters at all. For instance, I use these terms so rarely that I had to get the book out to look them up when writing the previous paragraph. The important part to learn is the fact that instruction mnemonics are always written destination first, and that brackets indicate that we are interested in the CONTENTS of the location `pointed at' by the address (immediate or indirect) supplied and not the address itself. No brackets means we are shunting data about inside the processor.

End of the theory bit, and back to the story. First I must locate the middle of the screen. Well there was a memory map that gave the screen area as being from 0800H to 0BFFH. Good, the middle must be 09FFH (and a half). Well I'll be happy if it's anywhere on the screen at this stage. Now how to get an asterisk there. Fumble through the Z80 technical book, and there is an instruction which is LD (nn),A. Now if I interpet the syntax of that correctly, it means, load nn with the contents of A. But (thinks) I don't know what A has in it in the first place. Fumble, fumble, ah-ah!! LD A,n. That must mean stick n in A. Well whats the code for an asterisk. Easy, look it up in the character set table. Good, I'm getting somewhere. If I write the following code, I should get an asterisk in the middle of the screen.

```
3E 2A        LD A,2A
32 09 FF     LD (09FF),A
```

So using the `M' command I enter 3E 2A 32 09 FF, starting at memory location 0D00H, (having already `sussed' that that was a RAM area I could play with). Exit from the `M' command and execute the program by typing E 0D00. Well, not quite what I expected. The screen filled with any old rubbish, and having pressed RESET, even the program had disappeared. So back to the drawing board. Oh dear, what a stupid mistake, I forgot to tell the thing to stop. So I type it in again and tag 76 (a HALT instruction) on the end. Re-execute it, - and - , nothing!!! Hitting RESET, and re-examining the program (at least it's still there this time), what's wrong now? Back to the technical manual; what's this about two byte operands having to be entered low byte first? (What ever that means.) Well I think a `two byte operand' might be the 09FF bit of the LD (nn),A instruction. Worth a try. Retype it, 3E 2A 32 FF 09 76. Re-execute it, Oh heck (or similar words) still no asterisk on the screen. What can I have done wrong now. Looking closely at the video map in the construction manual, I notice that the video addresses are not continuous. That is, they start at 080AH to 0839H and then the next line starts at 084AH. What happened to addresses 083AH to 0849H? A little mind boggling this, so I work out 16 lines times 48 characters across equals 768. Now even I know that there are 1,024 bytes in 1K of memory (and the book says it has a 1K video RAM) what happened to the other 256 characters? Well after more puzzling I decide that there must be a hidden margin in the screen, and if thats so, then my asterisk must have been put there. So I set to, using the video map this time, and recalulate where the middle of the screen is. I end up with 09E2H, re-enter the program and try it. Ye Gods!!!! There's an asterisk in the middle of the screen. It only took all evening to put it there.

Fantastic, I've written a program. It did what I wanted (at last). The most important lesson learned was that two byte operands are entered low byte first. Now to make it flash on and off. Simple methinks, put a space back at the same address I put the asterisk. The code for a space is 20H, so now my program looks like this.

```
3E 2A        LD A,2A
32 E2 09     LD (09E2),A
3E 20        LD A,20
32 E2 09     LD (09E2),A
76           HALT
```

Well having typed it in, I execute it. Nothing again!!! What the ..... Oh yes, it did it so fast I didn't see it happen. What I must do is replace the HALT instruction with a jump back to the start. Ok, so I look up a jump. Yipes, I've got a choice of `jumps absolute', `jumps relative', jumps on zero, jumps on no zero, carries, no carries, and goodness only knows what else. Well JP nn looks easiest. `nn' must be where to jump to, and as 'nn' is a two byte operand, I guess it has to be low byte first. So

```
C3 00 0D      JP 0D00
```

was added in place of the HALT. When executed, the screen filled with 'snow' (it was a Nascom 1 remember), but my asterisk seemed to be flashing on and off like the clappers. So fast in fact, that what I saw was the thing beating with the TV line scan. Second evening gone but flushed with success, I thought I had it licked. All I had to do was add a delay after loading an asterisk to screen and again after loading the space. The only problem was that there wasn't an instruction that said 'DELAY 1 SECOND'.

Now what? How do you make the thing hang about for a second before it goes onto the next stage. Well the idea of a delay subroutine occurred to me. The only problem was how to write one. Make it sit there and count to itself seemed a good idea, but how? Now I had read that the HL register pair could do 16 bit arithmetic. How about making it count to 65000 odd (from 0000 to FFFF), by using the INC HL instruction. That should slow it up a bit. Then another problem came to mind. I could see asterisks on the screen, but how was I to know whether the thing was counting. I'd read about the 'S' single step instruction, but didn't know what it did. Perhaps I could use this somehow to see what was going on. Ok, test it. I wrote

```
21 00 00      LD HL,0000
23            INC HL
```

using the 'M' command, and then typed S 0D00. Well it displayed something

```
1000 0D03 4800 0000 1000 0000
```

Yes, well, what did that all mean. Read the book. 1000, that must be the contents of the SP register (why 1000 I wonder), 0D03 that's the program counter, that figures, it's pointing to the next instruction. 4800 that must be the AF register pair, and look the HL register pair has 0000 in it. Now if I take one more step ....

```
1000 0D04 4800 0001 1000 0000
```

Cor, look at that, HL was incremented by one, just as I had instructed. The only thing now was to make it do that again and again until it had reached FFFFH.

It was about then that I decided that remembering the order of the 'S' command register display was a bit of a pain. So I stuck a bit of masking tape (left over from respraying the dent the Mrs put in the car) across the top of the monitor screen and wrote on it

```
  SP     PC     AF     HL     DE     BC
```

so that as the display scrolled up the screen, I could see what was happening.

Well, what happens when it gets to FFFFH? It overflows, and goes back to 0000H. Read the book. It says an arithmetic overflow sets the Carry flag. Good. But, by implication, does a non-overflow unset the Carry flag? The book doesn't say, so try it. If I load HL with FFFFH, then increment it twice, the first time should set the Carry, and the second time unset it. Away we go

```
0D00    21 FF FF        LD HL,FFFF
0D03    23              INC HL
0D04    23              INC HL

S 0D00
1000 0D03 4800 FFFF 1000 0000
```

So far, so good, lets increment it now, so I hit enter again

```
1000 0D04 4800 0000 1000 0000
```

Odd that!!! Something should have happened to the AF register, but it didn't change. Back to the drawing board. As an experiment I tried it with the A register and the same thing didn't happen. Or at least the F register changed, but when I set to and decoded the HEX number in the F register into its component bits (see part 1), and then compared it with the little map of bits in the F register in the book, the Z (zero) flag and the P/V (parity) flags had been set. (Now you may be wondering why I had to decode the F register bits, as NAS-SYS obligingly prints out the registers set at the end of the 'S' command display. Remember this was an original Mk I Nascom 1, with NASBUG T1, it didn't have such luxuries.) Anyway, it figures, The A register had gone to zero, but what happened to my 'carry'? Well as the Z flag had an effect, try it again when incrementing the HL register. Still zilch!!! Still not a twitch out of the F register. It seemed that these flag things had a mind of their own, and only worked when they wanted to. The truth was, of course, that I was wrong, INCrement instructions are not truly arithmetic and the affect on the flags is somewhat variable depending on the conditions. As a general rule, all flag changes except the Z flag (associated with 8 bit INCs only) should be ignored.

Back to the story. Boring isn't it, or perhaps your ears are burning. Having decided that the flags have a will of their own, I had to discover a better way of testing whether the HL register had reached zero. I'll put you out of your misery, and at the same time save me a modicum of embarrassment, by telling you how it may be done rather than the rather ham fisted way I actually did it. The 'logical OR' instruction is the trick. Don't expect me to explain 'logical' operations, read it up in a book like Martin Healey's 'Minicomputers and Microprocessors' (a bit pricey, but ideal fodder for frustrated electronics engineers. What's more, it's British, and part of the recommended reading for the Open University Computer Course). The thing about an 'OR' instruction is that when one register is 'ORed' with another, the Z flag will be set when, and only when, both registers are zero. So, bearing in mind that the HL register pair is in fact two registers, the ideal instruction in this case would have been OR H with L, but it doesn't exist. Don't forget that aritmetic and logical operations can only be performed with (or against) the A register. So the thing to do is to copy one register to the A register, and then OR the other register with A. So now I can guarantee to set the Z flag when both H and L are zero.

Better start learning about conditional jumps. Well these are the same as the unconditional jump mentioned earlier, but they only happen when the condition is true. So if I want my little delay subroutine to loop round until HL is zero, and I'm ORing H with L (via the A register) to set the flags, then testing the Z flag to determine this, the appropriate jump is a 'Jump Not Zero,nn', the mnemonic for which is JP NZ,nn. All set, I wrote an experimental program and tried out the 'B' Breakpoint command at the same time. My program looked like this:

```
0D00  21 00 00  LD HL,0000H
0D03  23        INC HL
0D04  7D        LD A,L
0D05  B4        OR H
0D06  C2 03 0D  JP NZ,0D03H
0D09  76        HALT
```

and executed it at 0D00H. After about one second, the HALT LED on the Nascom came on, so it looked as if it had worked. But how could I be sure it was doing exactly what I wanted. I had already found out that when a program 'crashed' it tended to go away, think to itself for a few seconds, and then the HALT LED would light. I suppose I could single step it, but pushing the button half a million times didn't sound like my idea of a fun evening, so what else? Maybe, step the first few times round the loop to see if the registers changed as I expected, and then set a breakpoint at the end where it would hit the HALT instruction, and see if the registers made sense. Ok?

```
S 0D00
1000 0D03 4800 0000 1000 0000     we've just loaded HL with 0000
                                  and PC is pointing to the INC HL
1000 0D04 4800 0001 1000 0000     HL has just been INCed by 1
                                  and PC is pointing to the LD A,L
1000 0D05 0100 0001 1000 0000     L has just been copied to A
                                  and PC points to the OR H
1000 0D06 0100 0001 1000 0000     H has just been ORed with A.
                                  No flags changed notice
1000 0D03 0100 0001 1000 0000     As there was no zero, the jump was affected
                                  and PC now points to INC HL again
```

looking good, so put a breakpoint at 0D09H to stop it when if 'falls through' the jump

```
B 0D09
```

and now tell it to execute from where it is

```
E
```

about 1 second later, the following appeared

```
1000 0D09 0044 0000 1000 0000     The F register has the Z and P/V flags set
                                  and the HL pair has 0000 in it
```

Well that just about wrapped up the program, all that was required was to put a 'call' to the subroutine in the right places, and it would be away. The finished program looked like this:

```
0D00  3E 2A     LD A,2AH
0D02  32 E2 09  LD (09E2H),A
0D05  CD 13 0D  CALL 0D13H
0D08  3E 20     LD A,20H
0D0A  32 E2 09  LD (09E2H),A
0D0D  CD 13 0D  CALL 0D13H
0D10  C3 00 0D  JP 0D00H
0D13  21 00 00  LD HL,0000H
0D16  23        INC HL
0D17  7D        LD A,L
0D18  B4        OR H
0D19  C2 19 0D  JP NZ,0D19H
0D1C  C9        RET
```

Success. A whole week of evenings, but I'd got there. The program wasn't very clever, but it was a program, and whats more it worked. I saved it on tape, and also wrote down exactly how it worked, so I wouldn't make the same mistakes again. I suppose in all, that took about 20 hours to write, about 5 minutes to do the right things, and 19 hours and 55 minutes of mistakes. Perhaps I could stave off the lynching party with my impressive demonstration of a blinking asterisk. So ends the story of my first program.

There are a number of points to be drawn from this long (and in many ways cautionary tale). First understand what the machine code instructions do. In many cases it is useful to write a simple little test program just to test the effects of certain instructions. Single stepping often reveals things going on (or not going on) that give insights into how the instructions work. The technical manuals often tell you explicitly what an instruction will do, but quite often neglect to tell you the side effects. This is particularly important when it comes to determining which instructions affect the flag register and in what ways. Often flags change (or don't change) for, at first sight, inexplicable reasons, and it's often up to you to figure out why. Another thing is always have a clear idea of what it is you are about. Drawing a flow chart is supposed to be useful. I personally rarely use them, but instead write down little lists of the things that should happen in a column. I don't worry about actual machine instructions at that stage, but the list 'flows' in a linear fashion rather like a program, and I find it easy to translate into instructions later. So my scribbled notes look a bit like this:

```
J3 Get the 2nd space recieved flag
   test if 1st space flag is still set
   if not set it again
   set 3rd space flag
   and go to J1
J4 test this char for space or char
   is control char?
   yes, go to J6
   is space?
   yes goto J3
```

and so on. In case you are wondering what that is, it's part of a proportional print routine I still haven't got to work properly (and I've been at it on and off for two months). Slowly, I've introduced assembly listings, in case you haven't noticed, thats what those lists of machine code instructions are called. Without a program called an 'assembler' using them is still tedious, as they have to be written down by hand, but the compensation is that by using the correct mnemonics, programs become much more readable, and so easier to understand. The preceding story taught me two things, patience, and a determination not to be beaten by a mere machine. The Nascom got its revenge on me also, after those first few hours, I became addicted. The thought that there is an electronic machine that I (even now) do not completely understand is compulsion enough to ensure sleepness nights, periods of mental abstraction, and withdrawl symptoms when I go on holiday. You have been warned.

The next episode will go on to talk about some more `correct' ways of approaching assembler programming, deal with relative jumps, and introduce the subject of `labels'. In the mean time, see if you can duplicate the second program I wrote. That started with an asterisk in the centre of the screen, followed after a delay, by a ring of 8 asterisks around it, repeating the routine 4 or five times over to produce an eight pointed radial star stretching to the corners of the screen. The star having expanded to the edge of the screen then shrank again back to a single asterisk. After the time it took to write the blinking asterisk routine, that one only took about 5 hours, but I found out what the IX and IY registers were all about.

Where.

Do You Know Where You Are ?

If you have ever tried to write relocatable code (say a subroutine that must be executable when stored anywhere in memory) you will know that invariably you need to find out where you are (where the subroutine has been put in memory this time).

For those of you who have not tried, a relocatable subroutine can be of great use because it can be included in any program, at a different address (place in memory), and only the program using it has to know where it is. To make this work however, all calls and all references to a data storage or work areas within the routine must not be at fixed addresses, but must be calculated from a known position. The result of the calculation is known as an offset.

The problem is the 'known position'. This needs to be found without the subroutine being told. A very simple method is to call a return instruction at a known place (say in the monitor), decrement the stack pointer and POP the return address into a register pair.

e.g. the routine starts at 0E00

```
addr  code    instruction
0D00  C9      RET             ;return instruction

0E00  CD000D  CALL 0D00H      ;call return instr. at 0D00
0E03  3B      DEC SP          ;decrement the stack pointer
0E04  3B      DEC SP          ;twice - addresses are two bytes
0E05  D1      POP  DE         ;put return address in DE register
0E06  18xx    JR   NXTINS     ;jump relative to next instruction
0E07  616263  DEFM /abc/      ;data held in work area
```

Note that the address now in register pair DE is 0E03 the address to which control returned after the call. Any calculations must be based on this address. To find the work area address 05 must be added to the value in DE.

```
    210500  LD   HL,05H  ;put 5 in HL
    19      ADD  HL,DE   ;add DE into HL
    EB      EX   DE,HL   ;put the answer back into DE
```

Now you know where you are. Easy really isn't it.

Printers

PRINTER PIE

by Len Ford

I first rubbed shoulders with computers some five or six years ago, though this was only in the data-prep department where 20-odd key stations loaded data to disks for subsequent transfer on to mag-tape and then to a main frame. However, I picked up a little bit about programming, and in due course learned how to display simple (and completely unauthorized) pictures on a VDU screen, with hard copy to a teletype. I also discovered, quite by accident, several methods of crashing system software, and derived a certain degree of pleasure plus appreciable easement of boredom from informing visting 'brass' that the system was 'down'. I also acquired a reputation for rapid restoration.

In due course, operating on the principle that a system which is working efficiently and satisfying all users is ripe for change, Management reminded all concerned that perfection is stagnation, to be avoided at all costs, by rehashing all routines and systems.

From my angle the only favourable effect of this change was the consequent opportunity to obtain hands-on experience on a CMC Realities terminal. Since I was a trespasser on the system some devious detective work was required to discover the passwords associated with the various accounts, such passwords being the pre-requisite to any worthwhile response from the machine. But with this information I was soon experiencing the delights of CMC Basic, a very powerful language. Star Trek or Hamurabi are fantastic games with 64K of immediate and masses of megabytes of virtual memory available.

As I mentioned, though, I was a completely illegal intruder in to the system, and always had to be ready to log-off whenever I saw someone approaching. The frustrations were nail-biting, and resulted in my purchase of a Nascom 1 some fifteen months ago.

I started off with a bog-standard system plus T2 monitor, but have expanded until now I have 32K of memory, T4 monitor and Crystal Basic 2.2, a language which started off in just under 7.5K, but is slowly increasing in size as I think of additional commands, write them in machine code and incorporate them in to the Basic, a fascinating process.

Despite the steady growth of my Nascom, the one peripheral which I really longed for was a printer. After putting together some 5 or 6K of machine code (this fella's a masochist, ED.), seeing it operate successfully, and writing it to tape, copy it down on paper the hard way was a real grind. And then I saw an advert for reconditioned teleprinters, at something less than 140.00.

Maybe I should explain that price. I have an arrangement with my wife whereby I may spend whatever I wish on the Nascom, provided she spends the same amount on jewellery or other goodies for herself. It makes Nascom add-ons a bit pricey - I would have liked an IMP, but at 650.00 (+ VAT)! However, to return to the teleprinter.

My wife knows nothing, or rather less, about teleprinters, but just when I was reasonably fixed on buying one, she noticed an article in one of my computing magazines wherein it was mentioned that teleprinters come in sizes ranging from that of an ordinary typewriter to something more resembling a grand piano. It took a lot of talk to persuade her that the one I was going to order was at the smaller end of the range, and I don't think I would have managed it had I not shown her a picture in an encyclopaedia of a nice small teleprinter (making sure my thumb obscured the caption which said "the latest model", and thus hardly likely to be for sale on the second hand market). In the end I ordered one model 7 Creed telepinter.

There followed a reasonably short period of gradually increasing impatience, until one morning a large yellow van pulled up outside our front garden gate. That rather shook my wife who thought the Creed would be delivered by the Postman, but when the van driver came to the door and said, "You'd better give me a hand with this, Guv.", and shortly afterwards she saw us staggering up the path carrying between us a box of almost unmanageable proportions and weight, her first reaction was, "I'm not having that thing in the lounge!". Which is where my Nascom lives, nicely tucked away on a tea trolley.

I must admit that there was some reason for my wife's ultimatum, not only in view of the size of the monster, but even more so in view of the noise it churned out. (I know someone who keeps a Creed in the bedroom, his wife's remarks about it are unprintable. Ed.) For a little while it looked as though I was going to have a real quiet time, and be the owner of a non-operating printer. However, after a little discussion and much heated argument, I finally installed my printer in the spare room, with a lead from the Nascom going up through the ceiling and across the floor of the loft (we live in a bungalow), and again through the ceiling in to the spare room, and so to the printer. I am now able to take hard copy of my programs, whether in Basic or machine code, and it is only occasionally that I have to go outside to explain to a street audience that the noise is only an adjunct to my full enjoyment of my Nascom and not a machine-gun attack on my family. Almost my only worry now arises from the fact that we are hoping to leave this address soon, and I wonder if I shall be able to install the Creed in as convenient a location in another house or bungalow: I fear that the noise would prevent our taking a flat.

One would have thought that it would now be an easy matter to submit programs to the Club library, but that is not so for several reasons. The first and most important, is that I do not have anything really worth submitting, secondly, I have got NASBUG T4, and it seems that NAS-SYS is all the rage. Lastly, although the printer will hard copy Basic programs, I haven't yet discovered how to run the printer with ZEAP, as entry to ZEAP immediately causes the teleprinter routine to crash. I am having a go at getting that one sorted out, and if anyone's interested I'll report on this later.

MY IBM'D NASCOM

by C.B.Frater

I decided to go about getting a printer the expensive way. I ordered a second hand reconditioned IBM printer from Display Electronics in Haywards Heath. With this, I also ordered an Aculab 735p interface. This has two advantages. The first is that I do not have to embark on the tiresome task of fiddling around with transistors, resistors, et al, which I do not really understand. The second is that using a golfball, I am able to use a number of different type faces, COURIER METRIC, COURIER, ASCII, BOOKFACE, SCRIPT, etc, which, as I had already purchased NASPEN, may well come in useful.

The officionados at Henry's Radio very kindly let me have a listing which enables a Nascom to drive a Centronics printer. As the Aculab makes the IBM look like a Centronics printer this was ideal.

The first problem I encountered was that I did not have the 26 way ribbon cable and a plug in my kit. This was apparently because I got an early kit, and I should have let Nascom know about this last December, but in my absent-mindedness I was not even aware that I should have recieved it. I couldn't work out what to do, nobody seems to have these leads and plugs in stock. (Most Nascom dealers now keep them, also AP Products - Saffron Waldon, and RS Components, ED.) I did what the high priests at the INMC are always telling us to do, I read the manuals. One of these days I'm going to rewrite the Nascom manual so that non-computer-minded folk like myself can understand it. When I finally realised PORT 4 was synonymous with PORT A and that PORT 5 was PORT B, I was able to deduce that only pins 1 to 16 were required to drive the printer. Nascom had provided the 16 way cable for the serial I/O which I had no use for, so I used that.

I connected everything up switched everything on and tried it. Nothing. My disappointment when the printer didn't leap into life when I typed on the keyboard was great. So I re-read the listing, and then again. I re-read everything I had. I even started to read the machine code part of the Nascom manual. I experienced the usual feeling of total blankness that occurs every time I open the manual at those pages. I phoned Nascom. I went to Henry's, but found them to busy to be of help, they tested the PIO though, it was ok. I phoned Aculab. Finally I wrote a long letter to Nascom explaining exactly what I had done, and after he came back from his holiday, I was able to speak to an excellent chap called Dave Lewis, who set my mind on the right track.

It seemed that Henry's listing converted all `carriage returns' to `line feeds, whilst the Aculab interface would only respond to `carriage returns'. He also explained that I would have to initialize the UOUT reflection (what ever that was). What this meant was that I had to insert the address of my printer routine into a location in the NAS-SYS workspace to direct the character to be printed to the printer routine. A little program was added to the front of the print routine to redirect UOUT automatically.

```
0C80  21 88 0C    LD HL,0C88H     ; Put address of print routine in HL
0C83  22 78 0C    LD (0C78H),HL   ; Load the contents of HL to 0C78H
0C86  DF 5B       SCAL MRET       ; Return to NAS-SYS
```

I typed this in using the `M' command at 0C80H, followed by the printer routine at 0C88H. What I didn't realise is that you have to run the program at 0C80H to tell the computer to redirect UOUT. I now have this down to a fine art, where the program (and the printer routine) is loaded from tape and then executes itself (see the `G' command in the manual). Next I type EB800 to execute NASPEN. Unfortunately, NASPEN assumes that a serial printer is in use at this stage, and it is necessary to change the NASPEN print reflection. I don't know what this means, but I do know that after executing NASPEN, I use its `N' command to return to monitor and alter the memory at location 101DH. My version of NASPEN has the following:

```
101D DF
101E 6E  (this must be changed to 75)
101F C9
```

There is a snag in this, in having returned to NAS-SYS, the UOUT reflection gets changed, so before returning to NASPEN, 0C80 has to executed again. So you have to type in E0C80 followed by EB806 (N.B. B806, NASPEN warm start). If you typed EB800, NASPEN would change the byte at 101E back to 6E, and the whole process would have to be repeated.

So I wrote a small piece of text using NASPEN and used the `P' command to print it. The printer burst into life EUREKA!!!!! -- ??? Well, it worked, but I wished I had a gothic golf ball, as the printer cleverly converted my piece of text into Chaucerian English with lots of repeated letters and lots of letters left out. Aculab provided the cures for this. The missed out letters only occured after spaces. This was because the device inside the printer which makes spaces was worn, and instead of turning once for each space, turned twice instead. This didn't print two spaces but it did send two pulses to the interface, and thence to the Nascom, telling the Nascom that it had already printed the next letter. Aculab told me how to adjust the pawl and ratchet under the printer mechanism which controlled the space bar clutch. A little bending soon put that right. The repeated letters came from noise on the data strobe, and Aculab suggested a capacitor of about 500pF between the strobe and ground. I found that 1000pF was satisfactory.

So after considerable effort, (mainly on the part of the suppliers) my printer worked, worked far too well, because I now have a sort of keyboard mania and can't stop typing. I hope this information will be helpful to someone.

Misc.

VIDEO PROBLEMS

Some time ago we had a letter from Mr. Willmott of West Drayton in Middlesex, who raised an interesting problem concerning the screen blanking on a Nascom 2. Mr. Willmott wishes to remove the undesirable black screen flashes which occur when using the graphics. (These occur because considerable efforts were made when N2 was designed to remove the white screen flashes caused when the processor accesses the video RAM during normal screen read/write operations. Unfortunately because of this, screen blanking becomes very visible when large areas of white are displayed in the graphics mode.) So here is Mr. Willmott's letter, if anyone has an answer, they're welcome to write to us.

Dear INMC,

The `snow plough' flash supressing circuitry on the Nascom 2 is very annoying when one uses the NAS-GRA graphics option. Also rapid access to the video RAM destroys the displayed picture. I have been attempting to find a way round these problems.

Firstly I ran the following machine code program:

```
0D00  21 E0 09                LD HL,09E0H
0D03  7E          REP         LD A,(HL)
0D04  EE 40                   XOR 40H
0D06  77                      LD (HL),A
0D07  18 FA                   JR REP
```

Then I tried reducing the value of R75 associated with IC58. A value of 5K6 appeared to be optimum. However, the blanking is still significant.

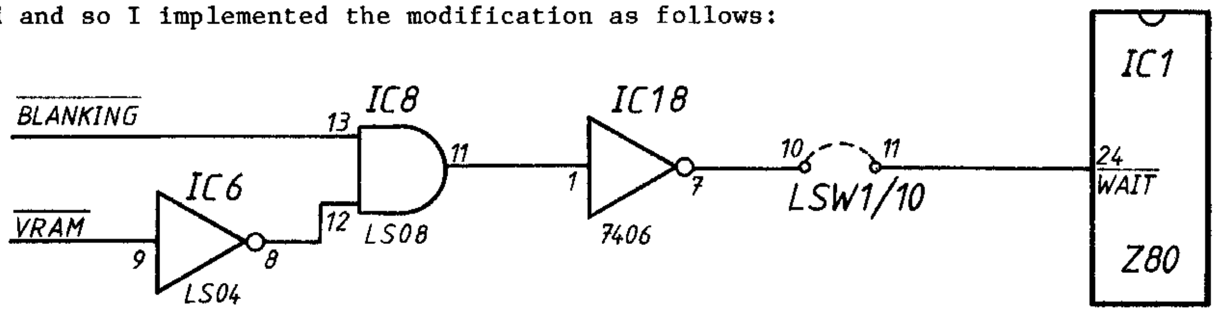
I tried attacking the heart of the problem, ie; resolving the conflict in access to the video RAM by the VDU and CPU circuitry. In other words, I decided to make the CPU wait when accessing the video RAM if the contents were being displayed. Succinctly:

$$WAIT = \overline{BLANKING} . VRAM$$

using the available signals:

$$\overline{WAIT} = \overline{\overline{BLANKING} . \overline{\overline{VRAM}}}$$

As I had implemented the no wait modifications on the RAM board in INMC NEWS issue 7, I was able to use some of the wait circuitry. I also found a spare inverter on the board and so I implemented the modification as follows:



The theory seems ok, but the practice is not! The best I could manage was by putting pin 12 of IC8 to a high. This had the desired effect except that:

- the first two columns of the VDU flickered
- there were occassional flashes on the screen
- the Z80 execution rate at 4MHz was greatly reduced.

The VRAM signal should have overcome the last problem with little effect on the execution rate. Perhaps I need a blanking signal that starts earlier along the line - this would prevent the CPU using the video RAM when the VDU wants to start displaying a line. However, I do not understand why including the VRAM signal should freeze the board.

Would you care to help or advise me or even try the modification and make it work? I'm sure other members of the INMC would be interested.

Yours sincerely
S. C. Willmott

Whilst on the subject of video problems, about the same time as the above was recieved, we had a little note from D. R. Humphries of Maypole Heath, Canterbury. Many N2 owners will be aware of the 'nasty' shape of the LD signal to IC65 which manifests itself by making the video display the front half of each character twice. The recommended cure is to zap a 100pF C from pin 1 of IC65 to ground. Not very elegant!! Mr. Humphries suggests changing IC71 from a 74LS20 to a 74LS13. This works well, and is nice and tidy. The same trick works with the shifted video blanking problem that can occur when IC55 is a bit slow. This time IC55, the 74LS20 may be replaced with a 74LS13.

Yet another PRINT USING routine

by J. C. Curtis

The routine involves deciding the largest number of significant figures to the left of the decimal point and assigning this to variable SF. Then decide how many places to the right and assign this to DP (SF > DP please). The variable is assigned to Z and is returned in Z$.

```
1000 Z$=STR$(INT(ABS(Z)*10^DP)
1010 Z$=RIGHT$(Z$,LEN(Z$)-1)
1020 IF LEN(Z$)=SF THEN 1050
1030 FOR I=0 TO SF-1-LEN(Z$)
1040 Z$="0":NEXT
1050 IF DP=0 THEN 1070
1060 Z$=LEFT$(Z$,SF-DP)+"."+RIGHT$(Z$,DP)
1070 IF Z<0 THEN 1090
1080 Z$=" "+Z$:RETURN
1090 Z$="-"+Z$:RETURN
```

Moot point

by Dave Hunt

For the last two days I've sat at the keyboard of my computer, typing up stuff for this issue, I've a file an inch think in front of me, full of readers' letters and articles. What really depresses me is the number of beautifully written (and in many cases, illustrated) articles on "How I built my Nascom and made it work", which of course we will never print. You see, if you think about, you didn't learn about the INMC until AFTER you'd built your Nascom, 'cos by the time you had written to us, and maybe even got some back issues, you'd already rushed ahead and got it to work. Hadn't you? So much as we like to encourage articles, no more on constructing Nascoms please. Because by the time we could put them into print we would only be preaching to the converted. Thanks for all those construction articles we haven't printed (we have about 30). It's not 'cos we don't like you, or that they aren't good enough, it's just that they are irrelevant.

I wondered why they kept coming, then I realised that it was because we hadn't said we didn't want them. Almost every Nascom owner has constucted his (or her, any female members out there?) Nascom and is justifiably proud, and that is probably, then, one thing almost every member is qualified to write about. If on the other hand, you really want to know how much trouble the next guy had, then let us know. We'll print'em all as a serial.

Instant diagnosis

In my pile of letters I've found a number of small queries which I haven't replied to. I decided that it probably wasn't worth it as most of my mail is at least 3 months old. So instead I'll precis them here and maybe answer some of those unasked questions you feel some one might have the answer to.

M. R Hughes asks about the fault on the video where the video dot generator (IC65) on a Nascom 2 loads the front half of the video data twice. If you use a scope you'll notice that the LD waveform on pin 1 of IC65 has a nasty spike in it. It comes from IC71. The cure, to change IC71 to a 74LS13.

Another is the sideways 'weave' of the N2 display. This is caused by the fact that the frame sync pulses are not exactly 50 Hz (about 50.15Hz actually). If you're using a TV or monitor with poor smoothing, or there is ripple on your Nascom power supply then this will be apparent to a greater or lesser extent as the 50 Hz beats with the 50.15 Hz, and moves the line sync pulses in sympathy. There is no simple cure except to ensure that both Nascom and video monitor PSUs are ripple free. Bits and PCs have a mod up their sleeves which alters the frame rate to 50Hz, but this involves cutting tracks and wire links.

There have been some comments about the INMC book of Basic programs. One thing we totally missed when we printed it was that the old faithfull IBM didn't have an up-arrow, instead it printed a fullstop. This had the unfortunate affect that some thing like X=30^2 got printed as X=30.2, which if course is something else entirely. We haven't been through the book to find them, but watch out!!!

Another has been about saving a Basic program after a system 'crash'. Now this should never ever happen in Basic, and if it does, then suspect a touch of memory plague (read the index in INMC 7 for all references to memory speed and plague). Anyway, given that something nasty has happened, a Basic program is likely to be complete up to a point, followed by rubbish. This is because Basic has lost the pointer to the next line. It is possible to reconstruct the program by going into memory using the machine code 'M' and 'T' commands, and once you understand the way Basic puts pointers into each line, terminate the last valid line with three NOPs. I won't go into detail, 'cos if you're clever enough to figure out how it works, you're clever enough to sort the problem yourself. Otherwise 'LIST' the program to tape (refer to INMC 7 again) and then reload it just short of the point where the data becomes corrupt. Moral, always dump any program to tape before running it.

Watch out, pin connections to the serial and parallel outputs of Nascom 1 and 2 differ, make sure which version is meant before connecting anything to these pins.

Turning a parallel printer on and off from Basic is easy, assuming the U command is used and has already been activated. To turn on simply DOKE 0C78H (3192 decimal ... I think) with the address of your print handler. To turn off, take a DEEK at 3192 when the 'U' command is unactivated ('cos it varies between NAS-SYS 1 and 3) and use that number when DOKEing location 3192.

One thats in the manual but yet remains a hardy perennial. You can enter lines up to 72 characters in length into the 8K Basic by the following sequence. Note that 72 is the maximum number of characters that can be entered.

```
MONITOR
X0
Z
Ok
1233 REM This is the line you wanted to enter and may be up to 72 chars
MONITOR
N
Z
Ok
```

We have often been asked what golfball we had on the old IBM. Sorry but we don't know, it was 10 pitch and had M2 written on it.

Don't forget to type 'ENTER' before going on to a new line.

In INMC 7 we talked about the IEI IEO daisy chain priority. We mentioned that the N2 board could be made either high or low in the chain by its position on the buss relative to the I/O devices. If highest priority is not required, then the 10K resistor, R40 should be removed.

The BAI and BAO lines are never likely to be used in the Nascom, but they are intended to give priority to devices which use DMA (direct memory access to take over the buss from the Z80). The daisy chain works in the same way as the I/O daisy chain, using a DMA controller chip.

It seems we've never mentioned the 'A$' problem at 4MHz on an N2 except in the Basic Progs Book. Run this program:

```
10 A$="JHGUOYUUT - ANY OLD RUBBISH - GHKYTIUO"
20 FOR A=1 TO LEN(A$)
30 POKE 3017+A,ASC(MID$(A$,A,1)):NEXT
40 CLS:GOTO 20
```

If it runs, then Ok. If after the computer has had time to warm up you get:

```
?FC Error in 30
```

then you've got it. Another symptom is that ZEAP will convert opcode 2A into 21 so you get this:

```
21780C          LD HL,(0C78H)
```

as a final touch it will drive the Gemini/Henelec disk system mad (if it works at all). This all comes about because there is a nasty 'skew' on the 4MHz clock signal. It's well and truly out of spec. Connect a 220R resistor between pins 6 and 14 of IC11. That helps but does an OoH-nasty to IC11. The laws of physics say that IC11 must fail sometime, but it took the shop demo machine 11 months to die, and thats on 9 hours a day 6 days a week. For the 42p the 74LS14 cost it was a small price to pay. There are much more elegant cures, but they are messy and complicated (unless, of course, someone somewhere has cracked it).

Lots of confusion is caused when adding EPROMs to the N2 board. Blocks A and B never seem to stay still, and are guaranteed to be wrong whatever you do. Now this is a knotty one, and difficult to explain. It all concerns the way IC46 works. The simplest way to think of it is that the blocks change over at each 4K boundary, A becomes B, B becomes A. This gets even more fraught when non consecutive 4K blocks are addressed. Perhaps you should do what I do. Write a small machine code program that continally reads the start address I want, and use a scope to see which (if any) of the chip enables (pin 20) is waggling up and down. We'll deal with this one with a proper article when we can find someone rash enough to write it.

Gemini G805 / 'Henelec' Disk System Rules.

Following our comments in the last issue re. competitive disk systems for the Nascom, the following figures have been brought to our attention:

SYSTEM	APPROX. SALES	
Airamco System	0	(Product never taken into production)
Comp Disk System	30	
Gemini G805	200	(Uses 'Henelec' controller card)
'Henelec' card (kit)	50	

Machine Code Programming for Nascom 1 and 2

or "How to find out what it really does" - a sort of review by K. Hamlyn.

When I started to look at this volume, I was nearly a Dodo - not quite, because I could spel "Zeep". However there was a vague impression in the back of my mind that a "shift right with carry" might be connected with the political activities of some members of the Parliamentary Labour Party.

Seriously, though, having tried two other books, this one provided a refreshing view onto a subject which has an unwarranted mystique, brought about by people who can speak the jargon. As an interpreter it is most useful.

You start by learning the fundamental operating procedures of the strange 16 base arithmetic, followed by a rundown on the ways in which the microprocessor and memory work. Having been given an outline of flowcharting, the reader needs to sit by the keyboard of his machine, as the rest of the book is entirely practical. Having put in the first program, and executed it, the HALT light glows and the reader bounces slightly because it has worked first time and he eagerly anticipates the next one. In working through this course, there is a definite wifeware problem and a drain on the coffee stores as the dawn breaks and the sparrows go coughing and stomping in the gutters.

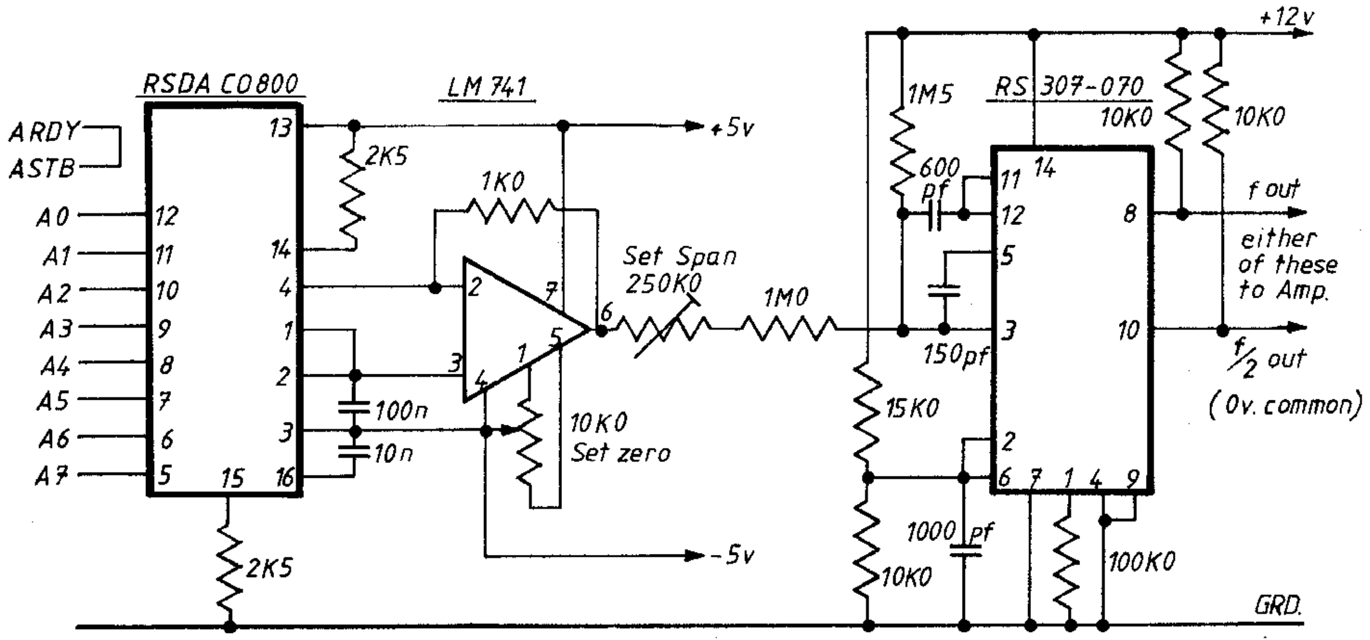
Finally there are some useful circuits which enable the reader to monitor the ports and to operate in an analogue mode from and to these ports.

This is a well written book which will help the prospective machine code programmer to go from Dodo status to a reasonable standard. A necessary volume for any Nascom owner.

'Machine Code Programming for the Nascom 1 and 2' is published by Interface Components, costs 4.95, and is available from your Nascom distributor.

Circuit Diagram For A Musical Nascom

(See following article)



If Music Be....

Musical Nascoms

===============

by M. L. Trim

I started out knowing nothing, but have slowly learned over a period (with a little help from articles in the newletter). The back issue articles on the PIO are a little heavy going, and being mainly concerned with the use of interrupts (vital information I'm sure ... but), tend to gloss over the obvious. The obvious is how to get information into and out of a PIO in the first place. Reading and understanding the manuals is most important, but these tend to leave out important points. The manual makes it quite clear that before a port within a PIO can be used, a control word must be send to its appropriate control port to instruct the port which mode is to be used. But how to construct the control word. My PIO manual states on page 10 that the operating mode is selected as follows:

```
M1  M0  X   X   1   1   1   1
D7  D6  D5  D4  D3  D2  D1  D0
```

(refer to the manual for greater detail), but unless you realise the above is a bit pattern which must be converted to a HEX number then its relevance is unknown. For example, to select the output mode, M1 and M0 are both 0, and X may be anything, for convenience, 0, therefore

```
0 0 X X 1 1 1 1  =  0F
```

(For more detail on converting bits to HEX refer to the Kiddies Guide part 1, INMC80-1.) So to initialize port 4 of the PIO to output use the following piece of code, which must be sent to the port 4 control port, port 6:

```
3E 0F   LD A,0FH         ; Load A with the initialization code
D3 06   OUT (6),A        ; Send contents of A to port 6
DF 5B   SCAL MRET        ; Return to monitor
```

The PIO port is now initialized to output until another control word changes it. Note that HALT could have been used instead of a return to monitor, but this would have required RESET to start the processor again, and Nascom 2's reset the PIO when RESET is pressed. HALT would be quite satisfactory with Nascom 1. Having set the PIO to output, the 'O' command may be used to test its affect on port 4.

```
ie      O 4 00  sets the PIO output to zero
        O 4 FF  sets the PIO output to FF (or max)
```

use Q 4 to see if the PIO registers echo the data sent.

Having mastered sending data to the PIO, it is easy to grasp the opposite, inputting data. First send the appropriate control word, to initialize the port to input, from then on the PIO will be set to receive data until the control word is changed (or the PIO reset). One minor complication, and that is the input to the port is latched, and the latches are opened by the STB input going low. This is used to cause an interrupt if the interrupt mode is selected. But in the simple case, no interrupts are required, and data may be sampled at any time, so the input latches are not required. The latches may be 'opened' permanently if the STB input is tied to 0V.

Fine, what next? Martin Dyer helps here with his suggestion in INMC80-1, let the Nascom produce music. Here is my answer. For this we need a D to A convertor, I bought an integrated circuit from Radio Spares (they get 'up-tight' if you call 'em that these days, they prefer RS Components. Ed.) to achieve this, part 309-458. The circuit is shown on the attached circuit diagram, along with another IC from RS, namely a voltage controlled oscillator, part 307-070. (The circuit also shows an unmarked op-amp which from the pinout we presume is an LM741. Ed.)

Using a small bit of machine code, and an amplifier attached to the VCO, we can program notes. The program initializes the PIO to output, then scans the keyboard for a character. When a key is pressed, the character is sent to the port, changing the analogue voltage produced by the D to A convertor, which in turn changes the frequency of the VCO. The character pressed is sent to the screen for reference.

```
0D00   3E 0F              LD A,0FH        ; Initialize PIO to output
0D02   D3 06              OUT (6),A
0D04   CF        LOOP     RST RIN         ; Scan KBD for input
0D05   D3 04              OUT (4),A       ; Send the character to PIO
0D07   F7                 RST ROUT        ; Send character to CRT
0D08   18 FA              JR LOOP         ; Go round for next character
```

Execute the program at 0D00, and then type keys, different keys will result in different notes.

I prefer to program in Basic, so how do we use Basic to work the PIO. Answer, read the manual and you will find that
OUT 6,15
sets the PIO to the output mode, note that 15 is the decimal of 0FH. To output from the PIO in Basic, use
OUT 4,A
where A is a decimal value between 0 and 255. We can now write a program to play a simple tune, in the following example I used the machine code program to find the notes, converting the character displayed to decimal, and then putting the number into the data. Happy tune playing.

NATIONAL ANTHEM
===============

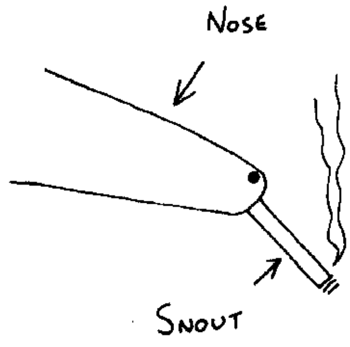
```
10 REM Set output port to output mode
20 OUT 6,15
30 REM Set up number of notes
40 DIM N(50)
50 FOR L=1 TO 50
60 READ N
70 OUT 4,N
80 FOR I=1 TO 250:NEXT I
90 NEXT L
100 RESTORE:GOTO 50
1000 REM Note table
1010 DATA 71,71,78,68,71,78,88,88,94,88
1020 DATA 78,71,78,71,68,71,71,78,88,94
1030 DATA 103,103,103,103,94,88,94,94,94,94
1040 DATA 88,78,88,94,88,94,88,78,71,88,94,103
1050 DATA 114,94,88,78,71,00,00,00,00,00
```

Option to give a mandolin effect insert the following

```
70 FOR I=1 TO 5
72 FOR K=1 TO 10
84 OUT 4,N:NEXT K
86 OUT 4,0:NEXT I
```

M. L. Trim

VERY OFFICIAL



HELP! Has anybody seen Lawrence since this shot was taken in INMC80-1? Have the Police got him? Where is Heloise? Don't watch this space for further details. Aaaagghhhh

ERWSMTFTH.

Cartridge Drive or Stringy-Floppy?

by C.W.Hobbs

This article describes the fun that I had interfacing the "Electronic R/W System Model 25-300" to my Nascom 1. The peripheral itself, whilst certainly electronic and certainly a read/write system, is probably better described as a cartridge drive, although I have seen it described in some magazines as a Stringy Floppy; a completely misleading and unnecessary name.

I ordered the unit by telephone from Breda, Holland one Wednesday morning after having all my technical queries answered immediately in excellent English and the unit arrived the next day by post (I must admit though that I live in Holland).

For 330 Guilders (about 73 pounds and dropping as Mrs. Thatcher pulls the pound through the ceiling) I received a deck and control board. For a further 80 Guilders (work it out for yourself) a box of cartridges (wafers) of assorted lengths from 5 feet (20 seconds recording) to 50 feet (200 seconds recording). These cartridges consist of an endless loop of tape with an End of Tape (EOT), which is also the beginning of tape (BOT), marked by a piece of tin foil.

Unlike some pieces of kit, this drive came with very adequate documentation but no case and no mounting brackets. Any capable metalworker with nothing more than a small lathe, milling machine and borer could knock up a suitable mounting in a few weeks. Anyway, I hacked and puffed and blew and finally mounted it in such a way which I do not feel like exposing to public ridicule. I feel more confident on the grounds of electronics and program design and will hastily move on.

The drive demands very modest +5V and +12V supplies and in return offers the following interfaces on a small edge connector :-

Input	Output
Start Motor	EOT Detected
Fast Speed on Motor	Cartridge is Write Protected
Enable Writing	Serial Data from Cartridge
Select the Deck	
Serial Data for Recording	

The beauty of the "Select the Deck" input is that when it is held false everything in sight goes into a high impedance state and many decks may be connected together.

The hardware interface wins no prizes for complexity but is very adequate and is shown on another page.

The LED's are necessary because once the cartridge is in place it is impossible to see whether you remembered to take the write protect tab off. The 100K resistor pulling up the Read line to the UART serves not only to make the pulses nicer but it is essential because when the deck is not selected the Data Out line goes high impedence and picks up all sorts of rubbish. And what does the Nascom do when it is idle? It picks the rubbish up and displays it at great speed on the screen making it impossible to type a command in to select the deck again to stop the monitor picking up the rubbish .... I will be honest and admit that this resistor was not part of the original design.

After a simple interface and a few test programs I felt ready to fill my second 2708 socket with the Hobbs uncopyrighted Cartridge Operating System; surely an impressive title. Here I tempered good design techniques (I design computer systems all day as well) with the knowledge that it had to all fit onto 1K bytes plus whatever I could save by taking Dump and Load out of Nasbug. The original 4 layer design has reduced to three layers :-

A) Drive level interface with routines for -
```
        Rewinding a cartridge
        Erasing a cartridge
        Pre-marking a cartridge
        Finding a sector
        Reading a sector
        Writing a sector
```

B) File level interface which sits on top of the drive level interface and offers routines for -
```
        Creating a file
        Erasing a file
        Opening a file
        Writing a sector of a file
        Reading a sector of a file
        Returning the status of a file
```

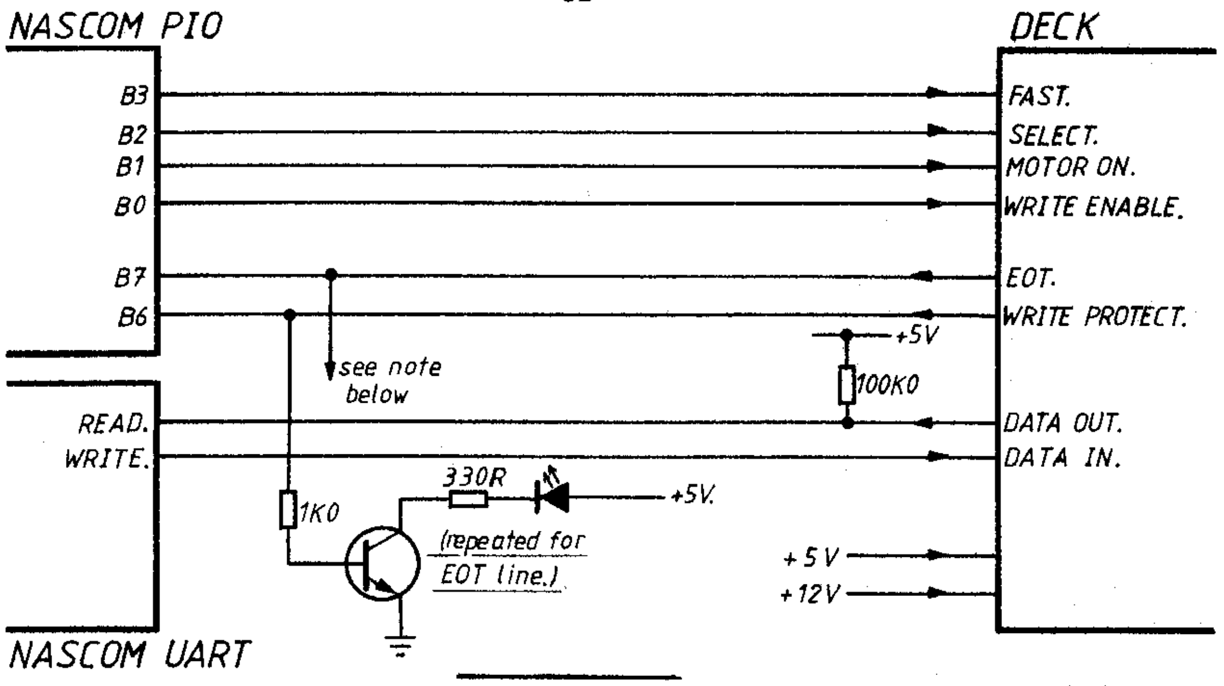
C) Operator level interface which sits on top of both the other levels and which offers the following commands, augmenting those of Nasbug -
```
        1) Drive level commands
                W3   to rewind the cartridge in drive 3
                       and initialise the PIO
                V5   to display the catalogue of files
                      on the cartridge in drive 5
                P6 1234  to initialise a new cartridge
                      in drive 6 as number 1234

        2) File level commands
                R5:PRINTFILE   to remove the file PRINTFILE
                      from the cartridge in drive 5
                A3:PRINTFILE 12  to add file PRINTFILE,
                      12 sectors long, to the cartridge
                      in drive 3
                L3:PROGFILE    to load the object code
                      in file PROGFILE on the cartridge
                      in drive 3
                D1:PROGFILE 400 800  to dump the contents
                      of memory between 400H and 800H
                      to the file PROGFILE on the
                      cartridge in drive 1
```

So what does the tape actually look like when viewed under a magnetic microscope, how are the sectors formatted, what does the directory look like, how do I fit it all into 1K bytes? Indeed do I? I will refuse to answer here but will try to draw some morals. (Cor, what a cop-out. Come on lets have a listing of it for the library. Ed.)

* Make the hardware as simple as possible at first. Software is more flexible and once you know the interface really well it is easy to increase the load on the hardware and decrease the software.

* Separate the software functions into clearly defined layers, decide whether subroutines will save registers or not (of course they will! Ed.) and start the design at the top (what do I want it to do?) and the coding at the bottom (how will I do it?).

* When you have only 1K bytes of memory do not stick too rigidly to the strict layering of the design.



STRINGS

How to save strings or string arrays on tape in BASIC.

by David Reddington

One of the shortcomings of Microsoft's 8k BASIC as implemented on a Nascom is its inability to CSAVE string arrays. Even to save numeric data requires the whole numeric array to be saved when perhaps only a few elements contain valid data.

The way round this problem is to utilise the ability of NAS-SYS to change the destination of the output. Change the reflection at $OUT (0C73 or 3187 Decimal) from 077F (CRT) to 077B (SRLX) thus:

```
10 DOKE 3187,1915
```

All subsequent PRINT statements will print to tape until output is switched back by a DOKE 3187,1919.

NOTE: a DOKE 3187,1914 will cause all output to go to both the CRT and the CASSETTE.

A typical program might be:-

```
10 REM A PROGRAM TO SAVE X,Y,Z,T$, AND A$(100) FOR NAS-SYS 1
20 DEVICE=3187:ZCRT=1919:ZSRLX=1915
30 PRINT "Start recorder in RECORD mode - then press RETURN"
40 INPUT A$:PRINT "PLEASE WAIT"
50 DOKE DEVICE,ZSRLX
60 PRINT X;Y;Z;
70 PRINT T$;
80 FOR N = 1 TO 100
90 :  PRINT A$(N);
100 NEXT
110 DOKE DEVICE,ZCRT
120 PRINT "DATA SAVED - SWITCH OFF RECORDER"
```

To Load the data you merely:-

```
200 INPUT X,Y,Z     220 FOR A = 1 TO 100
210 INPUT T$        230 :  INPUT A$(A)
                    240 NEXT
```

Useful Subroutines in BASIC

F9AD	Prints HL in decimal on the screen
F210	Prints the text pointed to by HL (Until it hits a 0).

Jumper.

RESTART JUMP MULTIPLEXER FOR NASCOM 1

by M. V. A. Alberry

Shortly, with the advent of new hardware opportunities, ie: disks with CP/M, many Nascom 1 users will no longer want to RESET to 0000H, as unfortunately CP/M requires RAM down there, and system ROM somewhere towards the top of the memory map. The circuit used is lifted straight from the Nascom 2 circuit diagram, so I claim no originality - however, it works, so I pass it on to others.

The object of the exercise is to force the top four address lines (A12 to A15) to some value other than 0000H on 'reset'. This is required for the first instruction cycle only, and this forced condition must be released before the next instruction cycle. What happens is that on 'reset', the top four address lines go to 0H (as usual), but because of the action of the 'reset multiplexer', the top four address lines are modified to the required address. This condition remains true, commencing at the first M1 cycle following the 'reset', and termiantes on the next following M1 cycle.

To ensure proper operation of the program thereafter, the first instruction encountered at the multiplexed address must be an absolute unconditional jump (C3XXXX) to the next instruction. This will force the next instruction address into the program counter, and from then on operation will continue normally, and the effect of the 'reset multiplexer' can be removed. For example, we wish to 'reset jump' to F000H, and then continue the program from that address. On 'reset', an M1 cycle is put out by the Z80, and an op-code fetch occurs. The Z80 address bus is set to 0000H, but due to the action of the multiplexer, the op-code is fetched from address F000H. This op-code must be an immediate jump instruction, and for sake of argument, we will make this an absolute jump to F003H. The code will look like this:

```
F000  C3 03 F0      JP F003H
F003  XX            Next instruction
```

as the op-code will be translated by the Z80 as an absolute jump, the next two bytes will be fetched, advancing the program counter to 0003H, and the multiplexed address to F003H. Having fetched the byte at F002H, the address F003H will be placed in the program counter. The Z80 will then put out another M1 cycle to fetch the next op-code, this time from F003H. The falling edge of the M1 cycle will cancel the 'reset multiplexer' and the address bus will carry the next address put out by the Z80, which will be F003H as required.

The circuit is very simple and uses a 74LS257 (quad tri-state multiplexer) to gate the forced address onto the bus, or to gate the processor bus onto the bus depending on the condition of pin 1. The 74LS257 may be likened to a 4 pole 2 way switch. As we want to intercept the address lines prior to them getting to NASBUS, the buffer board is the logical place for it. Fortunately for us, Nascom left a vast open space on the buffer board, and this is ideal area for building a small 'piggyback reset multiplexer'.

The four top address lines, M1 and RFSH all pass through IC3 on the buffer board on their way to NASBUS, so by removing this IC and moving it to the 'reset multiplexer' board, we have an ideal connector point for the 'reset multiplexer' board via a 16 way cable and header plug. Only one minor modification is required to the buffer board itself to get the RESET signal on to the 'reset multiplexer' board. Again fortunately IC3 has two gating inputs on pins 1 and 15. We only need one gating signal to the 'reset multiplexer' board, so one pin is spare. It is most convenient to use pin 1 for the RESET and pin 15 for gating. Thus we need to cut the track between pins 1 and 15 of IC3. As this is on the top side of the pcb, this can be a bit tricky. Remove IC3 and you will see two adjacent tracks (See alongside circuit diagram.)

Cut the track indicated and try not to cut both (as I did), and connect RESET from some suitable point on the buffer board to pin 1 of IC3 via a flying lead. Use a piece of veroboard about 2" square and construct the circuit shown on the circuit diagram, bolt (or use double sided sticky) the veroboard to the buffer board as close to IC3 socket as is convenient, having prepared a suitable length of 16 way ribbon cable with 2 16 way header plugs to connect from the multiplexer board to IC3 socket. A tip, my board didn't work first time, so I found 12" of ribbon cable useful, so that I could get at the 'piggyback' board to modify it. Having found and corrected the fault, the ribbon was shortened, and the board mounted permanently.

A word of warning about the action of the 74LS74, it fooled me (fortunately I know Dave Hunt's phone number, and he provided the answer). The 74LS74 circuit is cleverer than it looks, it's primed by the RESET signal, and then counts the first M1 signal it sees, it ignores the rising edge of the first M1 signal, but the output is tripped on the falling edge of the next M1. This occurs before the next address from the Z80 has stabilized on the bus and not after, as I thought.

The selection of the required jump address is made by forcing pins 3, 6, 10 and/or 13 low (that is, connected to 0V), or leaving the connections open so that the 10K resistors pull the inputs high.

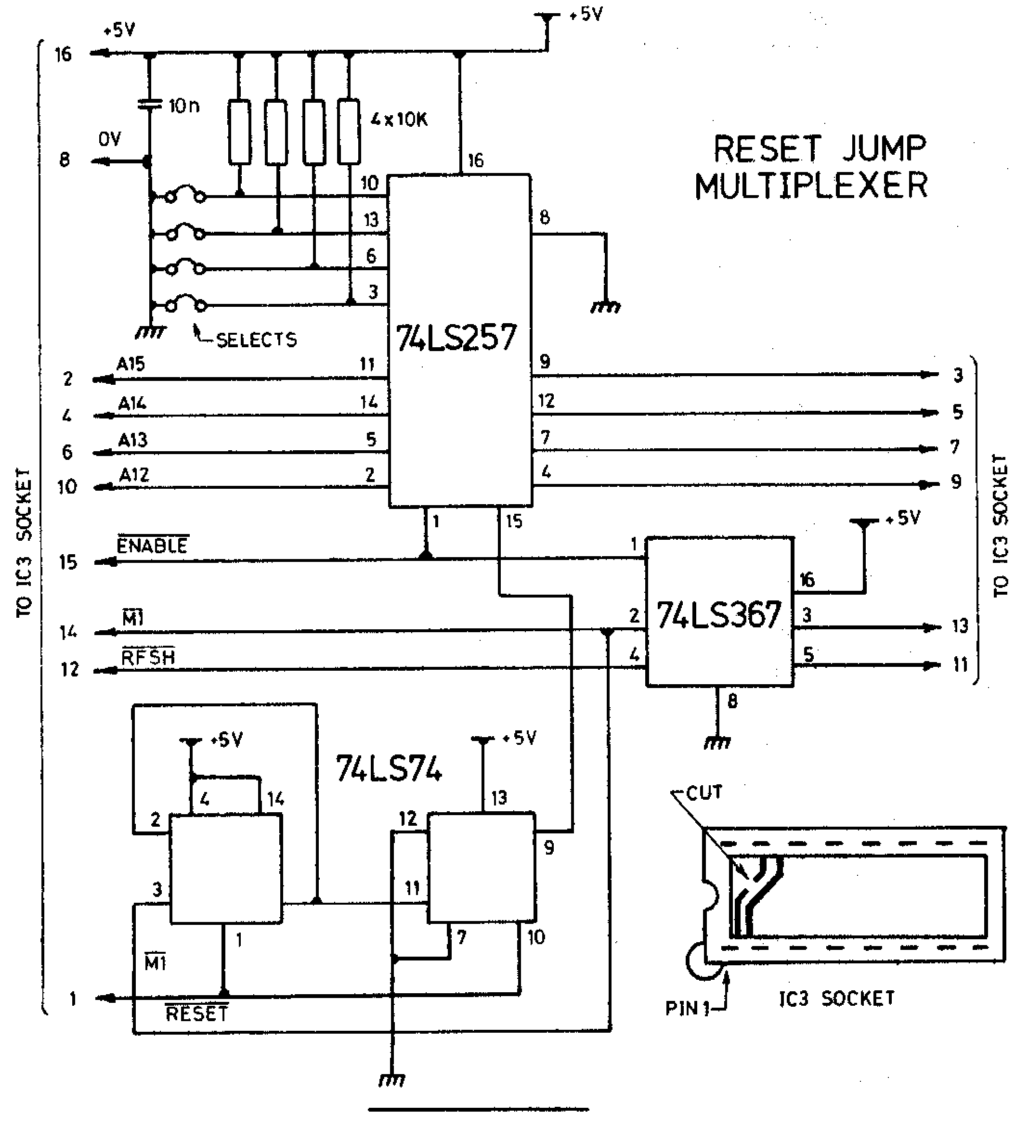
A12 = pin 3 A13 = pin 6 A14 = pin 13 A15 = pin 10

Personally, I use a DIP switch here, but permanent links could be used. You could use this circuit to jump straight to Basic on ROM (at E000H) if you wished. If you want to move the EPROM area and the video up to F000H (for CP/M, or other uses, then this is simply accomplished by selecting P4 to 12 on the RAM (A) board. This allows MEXT to pick up an F000H decode on 'reset'. Obviously, this may be any one of the 16 possible 4K boundaries allowed by the RAM decoder.

Parts list

1	74LS257	1	74LS74
1	74LS367 (IC3 from buffer board)	2	16 pin dil header plugs
3	16 pin dil sockets	1	14 pin dil socket
6"	16 way ribbon cable	1	4 pole, dip switch (optional)
1	pc Veroboard	4	10K resistors

(Ed. - Gemini Microcomputers have brought out a small kit based on this circuit and it uses a wire-wrap socket to plug into the buffer board. Price 10.00 + VAT and available from your distributor.)

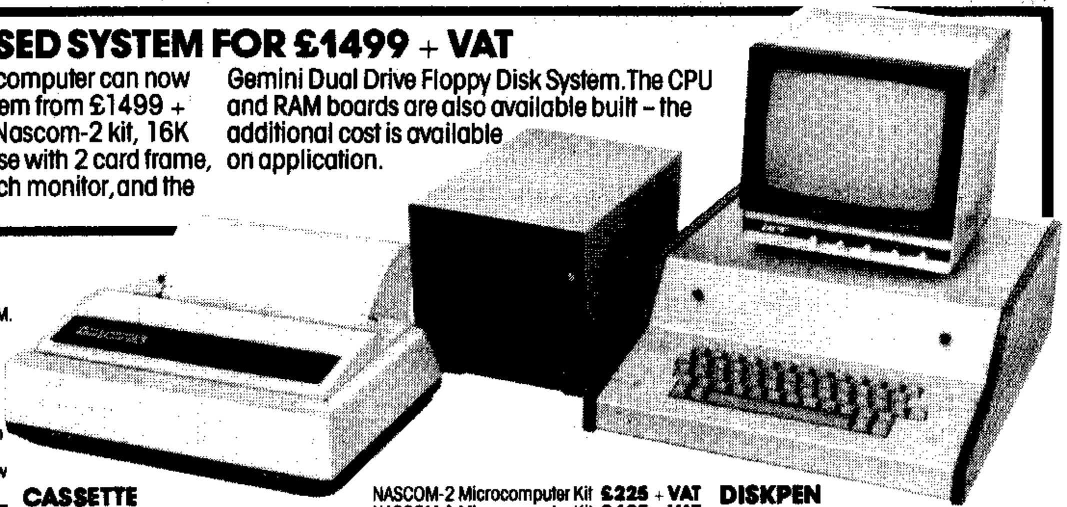


NASBUS NEWS

More great deals from 6 Nascom Dealers

A NASCOM-2 BASED SYSTEM FOR £1499 + VAT

The proven Nascom-2 microcomputer can now be bought as a complete system from £1499 + 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 hardwood 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

The Nasbus compatible EPROM board accepts up to 16, 2716 or 16, 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. **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 with optional Page Mode facility. **Prices: £90 (16K), £110 (32K), £130 (48K), £150 (64K). Add VAT to all prices.**

All prices are correct at time of going to press.

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 £40.50 + 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.**

PROM-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. **Price £25.95 + VAT.**

WE STOCK Nasbus COMPATIBLE COMPUTER PRODUCTS

All the products are available while stocks last from the Nascom dealers below.
(Mail order enquirers 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).

NASCOM 1 & 2 - MUSIC BOARD

Extremely simple to program in basic or machine code.
Generates notes over an 8 octave range.
Driven from parallel output ports.
Plugs into Hi-Fi for full sound.
Sustains notes with no CPU overhead.
Complete with full documentation and programming examples.
£18·65(kit), £21·65(assembled and tested).
Multiple boards may be used to great effect!

REPLACEMENT NASCOM INTER-CONNECTING CABLE WITH MUSIC BOARD HEADER CONNECTORS - £4·99

SOFTWARE FOR INSTANT ENJOYMENT - Comprehensive music entry program, 'MUSIC MAKER', can handle multiple channels, (16k min.) - £7·50

P.&P. 65pence per order.

BBF engineering

Dept. N, 82 Buckingham Drive, Luton, Beds. LUTON 35930

NASCOM 2 GRAPHICS

The IO graphics board gives 384 * 224 addressable points

FEATURES:

*fully bit mapped from user RAM
*software controlled
*line and block draw s/w supplied
***mixed text and graphics**
*real time ASSEMBLER plotting
*BASIC real time with NAS SYS 3
 (BASIC plot-then-display with
 other monitors)
*display size variable
 (10½k reqd for full screen)
*professional double sided PCB
*built & fully tested
*comprehensive documentation and
 full instructions for simple
 installation

Price ... £55 + 15% VAT (post free)

send SAE for free data sheet

Available **now** direct from:-

io systems ltd.
6 LALEHAM AVENUE, LONDON NW7 3HL.
Tel. 01-959 0106

interface components

GEMINI 64K RAM BOARD

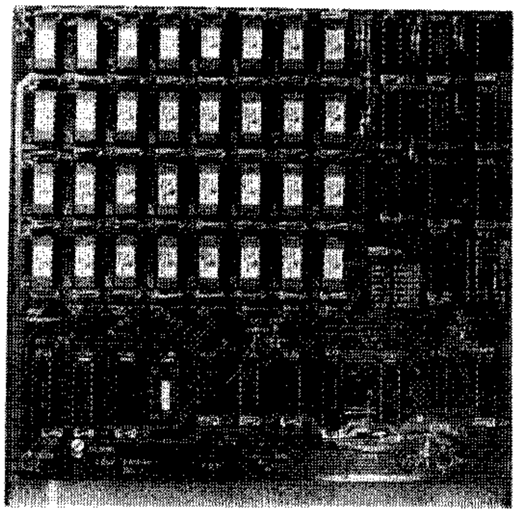
The Gemini G802 'RAM64' board is a newly developed Nasbus compatible RAM card that will accept up to 64K of dynamic RAM. The board utilises an active delay line in order to give full 4MHz operation. The board can also accomodate a page mode option enabling up to a total of four Gemini 'RAM64' and Gemini 'EPROM/ROM' cards to be used in the same system.

The Gemini 'RAM64' is available in four memory size options: 16, 32, 48 or 64K. A 20 pin header plug allows each 16K memory block to be located from any 4K boundary. Any option can easily be updated to the next at any time by the addition of 8-4116 memory devices.

'RAM64' is subject to special introductory pricing:

16K	90.00	32K	110.00
48K	130.00	64K	150.00

Add 2.00 P&P. Add VAT to all prices.



INTERFACE COMPONENTS LTD.
OAKFIELD CORNER, SYCAMORE ROAD, AMERSHAM, BUCKS HP6 6SU
TELEPHONE: 02403 22307. TELEX 837788