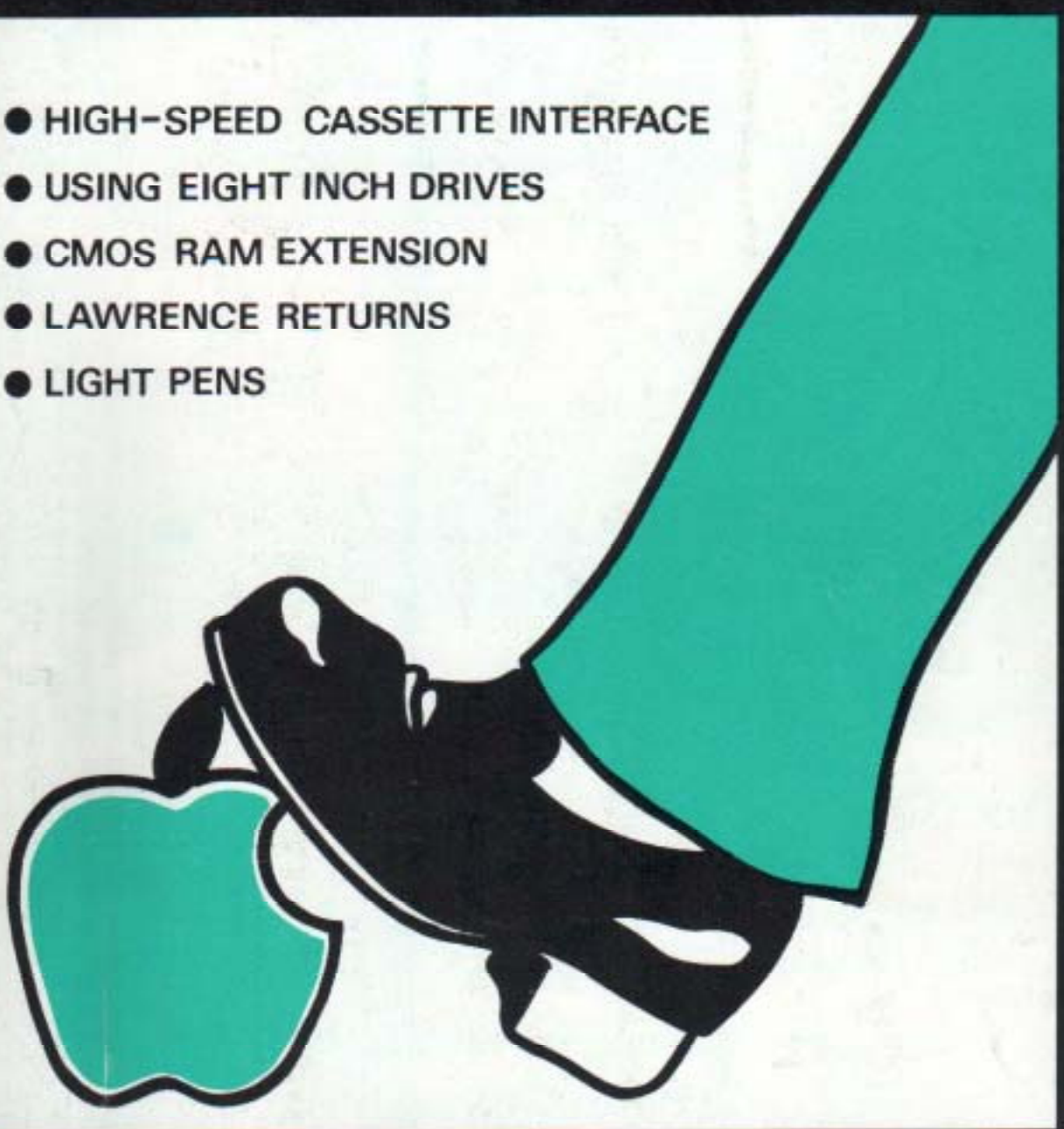
80-BUS NEWS

JULY–OCTOBER 1982 VOL.1 ISSUE 3

- HIGH-SPEED CASSETTE INTERFACE
- USING EIGHT INCH DRIVES
- CMOS RAM EXTENSION
- LAWRENCE RETURNS
- LIGHT PENS



£1.50

The Magazine for NASCOM & GEMINI USERS

JULY—OCTOBER 1982 **80-BUS NEWS** VOL.1 ISSUE 3

CONTENTS
========

SUBSCRIPTIONS

Annual Rates (6 issues)	UK	£9	Rest of World Surface	£12
	Europe	£12	Rest of World Air Mail	£20

Subscriptions to 'Subscriptions' at the address below.

EDITORIAL

Editor : Paul Greenhalgh Associate Editor : David Hunt

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

ADVERTISING

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

ADDRESS: 80-BUS News,
Interface Data Ltd.,
Oakfield Corner, Sycamore Road,
Amersham, Bucks, HP6 5EQ.

EDITORIAL

Dave Hunt has asked me to categorically state that the delay in producing this issue is NOT his fault [this time!!]. So I think I'll have to blame the current aspects of the planets, or something. Anyway, (he says, rapidly changing the subject) thanks once again for all articles that have been coming in. The response to the request for reviews of various new products has been good, but unfortunately most were received too late for this issue, so next issue will be a bumper review one.

The summer (Now, when was that? Ah yes, July 23rd.) has, as usual, been relatively quiet on the 'New Product' front. Now, as Winter draws in and the computers come out of hibernation (?), let's hope that we can look forward to lot's of nice (cheap) goodies, both hardware and software. As these come out we'll try to keep you informed, provided that we are kept in the picture ourselves (certain vendors please note). I also hope that in coming issues we'll be able to publish more programming tips and tricks, as the mag. has so far tended to be very hardware biased. So, if there are any budding Bill Gates/Paul Allen's out there, please drop us a few lines. (Note: BG & PA are the guy's who wrote a little BASIC interpreter, formed a company called Minisoft or Microsoft or something, and were/are multi-millionaires by their mid-twenties. (Not that I'm jealous or anything, honest.))

PCW Show Review

Many of you were probably not fortunate (?) enough to get to the recent PCW show at the Barbican (Barbarian?) Centre in London, so a few notes follow on what was there worth seeing.

The show this year was split into two sections in two halls, supposedly one section for 'hobbyists' and the other for 'more serious' users. In reality life is not so simple, and there were quite a number of companies that were either in the wrong hall, could really have been in either, or who hedged their bets (notably Acorn) and had stands in both!

Nascom were there with quite a large, but relatively empty stand. On show was a production version of the AVC running colour, another running 80x25, a Nas-Net slave (Nascom 3), and a Nas-Net master (N3 with disks). IO Research were there with a somewhat narrower, but fuller stand, showing off 'Pluto' on a number of different machines including Apple, IBM, ACT and Gemini Galaxy. Also on show for the first time (in prototype form) was 'Pluto Pallette', an expansion board for 'Pluto' containing umpteen more Kbytes of RAM to give lots of extra colours.

In contrast to the above 'tidy' stands, was the 'MicroValue' one. Quite a lot on show, but not enough space for it! The Quantum was making its first public appearance since production began, and there were two there, one standard and one running 'Pluto'. Then there were a number of Gemini Galaxy 1s (Galaxies?), one boasting an add-on 6MB Winchester unit, one with a prototype Climax colour card (looks very good, uses the Thomson 9365(?) chip, available Novemberish, about £200), two joined together via EV Computing IEEE488 cards (and one of those also sporting the EV printer spooler unit), one with the EV Computing RTC and bleeper boards (both add onto the IVC), one not in a case (i.e. a standard MultiBoard system) with the new GM825 disk drive unit (2 x Micropolis 5.25" + PSU), and a number of the afore-mentioned now sporting the new Gemini keyboard with fully programmable function keys and numeric pad. (Phew!) To add to the chaos MicroValue were also running a number of 'Special Offers' on Nas/Gem compatible products and showing a couple of machines that I won't mention here.

I believe Ikon were there showing their digital cassette system for the Nascom, and Vero with their prototyping cards, frames, etc, but I didn't get to either of these stands unfortunately. In addition there were numerous companies with printers, monitors, media etc. All in all quite interesting.

Next issue we should have reviews on a sound board, CMOS RAM board, digital cassette system, CPU board, plus more from Doctor Dark and no doubt another book from Dave Hunt!! Watch out for it.

LETTERS TO THE EDITOR

Real Time Clock Mods.

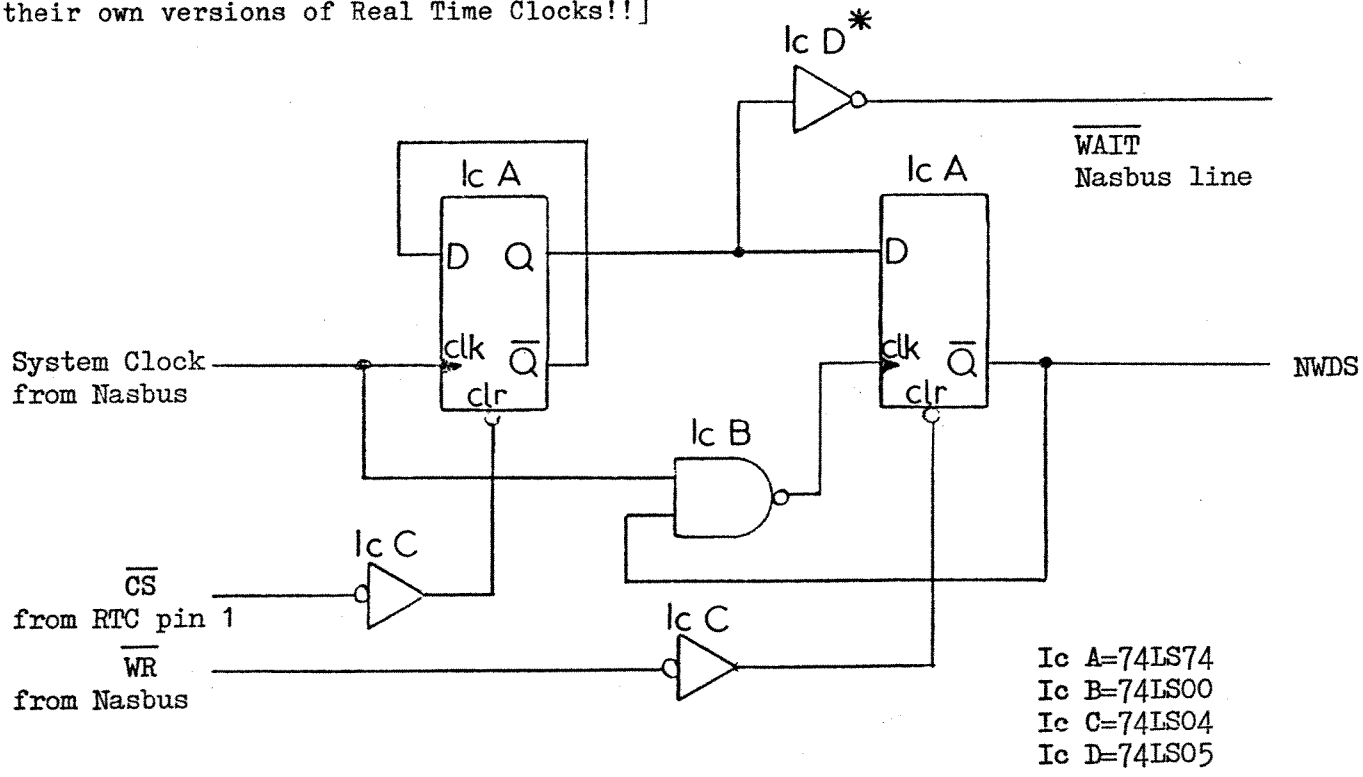
I was interested in the article by J.R. Williams in Vol. 1 Iss. 1 using the National Real-Time Clock (RTC) chip. I have been perusing this device for similar reasons, and the following comments may be of interest.

The RTC requires a delay between chip select and the write pulse (NWDS) of at least 310nS., and a write pulse width of 430nS minimum. Although Mr. Williams' design is working at 2 MHz on a Nascom 1, it may not always because it does not provide the chip select delay. This would be worse on a Nascom 2 at 4 MHz.

I am not yet in a position to try the circuit shown below but I hope that the idea behind the modification may assist someone. Basically, the circuit forces one extra WAIT state, and creates a new NWDS at a later time than the Z80 WR. (The Z80 hardware manual fig. 4.0-3 can be used to analyse the circuit timing.)

P.R. Verity, Addlestone, Surrey.

[Ed. - It looks like this is the year of the RTC! Not only is there the article in 80-BUS Iss. 1, plus the Gemini GM816 I/O board with RTC and Gemini GM822 PIO driven RTC, but now I hear that there are two or three other manufacturers about to launch their own versions of Real Time Clocks!!]



This replaces the direct connection between WR and NWDS in the original diagram.

A Club and Other Things

Thank you for issue 1 of 80-BUS News which continues in the vein of INMC News and INMC80 News, maintaining the highest standards.

Firstly may I make 2 announcements to fellow readers:

1. York Micro computer Club now meets at:
   Guppy's Enterprise Club
   17 Nunnery Lane, York.
   Nascomers, Geminiers (?), 80-Bussers (?) most welcome.

2. Weightman & Bullen Architects
23 Front St.,
Acomb, York
Tel 0904 792023
would be grateful to hear from other Nascom owners who are using them for Architectural purposes in accounting or Design. Currently we have a Nascom 2, 48K, Nas-Sys 1.

Secondly, in response to Mr. O'Farrells book reviews in 80-BUS News Iss. 1:-

Software Tools in Pascal.

Not having read the original I cannot make comparisons, but I recommend it as being very readable and well illustrated with examples of Programming and Documentation!!!! The cost is quite high (£8.95) but is very good value though the example programs will need rewriting to overcome User definable type limitations on the current Nascom Pascal. Please, please, BLS give us user definable types!

Since I am currently reading 'Computing Science' at Imperial College as a degree may I recommend the following books to the interested reader:

Thomas Bartee Digital Computer Fundamentals McGraw Hill £5.95
Thick, cheap, informative, lots of diagrams and examples of "Real" machines.

Martin Cripps Introduction to Computer Hardware Arnold £4.75
A less detailed view than Bartee but very readable and a good insight into "How it Works"

Further to comments on "Algorithms & Data Structures" by N Worth Prentice Hall. Gemini's disk version of Comal-80 contains a Comal version of the Pascal PL/0 compiler described in the book.

Thirdly. A Virtual Nascom??
Is this possible/feasible??

1. Tape routines for Nascoms/Geminis to read each others formats. Both use 1200 baud CUTS, but as yet I don't know how the RP/M routine works.

2. More Important. Video RAM Interrupt board. Is it possible for a board (ie 80-BUS) to be built that detects when a certain portion of memory is being read and then interrupts the host processor. This should really be user definable in 4K blocks a la N2 restart jump. Port addressing could be used to enable/disable this feature. For true N2 emulation the board must produce interrupts for memory writes between 0800-0BFF.

With this board in a Gemini type system, Nascoms with video RAM could be emulated fully enabling ZEAP, Naspen, Nas-Dis, Space Invaders, Pirahna etc. to be used and give full compatibility between both computers.

C'mon hardware hackers, someone design the circuitry and I'll write the software if it proves feasible.

Rupert Brown, Nasfan, Nascomaniac and Gemini owner, York.

[Ed. - Did you not see the routine published by R. Beal in INMC80? This allows the Gemini IVC to be used in a pseudo memory-mapped manner. Plus, I hear rumour (from Holland) that there is a program going around that allows most Nas-Sys programs to be run under RP/M and use the IVC for output, even when the output is done directly to a Nascom screen RAM address!]

Reading ZEAP Files to Disk

Quite a number of Nascom owners must have by now purchased the Gemini CP/M disk system, and no doubt many will want to transfer their ZEAP source files onto disk. The "Disk Basic Constructor" disk has software on it for doing this with tape files produced with the Nas-Sys "W" command, but there appears to be no easy way of doing this with ZEAP source files or Hisoft Pascal source files.

In fact, it is quite simple to read ZEAP tapes via the UART, using the command STAT CON:=BAT: which inputs data from the UART instead of from the keyboard. The problem is that as soon as the above command is executed the keyboard goes dead and you cannot now call up the editor into which the tape is to be read. From now on all commands and input have to go in via the UART. To call up the editor via the UART, you either record the ASCII command "ED filename.xxx" on the beginning of your ZEAP tape (followed by a CR, a big gap for ED to be loaded and I CR), or you have to use a second Nascom and join their UARTs together. Then the above command would be typed on the second Nascom and when ED is running the cassette deck is switched on.

The problem with the first method is that I cannot find an easy way of recording direct ASCII onto a tape, and the second one requires another computer. If it were possible to change the IOBYTE from within ED the problem would be solved.

When your ZEAP file is finally on disk you will want to remove the line numbers; most assemblers do not have numbered lines in their source files. The ED macro command Om5dl will do this by removing the first 5 characters from each line.

My Hisoft assembler accepts ZEAP source files (without line numbers) without any editing being required at all. Incidentally, it assembles in just 3 minutes a file on which ZEAP would spend some 22 minutes.

The reason for using the rather confusing and cumbersome ED is because it works via the BDOS and can therefore be directed via the IOBYTE to look at the UART. It also has some very useful macro commands.

Finally, does anyone know of a book which explains in detail the internal operation of CP/M?

P. Holy, Worthing.

Space Invader & Keyboard Mods.

I enclose information to upgrade the excellent game of Space Invaders as published in INMC80 issue 2.

I am sure most readers with Nascoms have this on tape and the modifications give a more pleasing effect, all that is required is NAS-GRA. Should readers like to put in their own solutions for the graphics then ignore the column marked 'Suggested graphic'.

The Gun

Hex location	Suggested graphic
1B72	= 8B (left character)
1B77	= AA (centre character)
1B7C	= 8A (right character)

The Invaders

Hex location	Suggested graphic
18CB=19E7=18AE=1D8B	= A6 (top row)
19EA = 18B3 = 1D8D	= B4 (next two rows)
19ED = 18BB = 1D8F	= BF (bottom two rows)

The Defence

Hex location	Suggested graphic
1B81 = 1D9D = 1DA5	= BB (main wall)
1D9E = 1DFE = 1DFF = 1DA5 = 1DA6	= BA (after one hit)

Space Invader

Hex location	Suggested graphic
1588 = 1594 = 15C1 = 1651	= B3 (left character)
158B = 15D0 = 1656	= B4 (centre character)
158E = 1598 = 15BE = 165B	= B3 (right character)

Changing all locations marked equal to each other for the same character is essential.

Reading 80-BUS News issue 1 regarding changing over the 'line feed' and 'newline' keys, I made an attempt to do this under software control, the following being the result. N.B. EPROM blower is essential.

Change Nas-Sys 1 location 05A8 to 89, 05AB to 0E, 05B5 to 09 and 05B9 to 8E and swap over the keytops for L.F. and Newline.

One problem, ESCape works OK in machine code routines, however to escape from BASIC (usually shift and newline twice) requires shift L.F. followed by shift newline. I cannot find the reason, I suspect something in the BASIC chip. Has anyone the answer? [Ed. - Yep, the BASIC ROM does shortened keyboard scans directly whilst RUNning programs, and is looking for the specific key depressions that correspond with the original shift and newline key positions.]

While in the Nas-Sys EPROM why not change locations 03BC to 03C4 (Nas-Sys 1 message) to your own, my own displays MIKES m/c, a very personal message. Location 0077, normally 5F, is the cursor blink character, changing to B6 will give you the little man.

Going back to messages, I have a second EPROM with software for the IVC from Gemini which I fitted at 0494 (load command for paper tapes). If you want a longer message than the 8 available at 03BC then put a jump to 0494 from 03BC and put your message from 0494. (The software at 0494 is my own and not from Gemini.)

Lastly, still in Nas-Sys, I changed location 07BE to 00B0 which gives me my Bits and PCs toolkit by simply typing D on the keyboard and newline.

Mike Trim, Ashton in Makerfield.

MONITOR, Naspen & Reset

As a relatively new convert to the 80-BUS and the associated products I have a lot to learn and I am greatly indebted to your magazine and its two predecessors for their assistance in my learning process. May I take this opportunity of wishing the new title a long and successful publication run.

My learning curve has now extended far enough that I am in a position to make some rather hesitant contributions to the store of knowledge that I have so far drawn on so heavily.

MONITOR.COM AGAIN

I rather regret that I did not receive my back copies of MICROPOWER, that other magazine, including Chris Blackmore's article on MONITOR.COM until after I had devised my own solution to the problem. However, your readers may be interested in a different approach. I dis-assembled Nas-Sys 3 using NAS-DIS, identified the references to VRAM and the work space from the published listing, and then re-compiled the resulting source code with a revised origin at 100H, VRAM set to F800H and the work space unchanged at 0C00H. Locations 100 - 102H were changed to a jump to new code at 900H - 0BFFH which writes appropriate jumps into the Restart locations, adds support for an EPSON Printer and adds disk routines. One advantage

is that you do end up with a machine readable version of the source code but if anyone else follows this path I would strongly advise them to spend more time than I did on identifying the DATA areas before dis-assembling!

For my particular purposes this is a better solution because, although relocated, all parts of Nas-Sys are still in the same relationship to each other. I am using a CP/M system to develop software that will then run on 'standard' NASCOMs, (without disk) which are used to drive other equipment and if I write the software with all absolute references to parts of Nas-Sys (such as revised command tables) of the form "Location + Offset", I can initialize "Offset" to 0 or 100H and produce two versions of the same program very easily. VRAM must be modified also, of course.

For the Disk routines I use the "F" command to switch between tape and disk and to set up the filenames. "R" & "W" then Read and Write from the designated devices. "D" is used to Delete files from disk. One facility that would be nice would be the ability to obtain a directory listing from within Nas-Sys. It galls me to have to write the code myself when the 'built in' CP/M commands are loitering in the top of memory. Can anyone tell me whether it is possible to call the built-in routines (DIR & REN in particular) as subroutines?

NASPEN

Following the advice given in Dr Dark's diary I have now converted my EPROM Naspen to a RAM version. However, the end result irritated me and I have made two other alterations that may be of interest to other readers.

Having the printer routines called via the Nas-Sys UOUT routine is a necessary evil when the program is in ROM, but a RAM version can be easily modified so that the printer is directly accessed when the P command is used.

The required changes are:

Existing			Replace with
B859 DF)		C3	JP ppqq
B85A 6E)	SCAL UOUT	qq	
B85B C9	RET	pp	

where ppqq is the start address of a printer subroutine.

The next problem is that the N command (return to Nas-Sys) operates by Executing from location 0000. Unfortunately that location is now the CP/M warm start and so the command does not have the required effect. A simple call to MRET doesn't work because Naspen has modified the Nas-Sys work space and the latter looks silly with the Upper and Lower case convention reversed! I have added the following code to the end of the program where the original authors have conveniently left 5 spare bytes:

```
BFFB  CD  )
BFFC  0D  )   CALL STMON
BFFD  00  )
BFFE  DF  )   SCAL MRET
BFFF  5B  )
```

(If you don't understand this you haven't read your Nas-Sys manual)

This new segment is called by modifying the execution address of the N command in the Naspen command table as follows:

Existing	Replace with
B99A 00	FB) the start address of
B99B 00	BF) the new segment above

The N command should now work correctly but I am afraid that it still prints that damned n! The warm start back into Naspen is still achieved by EB806.

RESET SWITCHES

I would like to thank Dusty Pulver for all his CP/M notes which are of tremendous value to a CP/M novice like me, but just in case no one else has pointed it out, his reset switch 1 (which resets Nas-Sys) will not give a CP/M warm boot because the switch is activating the NASCOM reset line and the NASCOM at that point clears all the Port set up for the disk drive controller card! He will just have to be content with EO enter from MONITOR.COM.

Robert Hill, Hemel Hempstead

RANDOMISE FOR NASCOM ROM BASIC

One of the niggling omissions from the Nascom ROM BASIC is a RANDOMISE function, to make the RND function produce unpredictable numbers. This USR callable subroutine should help a bit.

```
LD  A,R          ED 5F
LD  B,A          47
XOR A            AF
LD  HL,(EOOD)    2A OD EO
JP  (HL)         E9
```

This can be put anywhere in unused memory by a series of DOKEs, such as:

```
50 FOR I=0 TO3
60 READ A: DOKE 3360 + 2*I,A
70 NEXT I
80 DATA 24557,-20665,3370,-5664
90 DOKE 4100,3360
```

It uses the neglected Z80 instruction which reads the refresh register. If the computer has been waiting for an input from an unpredictable source such as the user, this will provide a number in the range 0 to 127. This can be used to send the RND function as in:

```
100 A = RND(-USR (0))
```

This will start one of 127 different sequences of random numbers, which may still be too predictable for some purposes. In this case, a subroutine to step through the sequence would be better. This can be called after each INPUT line if necessary:

```
50000 FOR I=0 TO USR(0):A=RND(1):NEXT:RETURN
```

Peter Dishart, Wembley.

ERROR

Please publish the following erratum to my article "INPUTting and READing Double Precision Constants" in 80-BUS News Vol. 1, Issue 2:

Line 2230 of the routine published should read:

```
LH = VAL(RIGHT$(A$,K-7))/10^(K-7)
```

My thanks to Adrian Sauter for pointing this out to me. Mike York.

Doctor Dark's Diary – Episode 12.

This article comes to you on time, and with far less than my normal quota of spelling mistakes, because I have run out of home brewed beer. And the last episode [Ed. - And this!] was printed with a very ordinary heading, instead of the usual fancy text. So, I'm in a meeeean mood...

Matters Arising.

First of all, a plea for help. Dave or other radio owner, does your Cardboard Box radio ever crash Nascoms that are nearby? I think my neighbours may be responsible for the crashing of the system that happened to me just now. And me in the throes of an urgent episode, on account of the magazine coming out so fast these days.....

Secondly, what nice things people are saying about MONITOR.COM, the CP/M program that breaks all the CP/M rules, but lets you use your old programs. (Somewhere out there, I bet, is a guy who has written a NASBUGT2.COM for his really old programs...) Don't forget that if you send a disk and a quid, I will send you a copy, and save you all that tedious typing. As some of you know, I also tend to copy silly programs of my own onto your disks, for your entertainment or something. If your disk has interesting programs on it that you wrote, I may even send the quid back with the disk!

And thirdly, it was nice to see that the Hisoft-Ringdale Engineering situation is not an on-going one. I hope I wasn't too sarcastic in my comments, as it was really none of my business anyway. I wonder if my beer is ready yet?

Syrtis Software Big Adventure review.

Adventure is a thing we have probably all heard of, and thought that it needed an enormous mainframe to run, and disks to store all its text on. I have certainly read quite a few articles about Adventure programs in the past, and it has always sounded as though it would need a humongous computer to handle it. In fact, the possibility of running programs like that in the home is one of the reasons why my system is constantly expanding. The adverts for the latest Adventure program from Bridgwater (not spelled correctly in Computnig Toady's version of the ad!) software firm Syrtis tempted me greatly, and in the end I sent off my twenty quid. A week later, or thereabouts, and the cassette arrived, and some documentation. The latter warned that the 1200 baud side of the tape might not load, and indeed it did not. Fortunately, the other side of the tape is at 300 baud, and as soon as I had fitted the speed change switch I had meant to add for about a year, I loaded the program, using MONITOR.COM, of course. Actually, the word "soon" is not valid in the context of loading 32K at 300 baud! The reason the 1200 baud version won't load is simple. It is recorded about 20 dB too quiet, according to the VU meters on my tape deck. (Syrtis, please note, for the sake of the poor devils who can't load it onto disk!)

"Enough of the flannel!" cries Bored of Accrington. "Is it any good?" You can't blame him. Now bearing in mind that I have no experience (well almost none) of using mainframes, I am glad to be able to tell you that I am very impressed with the program. It really is worth the money, Bored of A., and why do you have such a silly handle? (Picked that one up from one of them Citizens' Bondage chaps.) The game is very difficult, otherwise I would have cracked it by now, of course. Bad things can happen, and it is a relief to the beginner to find that the situation can be stored on tape, in case you get killed. I am not going to tell you what sort of things can go wrong, because with games of this kind, a lot of the fun is in having them happen to you just when you thought you were winning. In the same way, it would be a bad idea for me to say, for example, that you can get past the chasm by typing "leap",

because that would ruin your enjoyment of the success you will achieve if you do that... oh, what a giveaway! The cave system seems to be huge, and the messages on the screen contain considerable variety and humour. They don't always appear in the same form in a given situation, and are full of interesting insights into the mind of the author. There are very few spelling mistakes in the text, but there are more than I expected to find. Three weeks after starting to play the game, I am still getting messages I have not seen before, every so often. The text compression method must be a beauty, since I am sure I have read more than 32K of it! The use of text compression also has the advantage that you can not just copy the program into the screen memory to get clues as to how to behave in order to get past some of the more difficult obstacles. I am really looking forward to the 700K disk version! (Want any help with writing it, Syrtis?) Owners of the earlier 16K version can get an upgrade for a lower price, and it is well worth getting, chaps. It is a shame more people don't work in this way, to name but Digital Research...

I refer to these as "thoughts".

If you send a disk with 340K of data on it by first class post, which takes under 24 hours, you are achieving a data transfer rate of at least 322 Baud. (I wish I had a pound for every time I have seen that word printed as Band.)

Here is a nice idea, if you are short of ideas for what program to write next. Write a program that will read in a completed program from disk, and produce the specification for that particular program. This would be of considerable usefulness to all the people who hate the chore of writing a specification before getting on with the exciting program design and coding. The program could be tested by feeding it a copy of itself, and seeing if it produced this paragraph, in much the same way that Pascal compilers are supposed to be able to compile themselves...

When you have done that, you could go at least one better by writing a program which, when fed an article like this one, would use a dictionary of synonyms and circumlocutions to pad the article out so that it would earn more. Any suggestion that I may already be in possession of such a program will be treated as being flattery...

Short Hisoft Pascal 4 review, with free Procedure.

As with the Adventure review, you are all going to have heard that this program has finally appeared long ago. (I think he means you heard long ago that it had appeared! Syntax Checking Editor.) Hisoft Pascal 3 was for use with Nas-Sys, and highly recommended by just about everybody. This latest version of the compiler is for use with CP/M systems, and is probably very closely related to version 3. The manual lists the features of standard Pascal that are absent, as follows:-

(i) FILEs may only be of type CHAR,
(ii) a RECORD type may not have a VARIANT part, and
(iii) PROCEDUREs and FUNCTIONs are not valid as parameters.

Now it is possible that you find the preceding lines as close to double Dutch as I do, on account of not knowing Pascal. I am still in the process of learning the language, or trying to anyway, and am not sure whether the limitations mentioned above are at all severe. The manual makes it clear that it is not designed to teach you Pascal, and that you should buy a suitable book if you don't know the language. I see Rory O'Farrell says P.J. Brown's new one on this subject is good, and I have ordered it from Computing Toady's book service on these grounds. If I learn it fast, perhaps I can get in first with a series on how to write in Pascal; mind you, it looks as if I will have to type fearfully fast to beat Dave Hunt to it! Then again, some of us learned BASIC from the early Nascom BASIC manual (and a few have not yet recovered from this awful experience!) Some of my short experimental programs work quite

well, whereas others don't work at all even though they compile perfectly. It seems likely to me that Rory O'Farrell will write the definitive review of the compiler, and tell you all the things I should have, so I am going to change the subject, leaving you with a free gift in the form of a useful Pascal procedure that is all my own work. It is not particularly elegant, and is probably not the best way to do what it does, but it does have the great merit that it does work. It also proves that you can learn Pascal without having to have as many brains as Zaphod Beeblebrox...

```
PROCEDURE JUSTIFY;
 VAR
   PTRA, PTRB, I : INTEGER;
   LINEFULL : BOOLEAN;
 BEGIN
   LINEFULL := TRUE;
   PTRA := 1;
   PTRB := 1;
   REPEAT
     BEGIN
       {Test to see if there's more than a full line of text left.}
       FOR I:= 0 TO WIDTH DO
         IF REPLY[PTRB+I] = CHR(13) THEN LINEFULL := FALSE;
       IF LINEFULL
       THEN
         {Do this if there is a full line.}
         BEGIN
           {Look one line ahead.}
           PTRA := PTRB + WIDTH;
           {If not at a space, come back a bit.}
           WHILE REPLY[PTRA] <> CHR(32) DO
             PTRA := PTRA - 1;
           {Now print what is between the pointers.}
           FOR I := PTRB TO PTRA - 1 DO
             WRITE(REPLY[I]);
           {Print a CR LF if necessary.}
           IF PTRA < PTRB + WIDTH
           THEN
             BEGIN
               WRITE(CHR(13));
               WRITE(CHR(10))
             END;
           {Set pointers to do next line.}
           PTRA := PTRA + 1;
           PTRB := PTRA
         END
       ELSE
         {Do this if there is not a full line.}
         BEGIN
           {Find the end of the text.}
           WHILE REPLY[PTRA] <> CHR(13) DO
             PTRA := PTRA + 1;
           {Print the remaining text.}
           FOR I := PTRB TO PTRA - 1 DO
             WRITE(REPLY[I]);
           {Print a CR LF.}
           WRITE(CHR(13));
           WRITE(CHR(10))
         END;
     END
   UNTIL NOT LINEFULL
END;
```

What the procedure does is print out the contents of the array REPLY. in such a way that no words are broken at the edge of the screen. The following declarations must be made either globally or in a higher level block.

```
CONST
  WIDTH = 48; {Or 80, if you have an IVC.}
VAR
  REPLY : ARRAY [1..number] OF CHAR;
```

The value of "number" must be at least "WIDTH" characters greater than the maximum possible length of the text in REPLY, otherwise a nasty run time error will appear. I could have prevented this from happening, but it would have made the procedure even longer. Finally, the part of REPLY that is not filled with text must be filled with chr(13)'s, i.e. carriage returns. This routine does, in fact, more or less the same thing as the OUTPUT routine I gave away in my last article. You can tell it is a popular thing to want to do because the Extended BASIC has the WRAP command to do the same thing...

My conclusion is that this Pascal is a "must" for the CP/M user, although I have no experience of any of its rivals (are there any?) [Ed. - Yes, COM-PAS.]

A few comments on Dave Hunt's last article.

No, it isn't going to be destructive criticism, as that would be pretty pointless. I have been enjoying the series, and also marvelling at the speed he must be able to type at!

One of the things Dave says is that he is not a believer in flow charts. Not much further on in the article, is something that I and the lads at D.J.A.I. Systems would both call a flowchart. That highly famous program, The Last One, uses flow charts that look just like the summary of the program flow shown on page 26. I saw a demonstration of The Last One when I went to apply for a job there, and it is a really splendid program. Mind you, they did not give me a job, so they can't be all that clever!

The idea of standardising input and output routines is splendid, and is one that I had considered trying to do for this column. What went wrong? Well, I found I couldn't remember enough about Nas-Sys to produce the Nas-Sys routines to simulate CP/M's more useful routines exactly enough to give complete standardisation. Or maybe I never really got used to Nas-Sys before I upgraded Marvin again. Dave has approached the idea in what might be a less ambitious way, but it does work...

One of the things he has done that is much more naughty than writing self modifying code is to use a direct call to B2HEX. Surely, that should have been £DF £68 instead of £CD £F2 £0E? There are more of these in there, now that I look again. Oh, now I see why! For the benefit of the CP/M machines, of course. For a moment, I thought I had found a real naughty... I will write out a hundred times, "I must read all of the documentation."

There is a small point concerning the part of the program that reads:

```
LD A (TRYS)
INC A
DAA
LD (TRYS) A
```

The values of TRYS in the program are such that there will be no problems with this bit of code, but it is important to note that INC A does not affect the carry flag, which will be in the state produced by the last opcode that affected it. This can result in DAA appearing to have made a mistake, and formed the basis of a competition in a prehistoric precursor of this magazine. If you use ADD A 1, you will not get this problem, although you will use one extra byte of RAM. (Obviously, if you are doing BCD arithmetic on multi-byte numbers, you may use ADC A 1.) I think I really caught you this time, Dave!

More matters arising.

While I am still in self congratulatory mood, I wonder why Jeremy Gugenheim uses DDT to alter MONITOR.COM? It is perfectly capable of amending itself and putting the new version on the disk for you! Use the M command for the modifications, and then P0100 0A00 to invoke the disk save routine. The only decent substitutes for DDT.COM are ZSID.COM (not very nice) and Hisoft's MON.COM (excellent!) [Ed. - What about GEMDEBUG.COM?]

When changing ZEAP, you will get error 90 messages until you correct the checksum, at whatever location it is at in your version of ZEAP. I helped a user of ZEAP to do this once by writing a short program that added all of ZEAP into the accumulator, and kept incrementing the checksum byte until the result of the mighty addition sum came to zero. This worked rather well, and is worth a try, if you know where the checksum is in the version you have. The sooner that is fixed, the sooner we are likely to see a disk input/output fix for Naspen, which is something I could do with...

Persecution of the consumer - shock horror probe!

I have been a member of the Computer Book Club since it started, and at first all those stories you hear about book clubs seemed to belong to folklore.

Everything worked well for some time, although I was a bit put out when it took them two months to replace a book that the Post Office had folded (a hardback!). Then one month, I ordered a book they had run out of. Their computer sent me an invoice, which was amended in manuscript by their accounts department, telling me I didn't have to pay for the book. Fine, so I didn't pay for the book they didn't send. Next month they invoiced me for the same book again. I phoned them and was given a stock answer about it being a "computer error", and was told that everything would be OK. The next month, the latest book I had ordered did not arrive. Instead I got a very unpleasant letter telling me my account had been "frozen" until I paid up the (imaginary) debt. When I rang up they said of course it was a mistake, and the fault would be put right straight away. It was all the fault of their computer programmers, I was told, and was easily put right. I waited and waited. No book. So I wrote to them. And waited, and waited. They had not replied after six weeks, so I decided to write this, and send letters about them to all the other computer magazines as well. The very day I had finished the first version of this article, the last book I ordered arrived, so this is version two of this paragraph. Had they bothered to write and apologise, or explain what was going on, I might have edited out the whole paragraph. But they have not done so, and that makes me feel that you might like to know what happened when they blundered.

I hope I haven't bored you too much, going on about them like this, but I feel that it is vital to make a fuss when this sort of thing happens, or they will just keep treating people in the same offhand way.

And finally....

The mighty lists of computer clubs in Personal Computer World never mention this club. Rather than moan about this, I am going to write and tell them we exist. Let's all write and tell them! (Did I say I work for the Post Office?)

END. {A sort of Pascal joke, that!}

LIGHT PENS

by D. Parkinson

What is a light pen?

A light pen is essentially a photo-detector held in a pen-shaped carrier that is used in conjunction with a video display. When the light pen is held against the screen of a video display the photo-detector produces an output whenever it is over an illuminated part of the screen. A conventional video display uses a raster-scan, where the apparantly steady display is in fact being re-written at 50 frames per second. Every time the section of screen under the photo-detector is re-written a pulse is output from the light pen. If this pulse is routed to the display generation circuits of the system the position of the light pen on the screen can be determined from the current values held in the video display generation circuits.

In order for the system to work the screen used should not have a long- persistence phosphor, the photo-detector should have a fast response, and the light pen should be adequately shielded against unwanted pick-up from the deflection coils of the video display.

It is basically a poor man's Bit-Pad, costing in the 10s of £s rather than the £1000s. (A Bit-Pad is a tablet and stylus combination, where the computer can sense the position of the stylus on the tablet. These can offer large "draw" areas with a resolution of 0.004" or better).

What is a light pen for?

Good question, I offer two stock answers:-

a) It's use is limited only by your imagination. (i.e. I haven't a clue - have you?)

or b) Menu driven programs.

With a menu driven program the user is presented with a "Menu" of options on the screen. To select a given option all he has to do is to place the light pen against the appropriate option on the screen and press the button. This generally will bring up another sub-menu, and the process can be repeated until the final goal is reached. An example of such a program would be an information retrieval program for the inexperienced user (or inept typist!).

Such an approach can also be used to help the casual user run a complex analysis program.

Light Pens and Nascom/Gemini owners

There are a variety of commercial light pens available, and they are not beyond the capability of the home constructor. The Gemini IVC will support a light pen directly, but those intending to add one to a Nascom will need to exercise some ingenuity as there is no in-built support. (Possibly cross connect the outputs of the counters that address the screen ram to the inputs of a PIO??). I have only used the Arfon light pen with the Gemini IVC, but it should be possible to interface one of the other commercial pens to it. (Necessary now as, alas, Arfon is no longer with us).

The Arfon light pen (If you can still get it)

The Arfon light pen is housed in a black anodised aluminium cylinder about 18cm long and 2cm in diameter. It's not heavy, weighing in at about 2.5 Ounces (my wife's scales aren't metric!), and is supplied with about 130cms of lead terminated in a 5-pin DIN plug. This DIN plug will plug directly into the Gemini GM812 IVC card. I believe there is an interface available for a Nascom, but I have not seen this and cannot comment on it. There is a push button mounted in the pen body close to the end that carries the photo-diode. This push button falls conveniently under the fore-finger when the pen is held in a 'pen-like' manner.

The output of the photo-diode is amplified and limited to produce a TTL compatible STROBE signal. The switch is debounced and produces a TTL compatible ENABLE signal.

Using the light pen

I have used the light pen in conjunction with the Gemini IVC and shall thus restrict my discussions to this particular combination.

The 6845 CRT controller used on the IVC directly supports a light pen. The STROBE output of the light pen is connected through to the LPEN input on the 6845. As supplied the light pen provides a negative going strobe pulse. The CRT controller uses the rising edge of this strobe as a trigger for latching the screen address, and so all coordinates returned will be found to be offset by the strobe pulse width. Purists may like to dismantle the light pen and select the opposite polarity for the strobe, but this is of little consequence as it can easily be catered for in software. Note, however, that if the strobe is left as it is the offset required will be dependent on the light pen used, and also on the screen width in use. (A strobe width of 6us is 12 characters in the 80-wide mode, and nearly half that in the 48-wide mode).

The IVC software - ESC P

The video card supports the ESC P command for returning the coordinates of the light pen. At the time the IVC software was written no light pen was available to try with it, and so the function only performs the simple operations as follows:-

i) Wait until the light pen switch is found to be operated.
ii) Wait one TV frame.
iii) Read the light pen register in the CRT controller.
iv) Adjust the address to account for IVC hardware delays.
v) Test for a valid screen address.
vi) If the address is not valid then go to i).
vii) Convert the address to equivalent Row/Column coordinates.
viii) Return the coordinates to the host system.

This performs the basic function required by a controlling program, but to get the best out of it a little more support is required. This is illustrated in the following sections using the ESC L and ESC U functions of the IVC. (See IVC manual). One small point to take into account with the function is that if the switch on the pen is held down the IVC will immediately return a set of coordinates whenever the ESC P sequence is received. ie The function is level sensitive, not edge triggered. The controlling software will have to decide if the coordinate pair represent a new selection, or if the operator has still not yet released the switch from his last selection. (See example 4 for a way round this).

The IVC and high level languages (eg BASIC)

Commands can easily be sent to the IVC via the usual PRINT or WRITE state- ments, but the difficulty arises in getting any information back. For someone using the IVC on a Nascom 1 or 2 with the Nascom Basic and one of the Nascom monitors the solution is relatively easy. The ESC P command can be sent by a PRINT statement, and the WAIT and INP commands can be used to read the coord- inates back. (See example 1).

With the Gemini Multiboard system or the GM809 disk system (with CP/M 2.2) the situation is more complicated. This is because the IVC card is occasionally being interrogated by the system software to check if there is any character waiting from the keyboard. (This check occurs whether or not a keyboard is actually connected to the IVC). Because of this it may not be possible to use the method of example 1 to get the light pen coordinates, and a machine code subroutine has to be used. Such a subroutine is listed in example 2. This is slightly more long-winded than necessary, but it has been written to be position independent so that it may be copied directly, and no recoding is necessary wherever you decide to place it. This program is included in the Basic routines that follow within the DATA statements.

EXAMPLE 1 - NASCOM BASIC

This Basic program is for a Nascom 1 or 2 using the Nascom Basic and one of the Nascom monitors. It is assumed that the user has already entered a simple routine along the lines of PUTVID from the IVC manual in order to use the IVC in this environment. (The U command should have been used to activate the IVC driver which should also append LFs to CRs).

NOTE: The IVC ports OB1H and OB2H are 177 and 178 (decimal). The buffer full/empty bit is bit 7 which is equivalent to decimal 128 for use in the WAIT statement.

```
100 REM Demo program for the Arfon light pen
110 ROW=INP(177)              :REM Ensure IVC port cleared.
120 P$=CHR$(27)+"P"           :REM Set up <ESC> P command string
130 MAXCOL=80                 :REM Screen width
140 OFFSET=12                 :REM Correction required
150 GOSUB 2000                :REM Get the coordinates
160 PRINT "Row=";ROW; " Col=";COL
170 GOTO 150                  :REM Loop

2000 REM ************************************************
2001 REM  Subroutine to get a set of light pen coordinates
2002 REM ************************************************
2010 PRINT P$;                :REM Send the command string
2020 WAIT 178,128,128         :REM Wait for "READY" flag
2030 ROW=INP(177)             :REM Read back ROW coordinate
2040 WAIT 178,128,128         :REM Wait for COL
2050 COL=INP(177)             :REM Read back COL coordinate
2060 COL=COL-OFFSET           :REM Allow for offset
2070 IF COL>=0 THEN RETURN    :REM Return if all ok
2080 COL=COL+MAXCOL           :REM Not,so adjust COL..
2090 ROW=ROW-1                :REM...& ROW
2100 RETURN                   :REM return ROW & COL
```

Nascom Basic users should also look at the following examples which illustrate another way that the coordinates can be obtained.

EXAMPLE 2
MACHINE CODE SUBROUTINE TO GET COORDINATES

```
;       Subroutine to request and get the coordinates of
;       the light pen from the IVC.
;       This routine is for Microsoft's MBASIC

MAKINT  EQU     105H            ;Address of MAKINT routine
IVCST   EQU     0B2H            ;IVC status register
IVCDT   EQU     0B1H            ;IVC data register
ESCAPE  EQU     01BH            ;Ascii "ESC" code

;       This routine is invoked by the USR command and returns the
;       coordinates as an integer of the form:-
;         ROW*256 + COLUMN

LPEN:   IN      A,(IVCST)       ;See if IVC ready to rec.
        RRCA                    ;Flag to carry
        JR      C,LPEN          ;Loop if not
        LD      A,ESCAPE        ;Send "ESC"
        OUT     (IVCDT),A
LPENO:  IN      A,(IVCST)       ;Wait again
        RRCA
        JR      C,LPENO
        LD      A,"P"           ;Now send the "P"
        OUT     (IVCDT),A
;       Command sent, now wait for coordinates
LPEN1:  IN      A,(IVCST)       ;Wait for ROW
        RLCA
        JR      C,LPEN1
        IN      A,(IVCDT)       ;Read ROW
        LD      H,A             ;Put in H
LPEN2:  IN      A,(IVCST)       ;Wait for COL
        RLCA
        JR      C,LPEN2
        IN      A,(IVCDT)       ;Read COL
        LD      L,A             ;Put in L
        PUSH    HL              ;Save the value
        LD      HL,(MAKINT)     ;Get address of return routine
        EX      (SP),HL         ;Reset HL (MAKINT on stack)
        RET                     ;Return the value

        END
```

EXAMPLE 3 - MBASIC

This Basic program, which is well sprinkled with REM statements, is provided as a guide that others can follow. As it is purely for demonstration purposes the program has been coded for clarity.

Note: In line 1020 ADDR holds the address of where the USR routine is stored. Nascom Basic/Nas-Sys users should pick an area within the workspace RAM (0C80H-0FFFH). GM809 users on a Nascom can use the workspace RAM as well, though in their case it is located at 0FC00H-0FFFFH. (NB Your CP/M system size should be 63K or less if you intend to use the workspace RAM).

For Multiboard users the problem is slightly more complex. Either a CP/M system can be used that is 1K less than the available memory size, or the /M option can be used when loading MBASIC to leave an area of RAM free for the machine code program.

```
100 REM Demo program for light pen
110 GOSUB 1000              :REM Initialise the code
120 GOSUB 2000              :REM Get coordinates
130 PRINT "Row=";ROW;" Col=";COL
140 GOTO 120                :REM Loop
1000 REM ************************************************
1001 REM Subroutine to Initialise the USR function
1002 REM ************************************************
1003 REM
1010 ADDR=-128              :REM  Machine code Put here
1020 DEF USR0=ADDR          :REM  ***define USR0 function ***
1030 FOR I=ADDR TO ADDR+39  :REM POKE loop
1040   READ A
1050   POKE I,A             :REM Set byte
1060 NEXT I
1070 RETURN
1080 REM  ****Machine code subroutine ******
1090 DATA 219,178,15,56,251,62,27,211,177,219
1100 DATA 178,15,56,251,62,80,211,177,219,178
1110 DATA 7,56,251,219,177,103,219,178,7,56,251
1120 DATA 219,177,111,229,42,5,1,227,201
1130 REM
2000 REM ************************************************
2001 REM Subroutine to Read back coordinates from LPEN
2002 REM ************************************************
2003 REM
2005 MAXCOL=80              :REM Screen size
2010 OFFSET=12              :REM Character offset
2020 COL=USR0(ROW)          :REM Get coordinates
2030 ROW=INT(COL/256)       :REM Isolate ROW number
2040 COL=COL AND 255        :REM Isolate COL number
2050 COL=COL-OFFSET         :REM Allow for offset
2060 IF COL>=0 THEN RETURN  :REM return if all ok
2070 COL=COL+MAXCOL         :REM Not,so adjust COL
2080 ROW=ROW-1              :REM ..and correct ROW
2090 RETURN                 :REM return ROW and COL
```

Shown below is an example of how the subroutine 2000 can be altered to increase the accuracy of the returned value. Here line 2020 obtains four successive readings from the Light Pen. These are averaged in the next line, and then a check is made that all four lie close to the average value. (If you leave out the check and C1-C4 vary widely you may find strange values like Col=129 appearing!). The routine might be improved by including a small delay between the calls to USR0. Only two calls with a reasonable delay between them may be enough....experiment!

```
2000 REM ************************************************
2001 REM Subroutine to Read back coordinates from LPEN
2002 REM ************************************************
2003 REM
2005 MAXCOL=80              :REM Screen size
2010 OFFSET=12              :REM Character offset
2020 C1=USR0(A) : C2=USR0(A) : C3=USR0(A) : C4=USR0(A)
2021 COL=(C1+C2+C3+C4)/4    :REM Average four readings
2022 IF ABS(COL-C1)>2 OR ABS(COL-C2)>2 THEN GOTO 2020
2023 IF ABS(COL-C3)>2 OR ABS(COL-C4)>2 THEN GOTO 2020
2030 ROW=INT(COL/256)       :REM Isolate ROW number
2040 COL=COL AND 255        :REM Isolate COL number
2050 COL=COL-OFFSET         :REM Allow for offset
2060 IF COL>=0 THEN RETURN  :REM return if all ok
2070 COL=COL+MAXCOL         :REM Not,so adjust COL
2080 ROW=ROW-1              :REM ..and correct ROW
2090 RETURN                 :REM return ROW and COL
```

Note: Example 3 has used Microsoft's MBASIC in a CP/M environment. Other versions of Basic may be used in a similar way, but the syntax may need adjusting to suit. (eg The Nascom version of Basic supports only a single USR function which is accessed in a slightly different way). The same technique may be employed in other langauges, the exact approach to take depending upon the way the language executes. If there is a poll of the keyboard inbetween statements (as in the Basic interpreter), then the machine code subroutine approach will be necessary, otherwise the direct port access (WAIT/INP etc) can be used satisfactorily.

EXAMPLE 4 - "SINGLE SHOT" OPERATION

This example is in fact an extension to example 3, and uses the ESC L and ESC U commands to provide a "single shot" operation of the light pen switch. This is done by downloading a very short program to the IVC and calling it before requesting (or after reading) the light pen coordinates. What the routine does to sit in a loop until the switch on the light pen is released, and so the switch has to be released before another pair of coordinates can be read.

```
;       Subroutine for the USER area of the IVC card.
;       This subroutine does nothing other than sit in a loop
;       if the light pen switch is found to be operated.
;       The code will be loaded to the user area at 0E400H
;       but it is position independent anyway.

LPENSW  EQU     4               ;Bit number of LPEN switch on status port

USER:   EXX                     ;Get the alternate register set
LPENW:  BIT     LPENSW,(HL)     ;Is the switch operated?
        JR      Z,LPENW         ;Yes,loop
        EXX                     ;No, reset registers
        RET                     ;..and return

        END
```

```
3000 REM ******************************************************
3001 REM  Subroutine to set up a "User" command on the IVC
3002 REM ******************************************************
3010 U$=""                      :REM Clear U$
3010 FOR I=1 to 11              :REM 11 bytes to send
3020   READ B                   :REM Read a byte
3030   U$=U$+CHR$(B)            :REM Add to string
3050 NEXT I
3060 PRINT U$;"ESC U now set up" :REM Set command and say so
3070 RETURN
3080 REM   "ESC" "L" 7 0 +machine code routine
3090 DATA 27,76,7,0,217,203,102,40,252,217,201
```

Try adding the above subroutine to the code of example 3. Also include

```
     125 GOSUB 3000

    2015 PRINT CHR$(27);"U";    :REM Do ESC U. (ie Hang up until
                                       he/she lifts his/her finger).
```

These are only suggestions - take and develop as you see fit.

16K CMOS RAM extension for the Nascom 2 main pcb

by Paul Anderson

Reprinted from Nascom - Thames Valley User Group Newsletter No. 5 with permission

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

I will introduce you to an amazing little beasty called the HM6116-4 manufactured by Hitachi, available a year ago for the princely sum of £30.00 each - have I frightened you - but now available for about £4.00 each in 10 up quantities. Each of these devices is 2K x 8 bits (16K) so you will need eight chips, one for each socket!!!! This will cost about £40.00 inc VAT, sounds a lot, but when you consider that to update by that 16K you will have to sell your 48K and buy a new 64K (anyone seen any second hand 64K cards), plus the cost of the extra RAM, I suspect it will cost about this amount anyway.

As I said earlier the HM6116 is an amazing device - the fastest version will run at over 8MHz and the low power version consumes about 0.1uA (I do mean micro-amps) in standby mode. I used the standard version (cheapest) and out of a batch of 20 samples they all exceeded their quoted performance by an enormous margin, particularly in terms of their standby mode current consumption. For the standard version a standby current of 20uA is quoted, which for eight chips is 160uA, which would be Ok for battery backup. However, of the samples I have checked, the worst device had a standby current of only 5uA, only three were above 1uA and typical values were between 0.3 and 0.7uA (almost two orders of magnitude inside the spec.). So if we were to power them from one mempack battery (3.6V, 150mA/hours), we would be able to operate them for about 15,000 hours on one recharge, assuming a total consumption of about 10uA for the eight chips. That's approaching two years, in practice one year would be about right taking self discharge of the battery into account, and if we were to float charge the batteries at 1mA (milli-Amp) for four hours each week - I'm sure your Nascom is on for more than that time per week (if not, there's a little mod) - then the batteries will remain virtually fully charged for the life of your Nascom.

Now battery backed up RAM opens up many interesting possibilities, firstly it can be used like ordinary RAM, as workspace, to hold programs, for CP/M or whatever, secondly it can be used as loadable EPROM, and if a 'write disable' were incorporated the data in the RAM would be as safe as if it were in EPROM, but can still be changed virtually instantaneously when required. Other advantages of battery backup and write disable are when developing programs, where they can be safely tucked away whilst the program is tested, or you go away for a tea break and the wife decides to pull out the mains plug so she can use the Hoover. Or, if like me you are plagued by mains dropouts and have to save programs after virtually every modification just in case you loose the lot - very time consuming on tape, although I'm now disk based, it's still a nuisance. I am convinced that battery backed up RAM is preferable to normal RAM, and am now seriously considering fitting 64K of CMOS RAM. I noticed an advert the other day offering 32K of CMOS RAM on a card suitable for NASBUS/80BUS.

So after this extended introduction I will proceed to enlighten those interested parties on how it is done. It's based on using the eight EPROM/RAM sockets on the Nascom 2 pcb. (Anyone still using the EPROM sockets?? If so, my next article will deal with converting them into 2732 sockets, I've 64K of 'em on one card in my system.)

The eight sockets will accommodate most 'BYTE WIDE' 24 pin devices and the HM6116 falls into that category. For straight forward RAM expansion without write protect and battery backup, fitting is very similar to fitting MK4116's in these positions except that a higher order addressing is required to access all the RAM available. The only additional circuitry is to provide RAMDIS on to the Nascom 2 to disable this new RAM if you are using external RAM or ROM at the address chosen to locate your new RAM. It is suggested that this stage is done first anyway to check that RAM is operating correctly before introducing the complexities of power down, battery backup or write disable.

Refer to the link block diagram in your manual (you have got a manual, haven't you?) and on each link block, LKB1-8, connect pin 8 to pin 4 (chip select) - pin 6 to pin 2 (output enable to read enable) - pin 5 to pin 1 (write enable). Finally loop pin 7 (address line A10) along to the next pin 7 on the next link block, and so on, so that all eight pin 7's are connected together, and pick up address line A13 off the bus - best done on pin 12 of IC47 (N2MD). This has the affect of addressing through all the lower 1K's of each chip and then through the upper 1K's (the upper and lower halves of each chip are then separated by 8K, not a very clever idea for EPROM, but as it's RAM it doesn't matter), it's the simplest method. Finally, to make the whole thing go, connect the RAM-GATES on the 'MD' link plug to the addresses required, for instance D000 to F000 would give memory starting at D000 and finishing at FFFF. If you are using a CP/M type system, then set the memory card to go from 0000 to B000, giving memory from 0000 to BFFF, and strap the block addresses C000 + D000 + E/F000 together, to RG1 or RG2. This gives 64K from one end to the other.

To provide RAMDIS on to th pcb requires two ICs, one 74LS32 and one 74LS04. These devices will be used to provide the write disable control as well (so you might as well go the whole hog). In my system these two chips were piggy backed on to two other conveniently placed 14 pin chips close to Link Block LKS1, to which most of the connections will be made. Remove IC18 and bend out pin 13 then replace the IC. For the remainder of the connections refer to circuit diagram 1 along side the sheet 3 of the Nascom 2 circuit diagram. Look around the top left hand corner, that's where most of the circuitry will be found. I suggest that you now carefully test your Nascom with and without external peripherals either addressed, or not addressed to the same area as the new RAM. Make sure there is no interaction between the new RAM and the existing RAM, any faults should be found and corrected. My system has been operating now for many months and there have been no problems.

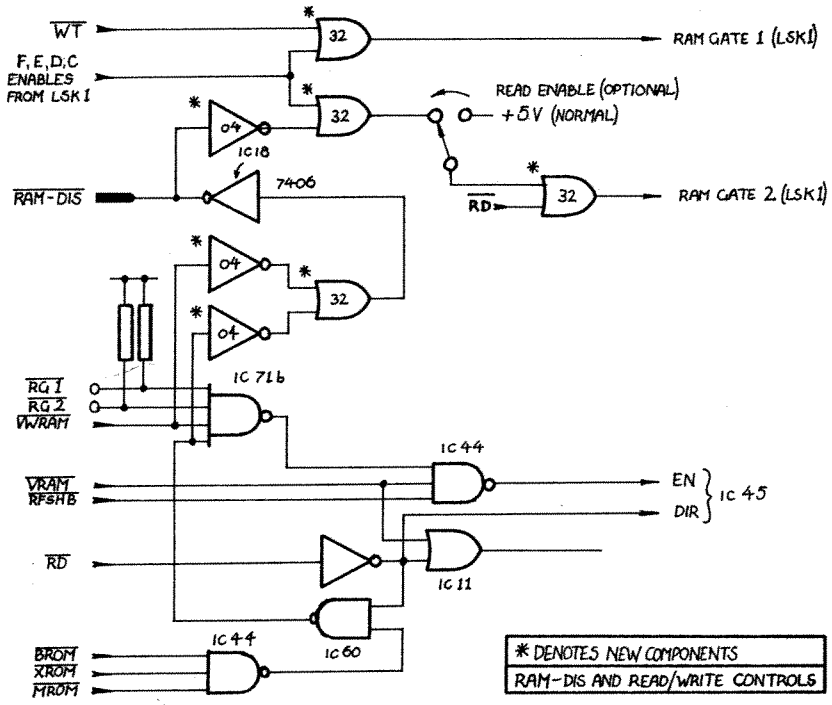
Now for the tricky bit, the battery backup connection, it's not difficult, just time consuming, and some care is necessary to ensure that the right connections have been made. The idea is to isolate all of the +5V connections to pin 24 of each chip and also to pin three of the associated link blocks. This can be done by cutting about five tracks and adding a few wire links. I would have included complete details, but it's very difficult to get at the underside of my N2 card now to see exactly what I did. Check that you still have +5V on your monitor and Basic ROMs, and any other RAM/ROMs in the vicinity. I didn't when I did it. Next, replace IC46 with a 74LS156 (the 74LS156 is an open collector version of the 74LS155 already fitted) as this is necessary to allow correct power down. Fit 8 off 1K resistors between pins 3 and 4 of each link block LKB1-8 these provide power to the device selects in the standby mode and should be connected to the new +5V backup supply when the battery is fitted.

Now for the battery backup, best to refer to diagram 2. The best place for the battery is by the 'power test points' adjacent to the Basic ROM. The test points provide a convenient point for connection to the battery (via the diode and resistor) and is conveniently close to the now separate RAM supply and device select rails. The 180R resistor in series with the battery is the charge limiting resistor and is chosen to fully charge the batteries on about four hours use a week. If the Nascom is used for a lesser amount of time over a weekly period, then the value of the resistor may be reduced. Note the value of the resistor must not be made less than 100R. Ensure that the battery is connected to the power supply and device selects ONLY. Any wrong connections and the battery will get quite hot and be flat in minutes.

It's worth considering connecting the monitor into the memory backup with another HM6116, with it's own write protect switch, as this allows the monitor to be changed at will yet still be there on power up. Further, if the Bits & Pcs programmable character generator is fitted, both 4116's can be replaced with a single HM6116 or even two HM6116's and the character generators made fully programmable as well.

The write protect switching is recommended as there is no protection against spurious 'writes' during power up or power down. However, to date I have not experienced any problems in this respect and have tried to provoke spurious errors without 'success'. Anyway, to be safe, always disable the write during power up/power down.

Having had this facility fitted for some while I wonder how I ever got on without it before it was fitted. The time spent fitting it and getting it right was well worth the effort.



CLASSIFIED ADS.

==========

NASCOM 1 with Nas-Sys 3, uncased, £50. 051-608-1796.

NASCOM 1 unexpanded using Nas-Sys 1 & 3A Power Supply & mounted in case. All in good working order. £99. Tel. 0603-613698 evenings.

NASCOM 2, 48K, a neatly built, cased system with extras, software and documentation. Excellent order. £325. 0935-850416.

NASCOM 2, 48K RAM, GEMINI 64K RAM & EPROM cards, ZEAP, NAS-DIS, DEBUG, Ext. BASIC, Teletext Colour, 4.8K tape, Verocase, enormous software & firmware library. Interested?
Phone Peter Smith, Petersfield (0730) 4059. W/E, Evenings.

EPROM PROGRAMMER. Bits & Pc's programmer for 2708s & 2716s. Fully assembled with application software, £20. Also 6 x 2708s, fully erased, £6.
Ring Keith Brown on Colchester (0206) 841293.

TELETYPE MODEL 33 page printer (110 baud ASCII), with manuals as new. £90 ono.
Tel. David Warwick, 0723-862160 (evenings).

HIGH SPEED CASSETTE INTERFACE

We have received the following letter and article from W. Van Malderen in Antwerp, which comes as a reply to our request for high speed interfaces. Far be it from us to ridicule our overseas friend's somewhat idiomatic use of English (and to that end parts of Mr. Van Malderen's article have been rewritten) but we reproduce his original letter here uncorrected, as in some strange way we feel it encapsulates the spirit of experimentally inclined Nascom 1 owners. DRH

B2000 Antwerp
Belgium

Dear Editor,

Until now I was rather a silent overseas member of the INMC, each time waiting until the next INMC News was published. Then when it arrived, it was again joy for lots of days. Sometimes the magazine came up with solutions for problems I had, sometimes it containt programs that could enjoy me, but for some also my children.

The only thing that stayed very annoying was the slowness of the cassette interface. First I speeded it up to 488 baud, thus doubling the standard baudrate (Nascom 1). Finally I found a promising circuit in a Dutch magazine. Then I thought I couldn't keep this circuit unknown for the fellow INMC members, and I started to write the additional article. I hope it will help on its turn someone else who struggles with these problems.

To conclude, I wish the INMC News to stay as excellent as it is.

Your sincerely,
W Van Malderen

PS. excuse for possible spelling faults.

Drive a Nascom 1 GTX, and other things

by W Van Malderen

Although the Nascom 1 is a very fine machine (and I have no experience with the Nascom 2), I find that the Nascom 1 needs something better than it's standard cassette interface to avoid frayed nerves during a Read, Write or Verify cycle. An article in the Dutch magazine 'Elektuur' [published in the UK as Elector. Ed.] of June 1980 brought the solution to this problem. I shall try to summarize the contents of that article.

Fast cassette interfaces

The most commonly used cassette interfaces employ some form of f.s.k. and Kansas City (CUTS) standards. The following cassette interface uses the rather more professional 'Manchester II' standard, which has the following advantages:

- Clock reclamation is fairly easy as it is a component of the recorded data
- The circuits are not particularly complex
- The redundancy (efficiency) is 50%

The redundancy figure being 50% means that the baud rate may be equal to the highest tone frequency used, the lowest frequency will be half the maximum baud rate and therefore half the maximum tone frequency used. This means that to use a baud rate to 4800, the cassette recorder has to be capable of reproducing 4.8KHz. Given a recorder with a response of 10KHz, it would be possible to use a baud rate of 9600, assuming the read/write software can go that fast. The version discussed here and used by myself and several other Belgian Nascom owners, and owners of other Z80 based machines runs at 3900 baud. This speed can be handled by the existing Nas-bug T4 and Nas-Sys software. The system has been found to be extremely reliable even on very large tape files.

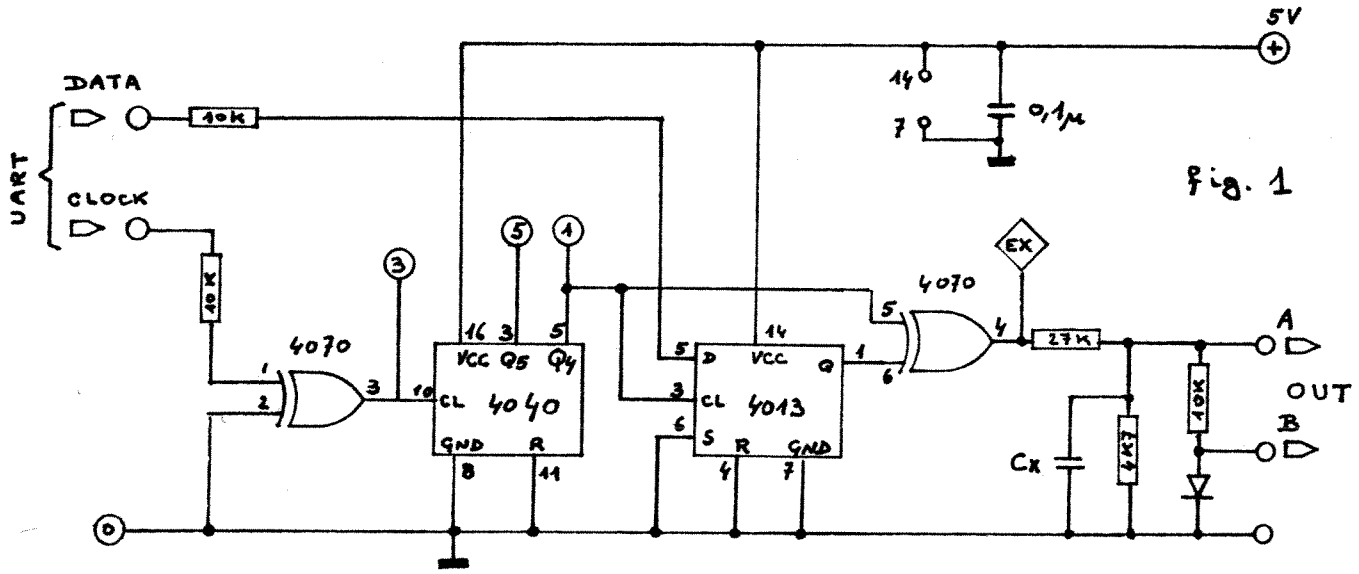
Generation of the Manchester II system signal is simple and is done by combining the clock and data signals into a composite signal using a NOR gate, pulses

are produced on each negative going transition of the clock signal. This causes the signal to have a positive going edge when the data bit is 0 and a negative going edge when the data bit is 1. When there is no change of data between consecutive clock pulses, pulses of the opposite sense to the original data bit are produced but are ignored by the decoding hardware.

Decoding of the recorded signal is not so simple, but makes use of the fact that if the signal does not change sign during a clock period, the data itself has not changed sign. The incoming pulses are detected by monostable which gives a pulse if the signal is longer that half a clock period. The pulse from the monostable triggers a flipflop, the output of which will echo the original recorded data, assuming that the data start was synchronised in the first place.

The Manchester II encoder

The bit train for the encoder is provided by the Nascom UART as before. As the data rate is to be 3900 baud, and the UART clock must be sixteen times the baud rate, a clock frequency of 62.5KHz is required. On the Nascom 1 this may be obtained from pin 11 of IC1. To avoid complication within the encoder (fig. 1), a 4040 divider is used, fed with 62.5KHz from the same point, the incoming frequency being divided by 16 to produce the 3.9KHz clock rate for the encoder, to match the data rate coming from the UART. A 4013 flipflop is used to ensure that the data changes with the positive edge of the 3.9KHz clock before being fed to one quarter of a 4070 quad NOR gate which does the actual encoding. The output signal is fed to a resistive divider and the recorder signal may be taken from the high or low level output depending upon the recorder requirements.



Cx - 10nF for 3900 baud
- 6n8F for 4800 baud

DATA - from Nascom 1 IC29 pin 25
CLOCK - from Nascom 1 IC1 pin 11
- also connect to central posn. of LK4

The Manchester II decoder

Fig. 2 is the circuit diagram of the decoder. The cassette output is fed into a cheap voltage comparator IC, TL081, which squares the signal, the output of which is fed to a quarter of a 4070 which further improves the signal shape. The shaped signal is fed to both the data input of the output flipflop, and the positive trigger input of half of a 4038 monostable which regenerates the clock. The time constant of the monostable is adjusted by the preset pot P, and to ease adjustment, this should be a multiturn type. The second half of the 4038 is used without a time constant to convert the regenerated pulses into spikes which clock the data into the 4013 output flipflop. The data and data complements are available at the Q outputs of the 4013. The output of one of these is connected to the data input of the Nascom UART. Unfortunately it is not possible to determine which data output should be connected to the UART until the unit is completed, when the correct connection should be determined experimentally.

This comes about because the Manchester II system is edge sensitive and it depends upon whether the cassette recorder inverts the data out relative to the data in (some do). If tapes recorded on different recorders are to be replayed, it might be advisable to fit a switch to select the appropriate Q output.

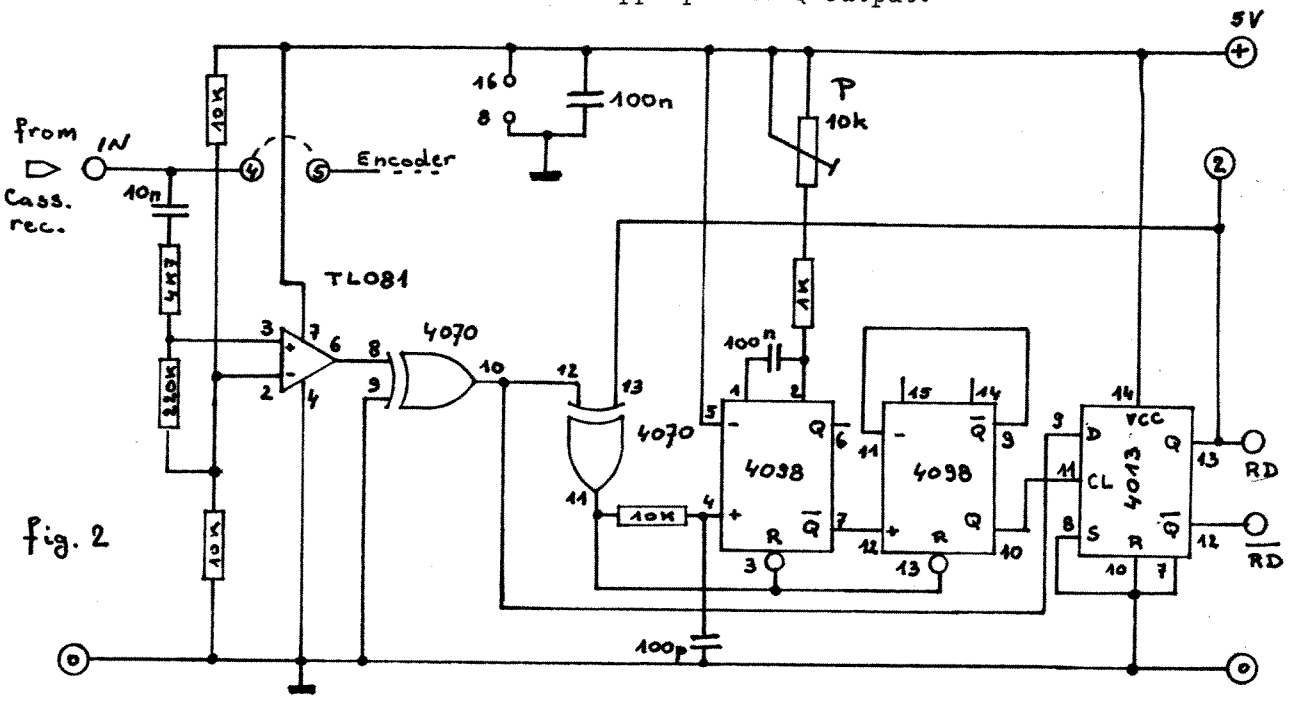


fig. 2

Adjustment

The encoder does not require any adjustment, and correct operation can be checked either with a 'scope or by recording a block of memory filled with code 03H, in which case the recorded data will have an even tone when listened to. The only decoder adjustment is the preset pot P, which may be achieved either with a 'scope by connecting the output of the encoder to the decoder input and then adjusting for correct decoding by displaying both the input and the output; or without a 'scope by first recording several K's worth of 03H and coarse adjusting for correct reloading on replay. Fine adjustment is then made by recording the system monitor several times over, and adjusting for correct reloading.

Figure 3 shows a small modification which may be added to reduce the tendency to pick up spurious noise. Pin 2 of the 74LS00 is connected to pin 3 of G1 of the tape control circuit published in 'Computing Today', February 1980, or to pin 12 of IC41 if the former is not installed.

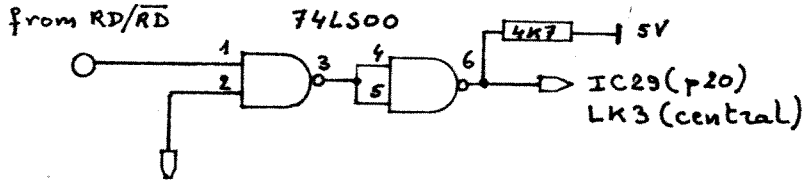


fig. 3

The whole circuit may be connected to the Nascom 1 using the switching system illustrated in fig. 4. The four pole four way switch gives the following functions:

1 - 3900 baud normal polarity
2 - 3900 baud inverse polarity
3 - 488 baud (twice Nascom 1 normal)
4 - 244 baud (Nascom 1 normal)

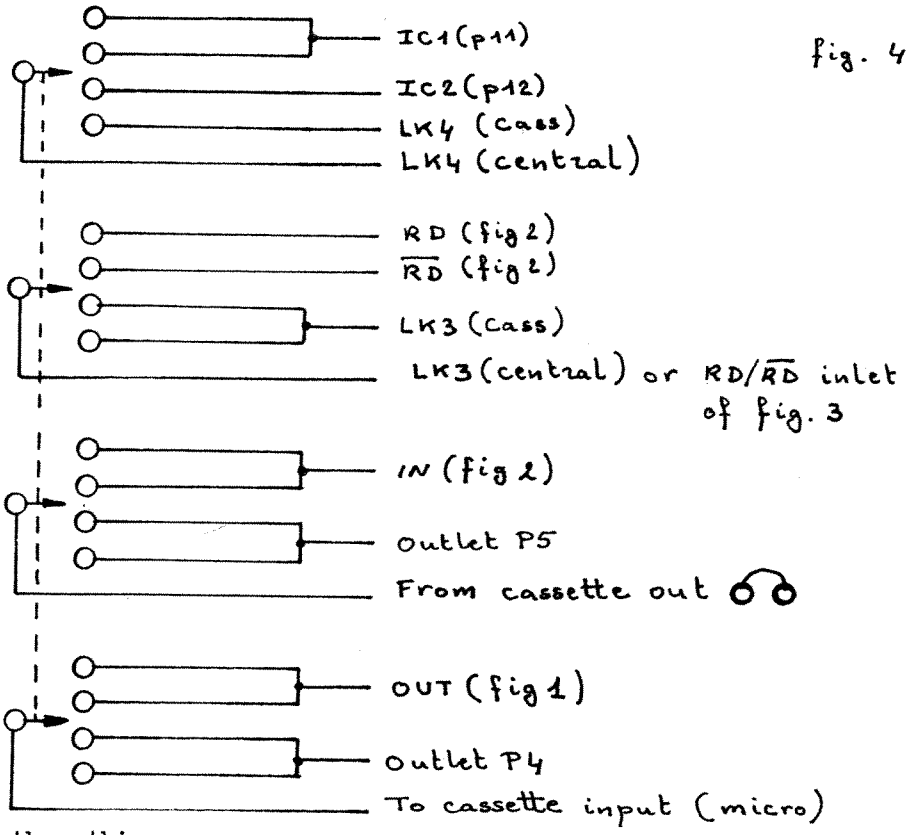


fig. 4

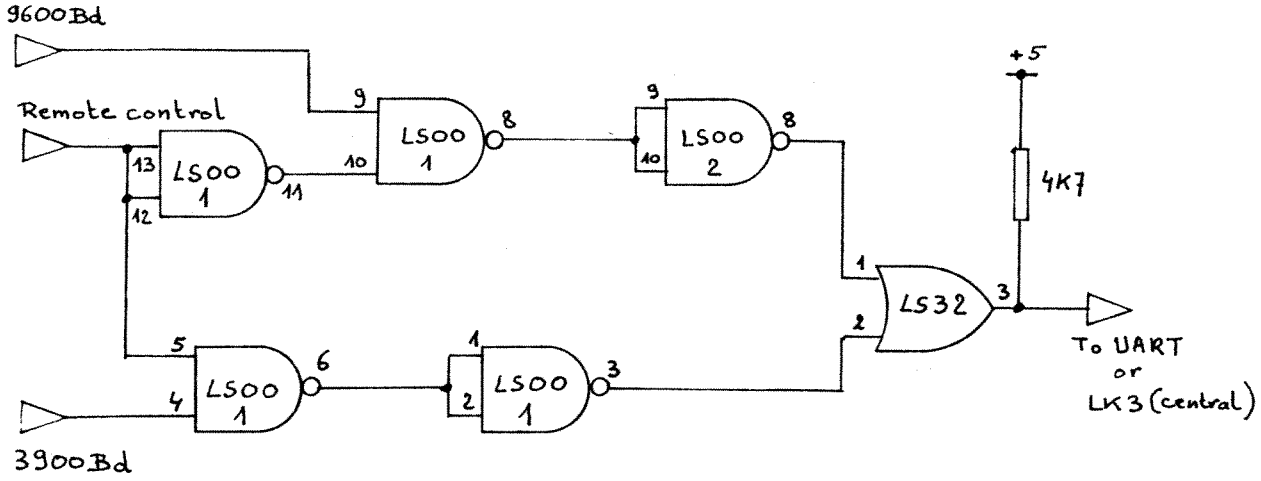
And now to other things

Recently I was able to buy a second hand Anadex DP8000 printer, and I will explain the interface to a Nascom 1, as this could be of interest to some people. The DP8000 is a dot matrix impact printer with RS232, 20 - 60 mA loop and parallel interfaces. The print rate is 112 characters per second.

I decided to use the RS232 interface at 9600 baud. Two things had to be considered, how to generate the 9600 baud clock signal, and how to control the printer. The first question was quickly solved by changing the TTY clock oscillator, IC33, so that it oscillated as 153.6KHz (16 times the baud rate), the following changes were made:

R30 changed to 1K8
VR1 changed to 10K
C12 changed to 300pF

I now had to switch two different clock rates to the UART, 62.5KHz for tape and 153.6KHz for the printer, this was achieved by using the circuit in fig. 5., and remotely controlling the change over by the state of the 'tape LED'. The control signal should be taken from pin 3 of G1 of the Computing Today tape controller, or from pin 12 of IC41 if the former is not installed. When the tape LED is active, during a Read, Write, Verify, CSAVE or CLOAD command, the clock rate is switched to 62.5KHz, at all other times it is switched to 153.6KHz.

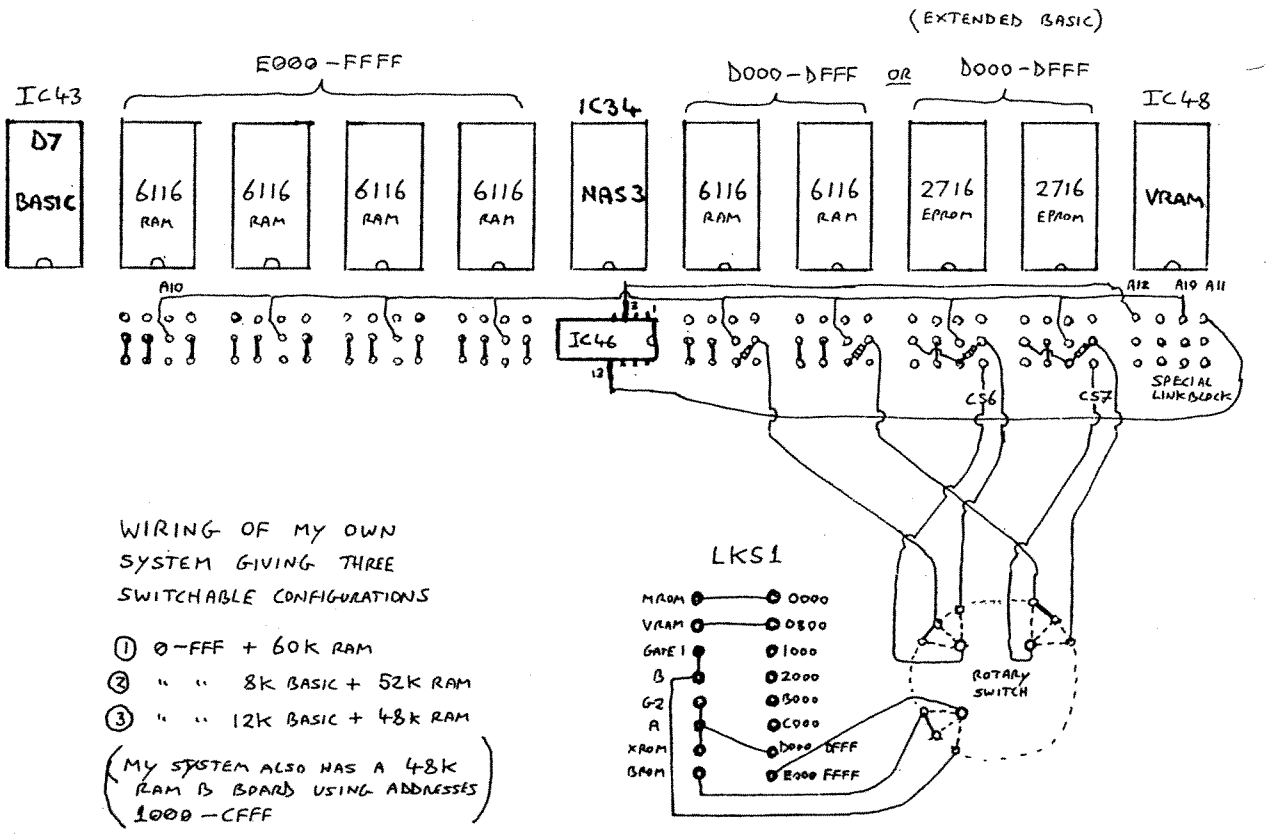


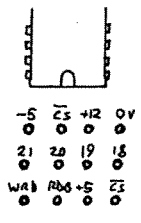
Handling the printer was the next problem, characters are output at 9600 baud, which is a lot faster than the printer can print them. The printer has a 1K buffer, but this is soon filled. Fortunately the printer is equipped with a handshake signal which may be connected to pin 2 of SK2, signalling when the buffer is full, or the printer is out of paper, etc. A simple piece of software supervises the printer. Before sending a character the following code is called:

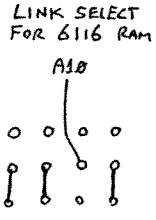
```
        PUSH AF          ; Save the flags and the A register
BUSY    IN A,(0)         ; Get the input into the A register
        RLA              ; Move the bit to be tested into the C flag
        JR C,BUSY        ; Test and loop if busy
        POP AF
```

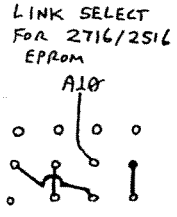
After which the software may direct the character to be sent to the printer.

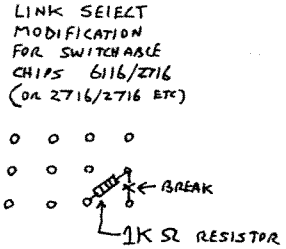
This finishes my summary of the major changes I have made to my Nascom 1, I hope the above will be of help to someone, and that it will solve some of their problems.











PINS 3 & 13 OF IC46 MUST BE CONNECTED TO A12 & A11

MODIFICATIONS TO ALLOW THE USE OF 2K RAMS AND EPROMS ON THE NASCOM 2. JEFF ROLLASON &2

2K 2716 EPROM & 6116 RAM for the Nascom 2 Main Board

by J. Rollason

Having read E.P.T. Anderson's interesting article (INMC80-5) on installing 2K EPROMs on the Nascom 2, I endeavoured to use the same technique to instal 6116 2K static RAMs but without success. Eventually I had to resort to the last option of actually trying to understand the address decoding on the Nascom 2 in order to come up with an alternative solution to the problem.

I found that it was only necessary to increase the 1K bandwidth pulses from the address Multiplexer IC46 (CS0 - CS7) to 2K and to connect pin 19 of the linkblock to A10 on the address bus.

The resulting solution is simple and flexible and can also be used with 2716 EPROMs requiring no extra components and less wiring than the published solution (without which I would not have attempted this).

The modifications for 6116 Rams are:

1. Wire up the link blocks as for 4118 static RAM except pin 19 which should be connected to A10 available on the link block below the VRAM IC 48 (I prefer to keep all my wiring on the component side of the board).

2. Lift out IC46 and bend out pins 3 and 13 to the horizontal position and carefully solder wires from A12 & A11 (available on the linkblock as above) to these.

3. Wire the appropriate decode pads as normal on LKS1. Note that since each block is now 8K and not 4K then extra decode addresses will need to be connected to each block e.g. to assign block A to addresses 1000-2FFF would require pins 5, 6, 13 and 14 to be mutually connected.

4. Switches LSW1/7,8 both need to be down (Open)

If 2716's are required then the Link block should be configured as in the diagram.

Switching RAM/EPROM or EPROM/EPROM

To do this it is necessary to pull pin 18 to +5 volts using a 1K resistor. The appropriate chip pulse can then be switched to or from the relevant chip without the chip enable causing spurious read/writes to occur. If it is only desired to select Block A instead of Block B or external memory /Block A or B, then it is only necessary to switch the address decodes on LKS1 to the appropriate RAM/ROM selects on pins 1 to 8 (or to open circuit).

Warning!

Note that "even" 4K blocks (e.g. 2000, 4000 etc.) will normally occur on the lower 2 chips in each block, whereas odd 4K addresses will occur on the upper 2 chips i.e. it is not possible to select two odd addresses to the same block (e.g. B000 & D000). This should not be a problem. Note !!! Pin 9 on LKS1 is for E000-FFFF, not E000-EFFF as claimed!

4K RAMS and EPROMs

It appears that the same technique could be applied for 4K chips, in which case pins 3 & 13 of IC46 would be connected to A13 & A12.

THE KIDDIES GUIDE TO Z80 ASSEMBLER PROGRAMMING

by D. R. Hunt

The Crossroads of personal computing (it goes on and on).

Part: The Seventh, the last

Putting it all to use.
Making programs do things.

It seems no time at all since I wrote the last episode of Kiddies Guide, although it's now early July, so it must be at least six weeks. This episode I have determined will be the last. It's taken some eighteen months and roughly 36,000 words to get this far (36,107 words, I haven't counted them, but my trusty Spell Guard program has, that's including this episode). I feel that it is perhaps time for a change, and to that end, my (Disk) Pen will be turned elsewhere in future. Thank you to all those people who wrote me encouraging letters, and thank you to those few who wrote letters highly critical of my literary style, finding it somewhat jarring on the nerves. To my critics, I can only reply that the style has been deliberately chatty with the hope that it might engender some interest in what is after all a very dry subject. Most people seemed to like it, but perhaps to get some practice at writing on more serious matters the style of this episode should be toned down somewhat, and the style made a little more turgid.

So, for this last episode, I have decided to look at the uses of a computer outside the pure enjoyment of writing software that works or the equally rewarding but considerably more frustrating task of debugging software that doesn't. Firstly, it occurs to me that I haven't published my philosphical view of what a computer is. How do I justify my statement that the computer is a remote mental extension of the mind of the man who programs it ?

Given that the computer is an intelligent machine, in that it may be preprogrammed (taught) to follow a logical course and make logical decisions based on the outside factors placed at it's inputs, it should therefore be possible to adapt the machine to take command of the outside factors which affect it. Of course I'm not talking about computers of science fiction which for instance can adjust, say the weather to suit themselves, or are capable of taking intelligent decisions beyond the scope of a thinking person. Rather, I am considering the computer from the point of view that it has somewhat less intelligence than an embryo woodlouse, and is capable of nothing without being imbued with the intelligence of the person who programs it. In otherwords, the computer IS the remote extension of the programmer. It can perform tasks set it by the programmer without supervision. Once set running by some other person it may perform these tasks without the programmer even being aware that machine is acting upon his instructions written may be months or years previously. However, it's intelligent progress through those tasks has already been predetermined from the stored image of the logical process of the programmer's mind in writing the program. It is incapable of any choice outside the program, and most of all it is incapable of a single 'original' thought.

The machine is, on the other hand extremely fast, and given satisfactory environmental conditions, meaning freedom from mains glitches, internal faults, etc, it is infallible. Or, rather, only as fallible as the programmer when the original program was written. Something outside the scope of the original program can always occur, and the machine fail to perform satisfactorily. This is not the fault of the machine, but of the lack of foresight of the programmer in failing to predict the fault condition. Software failsafe, and automatic learning techniques already exist for this sort of occurrence, but again, these rely on the programmer to predict that the unforseen could happen, and for him to endow that machine with the necessary intelligence to cope with whatever problems might occur. Again the onus is on the programmer. The machine can not learn from it's mistakes unless there is a program to allow it to learn from it's mistakes, and so on. This type of argument is never ending, as the next level down is the program to allow the machine to correct the

mistakes learnt in correcting the mistakes from the original program and so on. The ultimate learning machine cannot be programmed, as no programmer exists with the answers to all things, therefore the ultimate computer can not be built. No machine, no matter how intelligent, can be more intelligent that the man who programs it. All this is within the bounds of philosphical thought and current technology. So it seems that we are safe for a while yet from the computer that thinks it is God.

On the other hand, things in nature are always changing. As man's knowledge expands and his comprehension of things around him grows, who is to say that philosophies will not change. A way may be found to enable to machine to preprogram itself from the environmental factors which surround it. It may be able to pass it's intelligence from one machine to the next learning as it goes. It would probably take on some form of locomotion and have sensory and manipulative capability. When that day comes, the machine will probably have evolved into to bipedal structure for locomotion, have two arms terminating in hands with fingers for manipulation, a large single central processor in a safe place on top of the body structure. In fact the computer would probably be indestinguishable from the person who programs a computer today. You think that is far fetched ? I only hope I shall not be around when (not if) such a machine is built.

What has this to do with the computer and the outside world ? Not a lot really, it is simply an extension of the ways in which the micro is already affecting our lives. The vision of the mythical microprocessor controlled washing machine always springs in to view at this point. Well it now exists. What about processor controlled radios, televisions, video tape recorders, cameras, etc. These are here, and the micro is already finding it's way into more computer oriented equipment; printers with intelligence, dedicated disk controllers, intelligent keyboards and the like. In fact the medium sized micro these days does not contain a single microprocessor (although the one that does the number crunching will always be thought of as THE micro), no the medium sized micro computer already contains four or five dedicated microprocessors subserviant to the THE MAIN MICRO. What about the way these communicate. As a simple example I intend to discuss printer interfaces, and the whole theme of this article came about in a strange way, mainly from my feeling rather piqued by an insignificant action on the part of my favourite micro company.

A couple of days ago I reread the Nascom 'Dealer and OEM Technical Newsletter No. 3', paying attention instead of glancing through. Now this irregular newsletter is published by Lucas at Warwick and I attribute most of it to the pen of Mike Hessey, Nascom's Engineering Manager, a very capable engineer, ardent Nascom fan, and generally nice person to know (stop blushing Mike). The Nascom newsletter is distributed free to Nascom dealers and contains a stack of useful information although to date it has been unnecessary to draw attention to it as a fair chunck of what has been published so far has already appeared in the pages of the Program Power magazine, INMC, INMC-80 and this rag, albeit uncredited and now covered by a Lucas copyright. Which brings me to the point.

Part of the Nascom Applications Note AN-0006 caught my eye as being somewhat familiar, this concerns the software for driving parallel printers and would appear to be a re-write of the Centronics driving routines I sent Nascom, gratis, shortly after Lucas took over Nascom. Now I agree that the software driving routines published in the Nascom newsletter could have been arrived at independantly, as there are only a couple of ways of sensibly driving a parallel printer with the Nascom hardware after all, but I somehow doubt it. It's not the fact that they have published the software I object to, after all I wrote it to help other people, and I'm glad to know that it was considered adequate for general publication amongst Nascom dealers. No, it's the fact that it's not credited and now covered by a Lucas copyright I object to. You may have noticed that when I crib routines from other people that I always credit the source I pinched it from, after all, it's the polite thing to do. Once I wondered why Richard Beal put a copyright notice in the NAS-SYS source when he had declared it public domain software anyway, so I asked him, and he told me it was so that he could retain

some control over illicit developments of NAS-SYS away from the original philosophy of the design. So in future perhaps it would be wise of me to retain the copyright of any generally useful, silly little routines I write, and be wary about sending them in the direction of Warwick. If Nascom want to make something of it, I can certainly establish 'prior art' in connection with that piece.

So on to printer interfaces. Printers fall into two groups, those known as parallel printers and those known as serial printers. These definitions have nothing to do with mode of printing employed, the defintions refer to the method of feeding the computer with the data it is to print. Now the sort of printer we are concerned with, that is those likely to be attached to a home or small business micro are almost invariably themselves under the control of an internal microprocessor. Micro's are parallel devices, in that data is transfered about internally in parallel chunks usually eight bits wide. So it follows that a particularly convenient method of transfering data into this sort of printer, is in a parallel fashion via an input port. This explains why with few exceptions parallel printers are anything between £30.00 and £100.00 cheaper than the equivalent serial version. In the serial printer the data is sent one bit at a time at a fixed BAUD rate, in a similar (but by no means identical) fashion to the data being sent to a cassette recorder. Because the incoming serial data has to be converted to parallel data within the printer additional components in the form of a UART and its associated clock generator would be used, this explains the higher price for the serial printer, although not always the disproportionate prices charged in some instances.

The description of the electrical signals and timing used to control the printer is known as a protocol. The term protocol can also imply the type of physical plugs and sockets used to connect the printer, although this is often not the case. There are a number of different protocols in use, but fortunately these are mainly international standards or protocols that have been in use for so long that they have become internationally recognised although not necessarily having the authority of national and international standards committees behind them.

The two protocols most likely to be encountered are the RS232-C serial protocol and the 'Centronics' parallel protocol. RS232 has been around for many decades and has its origins in early teletype and signalling equipment. It is an international standard (although sometimes known by different names, V24, for instance), and provides adequate defintions for the electrical signals, the physical connections and the data rates, although the last is somewhat anachronistic and in most instances is replaced by a series of higher data rates more in keeping with modern equipment.

The Centronics protocol is more recent in origin, and gains it's name from the large printer manufacturer of that name. It seems to have become far more popular than the original designers intended because of its simplicity and ease of application to microprocessor based equipment. As far as I am aware it is not an agreed standard, and minor variations on a theme have been noted, it does however lay down the main connections, the type of connectors and the system timing, and seems to have gained general acceptance by a large number of printer manufacturers. It should be noted that recently Centronics themselves have departed from their own standard by changing the type of connector used on some of their more recent products at the cheaper end of their range although the electrical and timing specifications remain the same.

Each protocol has its own advantages and disadvantages. With RS232 the advantages are considerable immunity from extraneous noise coupled with the ability to transmit data over long distances (up to a kilometre or so) using a two or three wire system. Its disadvantages are that interface circuitry is somewhat more complicated when compared with the Centronics, and the data transmission speed is severely limited when data is to be transmitted over long distances. When used over short distances the data rate can be significantly increased, but the limit is still nowhere near as fast as the Centronics. The usual maximum practical speed for asynchronous RS232 data

transmission over short distances is 19200 BAUD, for maximum distance, speeds as low as 110 and 300 BAUD are typical. Variations on the RS232 allow for higher data rates but at greater complexity and usually by what is known as synchronous transmission where the system clock frequency is transmitted either as a component part of the data or sent separately along with the data. Synchronous transmission needs the clock to be sent over the same path as the data so that the clock suffers similar phase shifts to phase shifts which occur in the signals when serial data is transmitted over long distances.

The Centronics protocol is almost the reverse of the RS232. When compared with RS232 its noise immunity is poor, and the distances for data transmission should be measured in metres rather than kilometres. It is a multiwire system, typically eleven wires are needed, and if noise is to be kept at bay, these should be 'twisted pair' cables needing twenty or more wires in all. However, it is very fast and transfer speeds of 250000 BAUD are typical. The protocol also lends itself to direct interfacing with microprocessor equipment. In home and small business installations, the advantages offered by RS232 in terms of noise immunity are minimal as the distance between the host device and the printer is usually not more than a metre or so, thus the Centronics protocol offers significant advantages, because of its simpler interface requirements and lower cost.

This leads us to an interesting point concerning both Nascom and Gemini equipment. Both processor cards are equipped with the necessary hardware to cater for either serial RS232 or Centronics parallel, but both operating systems, NAS-SYS and RP/M, are only supplied with the necessary software to drive serial printers, and not the Centronics parallel interface. To be fair to Gemini, RP/M 2 now incorporates Centronics parallel driving routines, but as far as I know RP/M 2 is not yet available as a replacement for RP/M. This anomaly did not come about be accident, but shows the speed of change within the world of microprocessing. When NAS-SYS 3 was designed, a bit over two years ago with RP/M following a year later, only the large Centronics printers, type 101 and larger, were available and these cost four figures even second hand. On the other hand second hand serial printers whilst still expensive were at least available and new ones could be purchased for under a thousand pounds. Now, eighteen months later, an entirely adequate printer with a Centronics interface can be purchased new for just under £200 if you shop around.

The fact that both Nascom and Gemini are equipped with entirely adequate RS232 hardware and software means that I need not dwell on the RS232 side of things, except to cover the aspect of interfacing. Driving the printer is a matter of reading and understanding the manuals. It is all there, if you can not understand the words, then in the case of NAS-SYS try reading the software. In the case of RP/M if you can not understand it try reading my commentary on RP/M to be published shortly, failing that, the Zaks or Osbourne CP/M commentaries (Zaks for fun reading with inaccuracies, the Osbourne for a weightier and in many instances more abstruse treatment).

Handshaking is an important part of any protocol when applied to printers. It allows the printer to control the rate at which the data is sent. Put simplistically, this is the the case where data is being sent to a printer at, say, 1000 characters per second, where the printer is only capable of printing, say, 30 characters a second. The printer is either going to miss most of the incoming data or tie itself in knots trying to accomplish the impossible. All but the most primitive printers employ a handshake signal. This is a signal returned to the host device to indicate that the printer is ready to accept data. In an RS232 implementation this is known as the Data Terminal Ready (DTR) signal, indicating that the printer is ready to accept incoming data, whilst in the Centronics protocol, this is known as the BUSY signal, indicating that the printer is busy, so do not send data. With RS232 there is another protocol known as the XON/XOFF signal, this is rarely used and does not concern us here.

Taking the serial case first. On a Nascom there is no input provided for the DTR signal to be received by the computer. However, there are one or two spare inputs available on the keyboard port (one or two depending upon whether the machine is a Nascom 2 or a Nascom 1). By long standing convention with Nascoms, bit 7 of the keyboard port is used for this purpose. On a Nascom 2 this is conveniently returned to test point 3 whilst on a Nascom 1 this is available on the keyboard plug. Note that this input is at TTL levels, that is the signal applied to this pin must be between +5 and 0 volts. Also note that this input is not 'pulled up' to +5 volts on the Nascom 1, and only 'pulled up' on the Nascom 2 by a 10K resistor which is located for some improbable reason on the keyboard. This lack of 'pull up' on the keyboard input tends towards atrocious noise immunity and in either case the inclusion of a 470R resistor mounted on the back of the pcb, connected between the input pin and some nearby +5 volt point is a good idea.

Now with some printers the DTR signal is provided at TTL levels and this may be connected directly to bit 7 of the Nascom keyboard port. More typically the DTR signal swings between +12 and -12 volts, this being the standard voltages employed by the RS232 protocol. Direct connection of this signal to a TTL input will lead to the instant death of the input latch and possibly other damage. Some method of limiting or converting the +12/-12 volt signal to an acceptable +5/0 volt swing must be used. The ideal answer is a 4.7 volt zener diode with the cathode (the end with the coloured band) connected to the input pin, and the anode connected to the 0 volt rail. As the zener is reverse biased when the +12 volt signal is applied, it will conduct and limit the incoming voltage to +4.7 volts. When a -12 volt signal is applied the diode it is forward biased and the voltage will be limited to about -0.6 volts. Entirely suitable for the TTL input. Unfortunately, the zener diode can not be connected directly across the incoming DTR line, as, if the sending device, the printer, is capable of supplying any amount of current the conduction of the diode would allow a considerable current to pass, damaging either the diode or the printer or both. A series limiting resistor of 150R provides the answer. This should be connected in series with the incoming DTR line. The complete circuit is shown below:

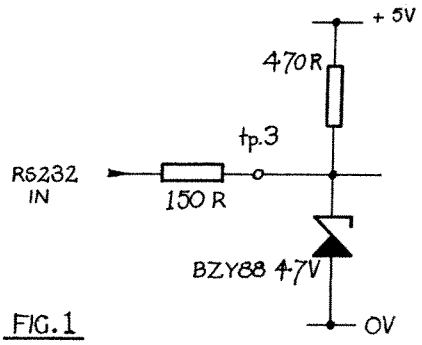


FIG.1

On the Gemini GM811 and GM813 boards connection for the DTR signal is provided in the for of the Clear To Send (CTS) input. This is another line in the RS232 protocol and is the complement of the DTR output. By connecting the two, the DTR output from the printer indicates to the CTS input on the computer that all is ready and data transmission can commence. On the Gemini cards there is no need to convert the DTR signal to TTL levels as the CTS input on the Gemini is fully RS232 compatible. The CTS input is also TTL compatible, although noise immunity may be improved by changing the values of resistors on the main board. The details in the technical manuals are adequate, so refer to them if the DTR signal is at TTL levels.

When considering the RS232 protocols what has been previously described is somewhat simplified, and RS232 has other signals which may or may not play a part in the handshaking process. Some printers, usually those with keyboards and designed for two way communication may require additional signals to enable them to print. My old Texas Instruments 745 and the Teletype 43 are two such printers. The additional

signals are the Request To Send (RTS), the Data Set Ready (DSR) and the Data Carrier Detect (DCD) signals. These are primarily concerned with modem control where the printer is attached to a remote computer via a telephone modem. The RTS signal is similar to the DTR signal, it is a request sent to the host device to send data. The DSR signal is sent from the host device to indicate data is ready to be sent, and would the printer like to accept it. The DCD signal is a signal sent from the host device to indicate that the modem is receiving a valid signal and that data could arrive at any at any time. As the printer will not print without the DSR and DCD signals being active, these have to be forced to a valid condition. This is done by the simple expedient of connecting the RTS output pin to the DSR and DCD input pins. In other words, the printer indicates that it is ready to accept data, and sends a signal to indicate this, as it is connected back into the DCD and DSR inputs, it automatically obliges by setting itself in a condition ready to accept incoming data. The DTR signal then acts as a handshake line as before.

The connector for the RS232 is what is known as D25 connector, and has 25 pins, one row of 13 pins and 1 row of 12 pins. The female sockets usually have the pin numbers moulded in the plastic adjacent to the pins, the plugs usually have the pin numbers moulded on the connection side of the plug. Equipment is usually fitted with sockets, whilst leads usually have plugs at each end. The pin connections are as follows:

1		Protective ground (chassis ground and mains earth).
2	TXD	Transmit data. Output. The signal from the terminal to the host device.
3	RXD	Receive data. Input. The signal from the host device to the terminal.
4	RTS	Request to send. Output. From the terminal to the host device.
5	CTS	Clear to send. Input. From the host device to the terminal.
6	DTR	Data set ready. Input. From the host device to the terminal.
7		Signal ground (Note: May not be common with (1))
8	DCD	Data Carrier Detect. From the host device to the terminal.
20	DTR	Data terminal ready. From the terminal to the host device.

The cable is usually connected pin 1 to pin 1, 2 to 2, etc. However, it must be apparent from the above that the sense of the RTS, CTS and DTR varies dependent upon use, the DTR or RTS signals being connected to the CTS inputs depending upon which is the host and which is the receiving device. This usually taken care of by jumper links inside the terminal and the host. When connecting an unknown device to another it is helpful to discover which thinks it is the terminal and which thinks it is the host, as the handshaking will not work in the case where two terminals or two hosts are connected together. Watch out for the protective and signal grounds. On something like a Gemini and a Nascom where the 0 volt rail is usually (or should be) connected to mains earth these may be considered as the same thing, but the TI 745 they are not connected together, and the signal ground floats about 100 volts above the protective ground if the mains earth of the printer is left unconnected. If in doubt, always connect the mains earth of the printer to the mains earth of the mains plug, and connect the signal grounds only.

So to making the handshaking work. This is fully implemented on the Gemini, in fact the Gemini serial interface will not work unless the handshake is connected even if it is not required. On the Nascom the handshake is not implemented in NAS-SYS and therefore must be provided. The scheme is simple. Data to be sent to the printer is intercepted by the U command and directed to a small routine which monitors the incoming handshake signal and delays until the printer is ready to accept data. Let us consider this. The handshake signal will be at the +12 volt level when the printer is ready to accept data and at the -12 volt level when the printer is not ready to accept data. The hardware interface described above will translate this into a '1' for ready and a '0' for not ready. All that is required then is a program that will read the keyboard port, and go into a loop scanning the keyboard port until such time as the printer is ready. As data is to be read in the A register from the port the current contents of A register which is the character to be sent must be saved. The simplest program is as follows:

```
        PUSH AF         ; Save the character
LOOP:   IN A,(00)       ; Get the keyboard port
        BIT 7,A         ; Test bit 7
        JR Z,LOOP       ; If 0 go round again
        POP AF          ; Get the character back again
        RCAL SRLX       ; Call the serial output routine to send it
        RET             ; Leave the handshake routine and carry on
```

That should explain itself, there is nothing complicated about it, it uses the Nascom output routines to do the hard work. Note that SRLX is called from within the handshake program, this means that it is not necessary to set the X command on, although the last parameter given to the X command will be the default protocol for the output. If the X command is to be used, simply omit the RCAL SRLX, and the routine will work as a handshake routine only.

Driving a printer fitted with a Centronics input is a little more complicated. As has already been mentioned, the Centronics output is parallel so use is made of the Nascom or Gemini onboard PIO device. The PIO device ought to be the subject of an article in it's own right. Howard Birkett covered it way way back in INMC 1 or 2, and I had a go at describing it in INMC 3. Although these magazines are long since out of print, Howard's article is reprinted in the 'Best of INMC' which is still available. I will not dwell on the PIO device itself, simply list the actions needed to prime the device to make it work as Centonics type interface.

Before starting to write an interface program it is necessary to understand what it is it is intended to drive. The Centronics interface works like this.

1 Seven bits of data are placed on the output
2 A slight delay to allow the data to settle
3 A strobe pulse is generated to cause the printer to latch in the data
4 A slight delay to allow the data to be latched
5 The strobe pulse is removed
6 The printer generates an ACKnowledge pulse
7 The printer also generates a BUSY signal which remains active until any print action is complete.

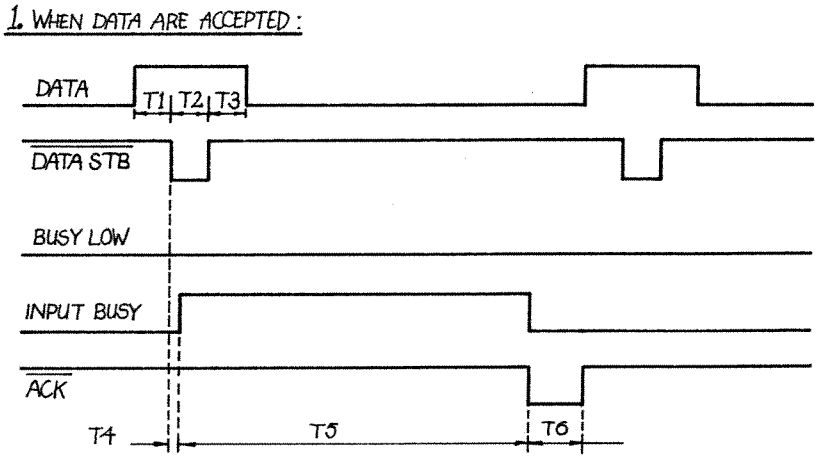
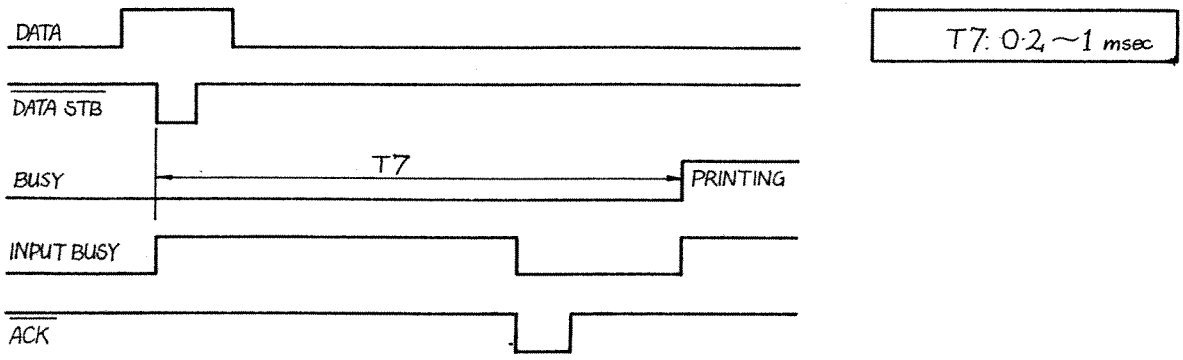


FIG.2

T1 ~ T3: 1 μ sec min.
T4: 100 nsec max.
T5: 0·1 ~ 0·5 msec
T6: 6 ~ 8 μsec

2. WHEN THE DATA BUFFER IS FULL:



There are variations, firstly, most graphic printers require eight bits of data not seven. The timing of the strobe pulse is subject to a minimum length, and the settle times tend to vary from printer to printer and also on the length of the interconnecting cables. The input requirements of the printer will most likely be at variance with the output driving capability of the PIO device. So how to overcome these problems ?

It is not significant that the printer may or may not require eight bits of data. If eight bits are not required simply do not connect the eighth bit. Having said that, the Centronics 730 alone of the whole Centronics range requires only seven bits, but will print double width if the eighth bit is not tied to ground.

The settle time can be found empirically by changing the number of NOPs in the driver routine, although those given below are have been found to work with almost all printers. The strobe pulse length must not be less than the minimum period, but making it longer does affect the operation, so the time chosen is one which has been found to work with all printers tried.

It has been noted that the printer sends an ACK signal after each character. At first sight this may appear to a good candidate for a handshake signal. However, almost all Centronics type printers can not accept data during a print and carriage return/line feed cycle, so the BUSY line becomes a better candidate as this is active during the print and CRLF cycles. The ACK signal is simply ignored, and in practice this has been no impediment.

Lastly, the input characteristics of the printer inputs vary widely, but are typically at TTL levels but 'pulled up' internally with 150R resistors for better noise immunity. This represents an input load of about 8 TTL unit loads, whilst the B side of the PIO device is (on paper) only capable of driving 1 TTL unit load and the A side of the PIO is only capable of an unspecified lesser amount. It would seem then that the PIO would require buffering to be able to drive the Centronics input direct. But MOS devices are considerably more rugged than we have been led to believe, and manufacturers' specifications are always on the conservative side. In practice the PIO drives the Centronics type interface very well without buffering despite the apparent 8 times overload presented to the outputs. Collectively the numbers of Nascoms and Geminis driving Centronics type interfaces must add up to many thousands of hours satisfactory operation. Of course overloading the PIO device is likely to shorten its statistical life, but as this runs into many hundreds of thousands of hours, premature failure after a few thousand hours is hardly likely to be an inconvenience. I have heard of only one PIO fatality caused through driving a Centronics type interface, and the cost of a new PIO is only slightly more than the cost of incorporating two dibrectional buffers such as the 74LS245.

The Centronics connector is different to the D25 connector. It is a 36 way connector arranged so that all signal leads are on the top row, whilst the opposite pins on the bottom row are all signal grounds. This allows for the use of twisted pair cables to improve the noise immunity. To confuse the issue, Centronics refer to the eight data lines as DATA 1 thro' DATA 8, whilst computer people think of these as Bits 0 thro' 7. DATA 1 is connected to BIT 0, DATA 2 to BIT 1, etc. Beware, as has already been noted, there are some variations on the theme, Epson for instance use one of the lower pins as a device select pin on some models, so it pays to read the manual referring to an individual printer before connecting up. A typical Centronics configuration is as follows:

1	DATA STROBE	19	Twisted pair ground for pin 1
2	DATA 1	20	Twisted pair ground for pin 2
3	DATA 2	21	Twisted pair ground for pin 3
4	DATA 3	22	Twisted pair ground for pin 4
5	DATA 4	23	Twisted pair ground for pin 5
6	DATA 5	24	Twisted pair ground for pin 6
7	DATA 6	25	Twisted pair ground for pin 7
8	DATA 7	26	Twisted pair ground for pin 8
9	DATA 8	27	Twisted pair ground for pin 9
10	ACK	28	Twisted pair ground for pin 10
11	BUSY	29	Twisted pair ground for pin 11
12		30	Twisted pair ground for pin 31
13		31	INPUT PRIME
14	System ground	32	
15		33	
16	System ground	34	
17	Protective ground	35	
18		36	

The DATA lines are connected to the PIO port B, bits 0 to 7, the STROBE is connected to PIO port A, bit 1, whilst the BUSY line is connected to PIO port A bit 0. The warnings given earlier about system and protective grounds apply equally in the Centonics interface, so care is advised.

Again, as with the serial output, the scheme is simple and follows similar lines. However, the PIO device must first be primed. The PIO has several modes of operation, input, output, bit mode and bidirectional/control mode, and uses a control port for each half of the PIO device to program it's respective mode. The control port for the PIO is addressed at the port address + 2. On a Nascom the actual input/output port for the A half of the PIO is port 04H whilst its control port is port 06H, likewise, the B half input/output is port 05H and its control is port 07H. On the Gemini Multiboard machines port A I/O is port 0B4H and its control is 0B6H, port B I/O is port 0B5H and its control is 0B7H. If we use labels for these addresses, the following listing will make sense regardless of the actual port addresses. The initialization routine need only be called once prior to use, but note that RESETting the computer will return the PIO device to it's default mode, and the routine will need to be called again. These routine are due to Richard Beal (again) as they are more elegant than my original version, although somewhat more complex.

```
BASE    EQU XX          ; Base address, 04H for Nascom, 0B4H for Gemini
CENPAD  EQU BASE        ; Port A data I/O
CENPBD  EQU BASE+1      ; Port B data I/O
CENPAC  EQU BASE+2      ; Port A control
CENPBC  EQU BASE+3      ; Port B control

; Initialize the ports, port B to output, port a to bidirectional
        LD A,0FFH       ; Force all outputs high, default condition is output.
        OUT (CENPAD),A
        OUT (CENPBD),A
        OUT (CENPAC),A  ; Tell port A it will use the bidirectional/control mode
        LD A,0FDH       ; Send the mask to say which bits are in and which are out
```

```
        OUT (CENPAC),A  ;   bit 1 is the STROBE (output), the rest are inputs
        LD A,OFFH       ; Tell port B it will use the bidirectional/control mode
        OUT (CENPBC),A
        XOR A           ; Send the mask to say which bits are in and which are out
        OUT (CENPBC),A  ;   all bits are set to output.
        RET             ; Exit from the initialization routine.
```

So the ports have been set up. The U command in NAS-SYS, or the LST vector in RP/M should be set to point to the output routine, and study of the manuals should reveal what is required to be done. Like the serial routine the first thing that happens is that the character to be printed is PUSHed on the stack whilst the BUSY line is examined until it is free. The character is then POPped back and sent to the data port. Bit one is then toggled to cause a strobe pulse thus latching the data into the printer.

```
        PUSH AF         ; Push the character out of the way

; Examine the BUSY line until free
LOOP:   IN A,(CENPAD)   ; Get port A into the accumulator
        RRA             ; Rotate the byte, shifting bit 0 into the Carry
        JR C,LOOP       ; If the C flag is now set, the printer is BUSY

; The printer is now free, send the character
        POP AF          ; Get the character back
        PUSH AF         ; Save it from being corrupted by the strobe pulse
        OUT (CENPBD),A  ; Send the byte to the printer
        NOP             ; Wait for the data to settle
        LD A,0FH        ; Send a strobe pulse by toggling bit 1
        OUT (CENPAD),A
        NOP             ; Wait for the strobe pulse to settle
        LD A,OFFH       ; Take the strobe pulse away again
        OUT (CENPAD),a
        POP AF          ; Get the character back
        RET             ; Exit from the routine
```

And that is all there is to it. Of course, there will be problems with both the serial and the parallel routines, but 'thinking down' to the bit level where it is all happening will help. I can offer little practical advise at this stage. Playing around with software is fun, and if you don't have a glimmer of insight after all I've written, I'm afraid you will be forever doomed to high level languages and their restrictions. Playing at the very heart of the machine is my idea of fun. But of course it requires a thorough working knowledge of both the hardware and the affects of the software.

In drawing this series to an end, I like to think I've helped someone along the way. I know what I have written is incomplete and inaccurate in places but that is probably not so important as the overview of systems I have presented. To encapsulate all that should be written on the subject would require several very fat books, and falls into the province of the professional writer. Unfortunately the most prolific writers do not necessarily have a deep understanding of what it is they are trying to convey. They are quite adapt at disguising this fact, usually with jargon, and the indadequacies of this sort of book is often overlooked. There is nothing to beat hands on experience.

Thank you for reading this series I hope you found it useful.

D. R. Hunt.

GM809 and Eight Inch Drives

by D. Parkinson

Every once in a while I get an enquiry (or statement!) passed to me about the possiblity (or impossibility!) of using the Gemini GM809 disk controller card (for which I have written various BIOS's) with double density 8" drives. Rather than publish a few lines of code showing how it can be done I've presented it in the form of a short tutorial in the hope that people other than those with 8" drives and GM809 will find it useful.

Problem: How to code a routine to handle the data transfer for the Western Digital 1797 disk controller and 8" double density drives? (In the following discussion it is assumed that the target system has a Z80A running at 4MHz with no Wait states. However the same arguments apply to a 2MHz system and double density 5.25" drives, but all the times should be doubled).

It will be useful, though not essential, if the following items are to hand:

GM809 circuit diagram.
1797 Data sheet.
Z80 Data sheet.

An 8" DD disk has a serial data transfer rate of 250kbits/sec which results in a byte transfer to/from the controller every 16us (64 T-states with a 4MHz clock). Therefore it would appear that the data transfer routine must be able to move data at this rate. Can this be achieved?

Once the 1797 controller has been given a Read or Write instruction the controlling program then has to transfer the data to/from the disk under the control of the DRQ (Data ReQuest) signal from the 1797. This DRQ signal appears externally (on pin 38) of the 1797, and also internally as bit 1 of the status byte that can be read from the 1797. When the data transfer is complete (or the 1797 gives up because of an error condition), the INTRQ signal, (INTerrupt ReQuest), on pin 39 comes on. (Also bit 0, the "Busy" flag, gets reset in the internal status register). With GM809 the external DRQ and INTRQ signals are connected to bits 7 and 0 of another input port, and so may be read directly as well as via the bits in the 1797's internal status register.

Shown below is a first attempt at coding the data transfer routine for the Read command. All it has to do is transfer a byte every time a DRQ (Data ReQuest) occurs, and exit when an IRQ (Interrupt ReQuest) occurs. Note that it uses the externally read status bits.

```
        code                                                T states
        ====                                                ========
        ...set up all the registers...
LOOP:   IN      A,(STATUS)      ; Read DRQ/IRQ bits       - 11
        AND     81H             ; Mask DRQ and IRQ        -  7
        JR      Z,LOOP          ; Loop if no request      -  7 / 12
        JP      P,IRQ           ; Exit on IRQ             - 10
        IN      A,(DATA)        ; Read the data byte      - 11
        LD      (HL),A          ; Store in memory buffer-  7
        INC     HL              ; Bump the address        -  6
        JR      LOOP            ; Get next byte           - 12
IRQ:    ..........
                        Total T states (no wait for DRQ)  =73
                        Total T states (one wait for DRQ) =73+37=110
```

The AND 81H masks out the DRQ and INTRQ status bits. If neither is set the Z flag gets set and so the routine waits. Once one or other bit is set it will fall through to the following bit of code. At this point there is no need to do a

further test to find out which bit was set as the AND instruction will have also set the sign bit in the flag register to indicate the state of bit 7. Thus if it is Positive (i.e. bit 7 was 0), then bit 0 must have been set to create the NZ result of the AND.

Turning now to the execution timing of the code: It can be seen that even the fastest loop is outside the time limit that has to met, so the code - simple as it is - needs revising to make the loop tighter. Setting the mask byte of 81H in a register and performing the AND with that register will save 3 T states, and making the last JR a JP will save a further 2, but that only reduces the loop to 68 T states, still 4 over the target. The "Exit on IRQ" JP in the middle of the code is a bit of a time waster as it only serves to terminate the loop and contributes nothing to the data transfer itself, and so could be moved with some benefit.

With some re-organisation the code could now look like this:

```
        ....
        LD      B,81H           ; Set the mask
        JR      LOOP            ; Skip
DRQ:    IN      A,(DATA)        ; Read the DATA byte    - 11
        LD      (HL),A          ; Store                 -  7
        INC     HL              ; Bump address          -  6
LOOP:   IN      A,(STATUS)      ; Wait for DRQ/IRQ      - 11
        AND     B               ; Mask/test             -  4
        JR      Z,LOOP          ; Wait if neither       -  7 / 12
        JP      M,DRQ           ; Jump if DRQ           - 10
        ....                    ; Else done
                        Total states ( no wait for DRQ) = 56
                        Total states ( one wait for DRQ)= 56 + 27 = 83
```

The re-organisation has effectively removed one JP from the critical loop (at the expense of an extra JR at the start to get into the loop), and has improved the timing figures as a result. Going straight through is now well under the 64 T-state limit, but the extra 27 T-states that occur when there is a wait for the DRQ are slightly worrying. A more detailed cosideration is required to see if they matter.

Assume that a DRQ has appeared towards the end of the execution of the IN instruction that is checking the status port, and has just been missed. Now the number of T states that occur until the data byte is actually read is: 4+12+11+4+7+10+11 = 59 - so that byte will be read before the controller assembles the next byte (which comes after 64 T states). When the label LOOP is next reached the DRQ bit will be set (because it has taken >64 T-sates to get back there) so the next data transfer loop will be a short one (56 T states) and some time will be caught up. This will continue until the routine overshoots and has to wait again. Anyway it all seems possible.

The code for a Write transfer is similar:-

```
DRQ:    LD      A,(HL)
        OUT     (DATA),A
        .......
```

So the time has come to code the routines up and to try them out.

Disaster! They don't work! Why not?

When all else fails read the data sheet. For those readers who like a challenge read no further until you've found the answer in the 1797 data sheet.....

The relevant parts of the 1797 data sheet are the boxes READ ENABLE TIMING on page 19, and WRITE ENABLE TIMING on page 20. Each contains a footnote in minute print in the bottom left-hand corner - there is a "worst case service time". The data may be coming at the nominal rate of one byte every 16us, but when reading, the byte must be read within 13.5us (54 T states) of the DRQ signal appearing, and when writing, the next byte must be supplied within 11.5us (46 T states) of the DRQ request - which is even worse! The code developed above does not meet these figures, so how can it be compressed further?

Fortunately there is a way out of this dilema. It utilises one of the facets of the unique Z80 instructions, and is catered for in the hardware of GM809. You may have wondered why the additional Status port is provided (at E4h) through which the DRQ and IRQ bits can be checked, when the same bits can be accessed through the internal status register of the 1797. Looking further at the circuit diagram will reveal that, as well as DRQ and IRQ being connected to the external input port, there is a connection to the "Motor on" monostable, (which should present a '0' during normal disk I/O), and that the remaining lines are connected to OV. Turning to the I/O instructions in the Z80 data sheets it can be seen that while the 8080 compatible IN A,(n) does not affect any of the bits in the flag register, the extra Z80 instructions - IN r,(C) where the port address is held in register C - does set the flags as a result of this operation. So if one of these instructions is used, the "Input and Test" operation can be completed in one go, and a very tight polling loop will result. However note that this can only be done if the "Wait" condition sets the flags suitably - this is why it is necessary to use a special port for the DRQ and IRQ inputs, as the internal Status register of the 1797 contains a variety of other bits which would prevent this happening. Adding the "Motor on" signal to the same port ensures that, if for some reason neither IRQ or DRQ occur, then the loop will be broken when the "Motor on" monostable finally times out.

The revised code for Read and Write now looks like this:

```
; ****** Write Section *****

        ld      c,status        ; C -> Special Status port
write:  ld      a,(hl)          ; Get the next byte ready
        inc     hl
wwait:  in      b,(c)           ; Input and test
        jr      z,wwait         ; Loop if no request
        out     (data$reg),a    ; Send the byte (in case was a DRQ)
        jp      m,write         ; Carry on if it was a DRQ
        .......                 ; IRQ or timeout exit

; ****** Read Section ******

        .......
        ld      c,status        ; C -> Special Status port
        jr      rwait           ; Enter the routine
read:   in      a,(data$reg)    ; Get the byte
        ld      (hl),a          ; Put in memory buffer
        inc     hl
rwait:  in      a,(c)           ; Input & test
        jr      z,rwait         ; Loop if no request
        jp      m,read          ; Jump on DRQ
        .......                 ; IRQ or timeout exit
```

Note that in the WRITE section a byte is pre-fetched and output immediately in response to the Status byte being non-zero. This ensures that the worst case service time of 11.5us is met, and the final (and extra) byte output this way (on the IRQ) will be ignored by the 1797 and so does not matter. However if it is intended to use the value of HL on return from the Write routine for any reason this point should be remembered.

In true text book style I leave it as an exercise to the reader to work out the timing of the above sections of code!

One other point for budding Bios writers to bear in mind - The most useful (and accurate) part of the 1797 data sheet are the various flow charts of command execution. For example the text for Read/Write commands goes...

..Upon receipt of the Type II command, the busy status Bit is set. If the E flag = 1 (this is the normal case) HLD is made active and HLT is sampled after a 15ms delay.....

Referring to the flow charts it can be seen that after Busy is set the "Ready" line is checked, and only if it is true will the 1797 continue with the command, otherwise it sets IRQ and terminates. If there has been no disk activity for a while there will be no Ready signal as the controller will have unloaded the heads and as a result deselected the drive. Thus, in order to get the Read/Write commands to execute, the software has to ensure that the drive is selected. (My software checks the Ready line before all Read/Writes, and issues a Seek command to the current track if it is found to be false. This activates HLD without actually stepping the heads at all, and ensures that the drive is selected and the Ready line represents the true state of the drive. All type I commands, of which SEEK is one, execute irrespective of the state of the Ready line - see the flowcharts).

Finally a note about hardware mods. These are covered in the manual supplied with the card, but don't forget that as well as changing a few straps some components need changing. The frequency of the VCO has to be doubled so capacitor C5 (100pF) should be changed to 50pF. The loop filter components (C4 and R12) also need to be changed. Early versions of the manual escaped with a misprint, C4 should be changed to 0.33uF not 33uF! One thing not noted in the manual is the fact that the "Motor on" and "Motor start-up delay" monostables are still connected through to the Ready line. These connections are not required for 8" drives and their presence is an irritant. Rather than butchering your board to get rid of them I suggest that you just disconnect one end of resistor R5. This has the effect of making the "motor on" time infinite, and the signals from the monostables will no longer interfere with the Ready signal from the selected drive.

Please note I have not written a full double-density Bios for GM809 and 8" drives - so don't try ringing up for a copy! (Anyway I've moved recently). In writing your own I suggest you follow the guide lines in the CP/M alteration guide.

Start with 8" single density (The only standard format there is!). Most 8" disks come ready formatted to this standard so you needn't worry about a format program initially. In your current environment write and test a keyboard routine and a screen routine. Next write and debug a disk read routine. If you're using the CP/M distribution disk at this point ensure it is write protected! Expand this routine to "GETSYS" (see the CP/M alteration guide). Now do the write part (using a scratch disk!) and work up to "PUTSYS". Then put the various routines together to form your Bios. Next use GETSYS to read in the system tracks of the distribution disk. Copy your Bios (and Boot) (suitably assembled) over the distribution Bios (and Boot) and use PUTSYS to write the system back to a fresh disk. Then, provided you've made no boobs, you're away, and can go on to refine the software in a disk-based environment.

WOULD YOU BELIEVE IT?

==========

Dear Dealer, The fault on my Nascom occurs as follows. As you will see, the display is stable with some degree of flickery as the CPU works. Hitting reset, some characters change, but some stay exactly the same all the time. Now and then the computer enters the HALT state.

Besides this, the machine is dead. Anon.

RANDOM RUMOURS (& TRUTHS)

by S. Monger

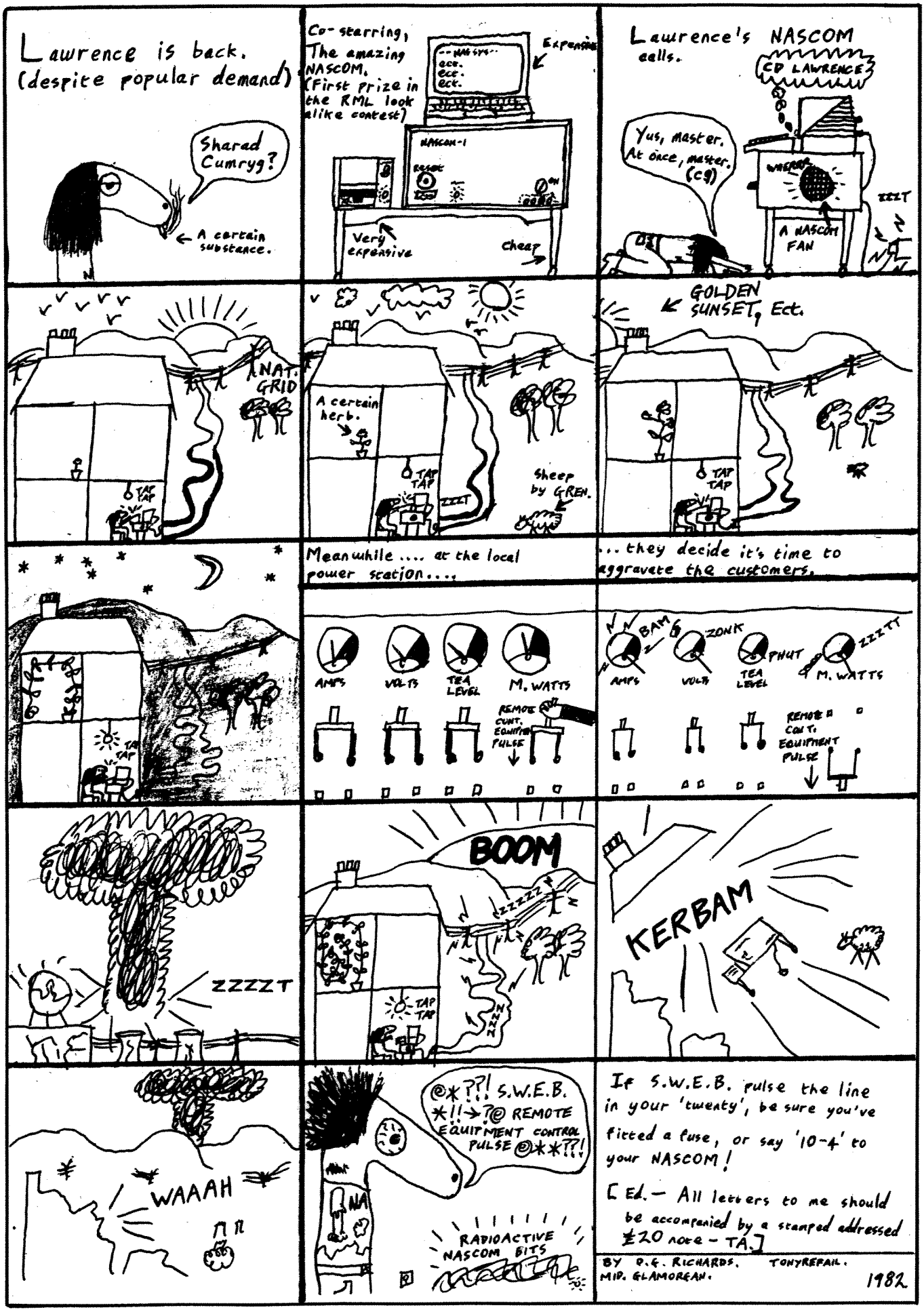
Well, hello again, and what wonderful new products and rumours of new products are there for me to tell you about this time? And will you believe anything I say anyway?

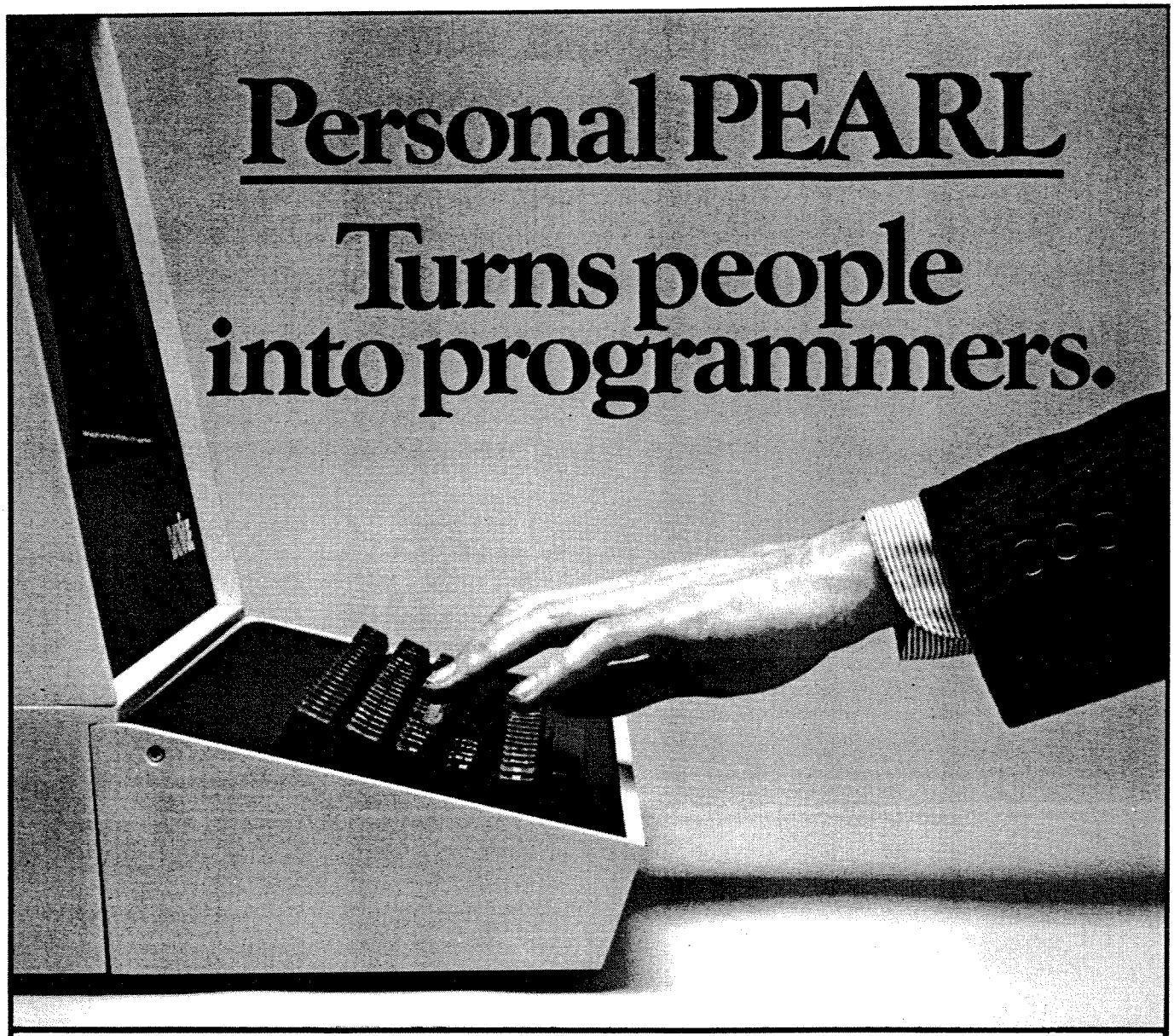
It seems that new product lines never come in ones, always in quantity. Nascom have recently been publicising there 'Nas-net' network system, based on Nasom 3 slaves with a Nascom disk system running Nas-Dos as the master. But they are not alone in the networking world. IO Research have started talking about an intelligent add-on 'IO-Net' board, and Gemini have let (only a little) info. out on their 'Multi-Net' system based on cased MultiBoard slaves and a Galaxy with hard disk as the master. Nas-net is available, but I'm afraid I've no dates on the others. As I've mentioned hard disks I also ought to add that Gemini Winchester units (pre-production?) have been spotted (5.4 MByte formatted & £1500-ish), and also a 'Winchester - TBA' line has appeared on Nascom's latest price list. Better start saving!

Whilst on about Nascom's price list, I wonder if they are running a secret competition to see who can find the most mistakes? There are some lovely entries, like a 'UART 6402' at £5.75, but a '6402 UART' at £6.50; an 'MK3881 PIO' at £7, but a 'PIO MK3881' at £4 ("It must be the way I tell 'em."); some 'C/O cassettes', and some 'Bridge rectifiles'! Interesting to also note quite a few price rises including; Nascom 3 £549 (or £599 - 64K), FDC card £145, NAS-DOS £60, CP/M £125, and oddly the only recently available AVC at £185. Mind you there has also just been a special reduced price offer on Nascom's disk drives, so I hope some of you got in there quick.

Having just mentioned things going up, a little story now of something going down. A company. Those of you who 'go back a very long way' in micros (that means about three years) will remember Kerr Borland, then marketing director of Nascom. It was the Kerr Borland/John Marshall team that pushed the Nascom 1, then Nascom 2 so forcibly and successfully onto the 'micro-ignorant' market-place. Then, unfortunately, and to most people extremely surprisingly, Nascom went into receivership. Some months later (as most readers will know) JM started up Gemini Microcomputers and KB started Arfon Microelectronics. Arfon's first products were a speech board (for 80-BUS systems and other machines) and a light pen (compatible with Gemini's IVC), followed by a growing range of VIC20 peripherals. Now I hear, for reasons not really known, that Arfon has gone into receivership. I understand that the products will continue to be available from other companies, but as for Kerr, I have no idea of his current or future activities.

Several miscellaneous comments have floated through to me from the dealers. How about the story of the customer who enquired about the availability of a modem for his Nascom? It seems that this guy wanted his computer to be able to phone up TIM, the speaking clock, and then set up its Real Time Clock itself!.....some dealers are agreeing with claims that 'Pluto' is 'out-of-this-world' as it would appear that for all their efforts to get hold of them, they might as well be.....other dealers report Nascom 2s being 'thrown away' as users change to Gemini 811s & 813s - please throw them in my direction!!.....one dealer wonders if Interface Components Ltd's (initials ICL) recent name change to Amersham Computer Centre (initials ACC) gives a warning about the level of support to be given in future (think about this one).....and finally there was the guy who was told that one dealer's (No hints as to who, but their initials are HR) 'forthcoming' EPROM programmer was 'simply years ahead'; his response was that if their past record was anything to go by then that is precisely when it would be available!


Lawrence is back. (despite popular demand)
Sharad Cumryg?
A certain substance.
Co-starring, The amazing NASCOM. (First prize in the RML look alike contest)
Expensive
NASCAN-1
Very expensive
Cheap
Lawrence's NASCOM calls.
CD LAWRENCE
Yus, master. At once, master. (cq)
A NASCOM FAN
ZZZT
NAT. GRID
TAP TAP
A certain herb.
ZZZT
Sheep by GREN.
GOLDEN SUNSET, Ect.
Meanwhile .... at the local power station....
... they decide it's time to aggravate the customers.
AMPS
VOLTS
TEA LEVEL
M. WATTS
REMOTE CUNT. EQUITMENT PULSE
BAM
ZONK
PHUT
ZZZTT
ZZZZT
BOOM
KERBAM
WAAAH
@*?!! S.W.E.B. *!!→?@ REMOTE EQUITMENT CONTROL PULSE @*?!!
RADIOACTIVE NASCOM BITS
If S.W.E.B. pulse the line in your 'twenty', be sure you've fitted a fuse, or say '10-4' to your NASCOM!
[Ed. – All letters to me should be accompanied by a stamped addressed £20 note – TA.]
BY D.G. RICHARDS. MID. GLAMOREAN.
TONYREFAIL.
1982



If you could program a computer by simply telling it the result you wanted, without using complex codes or languages, then anyone could become a programming professional. Sounds fantastic?

But now it's possible with Personal PEARL, and all for less than £200. It generates quality Business Programmes, Data Management, Costing, Mailing – in fact you create your own library of programmes that matches your operation today, and tomorrow.

After all, non-one understands your business better than you. So let Personal PEARL take the technology out of computer programming, and you'll find yourself writing professional business software – at the touch of a button.

Pearl Software International (UK) Limited,
PO Box 34, Poole, Dorset BH14 8AR
Tel: Parkstone (0202) 741275

Please send more details of Personal PEARL

Name: _____

Company: _____

Address: _____

_____ Tel: _____

PARKSTONE ELECTRICS
18, Station Road,
Lower Parkstone,
Poole, Dorset, BH14 8UB.

Tel: Parkstone (0202) 746555

adventure

SPECTRUM ZX81 BBC nascom

Colossal Adventure .. 16K/32K .. £8/£10
The classic mainframe game "Adventure", with all the treasures and creatures of the original. And with 70 extra rooms!

Adventure Quest ..... 16K/32K .. £8/£10
From the Great Forest, up Orc Mountain, braving fire, swamp and caverns on a quest against Tyranny. Face vampires, demons, wizards, 200-foot worms...

Adventure games are fascinating. You enter English phrases and the computer acts as a window to worlds of magic.

Every Level 9 adventure has over 200 individually described locations and a game may take weeks to solve! Only our combination of data and code compaction allows so much to be provided.

FREE P&P. NO VAT. Money back if unhappy. Supplied on TDK cassettes. Send order, describing your computer, or a SAE for full details of all our programs to:

LEVEL 9 COMPUTING
229 Hughenden Road, High Wycombe, Bucks

nascom Games

Missile Defence	£8
Asteroids	£8
Space Invasion	£7
Bomber	£5
Fantasy	£6
5 Games Tape ..... (Double Breakout+Gunner+ Wumpus+Surround+Minefield)	£6

Compression Assembler

- Source compressed to half the normal size so bigger programs fit in memory and load/save is twice as fast.
- ZEAP conversion utility to transfer your ZEAP sources.
- Easy and uncomplicated to use, with a full 22 page manual.
- Assembles at 3000 lines/minute!
- Only 6K and £12.

EXTENSION·BASIC

FOR NASCOM

Extension Basic adds 32 keywords to ROM BASIC and lets you define more yourself. EB is not just a toolkit: the new statements are used exactly as if they were provided in the BASIC ROM. In fact, it is effectively a compatible 12K BASIC.

Extension Basic provides: AUTO, BREAK, CALL, CHECK, COPY, DEC, DELAY, DELETE, EDIT, FIND, GET, HEX, IF..ELSE, INKEY, INLIN, LINE, PLOT, PRINT @, PUT, REPEAT ..UNTIL, RENUMBER, REDUCE, SPEED, TEST, TRACE, VDU, WHILE..WEND, WRAP, XLIST & XREF and all the ROM BASIC keywords.

EB is only 4K and costs £15 on cassette or £25 in ROM. The cassette version is relocatable to any address, for ROMs state start address and 2*2716/4*2708. Extension Basic has a 28 page manual.

Send order or SAE for details of EB and our range of Nascom programs to:

LEVEL 9 COMPUTING
229 Hughenden Road, High Wycombe, Bucks
(Nascom 1s need Nas-Sys & CB interface)

MicroValue

The Microvalue Group was formed in 1980 to offer complete support for Nascom products. The group has extended from this original base and now offers a wide range of other products, including the Gemini Multiboard range. All of the items in the Autumn Gemini Catalogue, included with this issue of 80-Bus News, are available from members of the Microvalue Group.

Other products stocked include a wide range of dot matrix and daisy wheel printers, monitors, Sinclair products, including the ZX81, Dragon and Sharp ranges of computers. All members can offer repair and back-up facilities for both the Nascom and Gemini products.

MicroValue

Microvalue Group members

Amersham Computer Centre Oakfield Corner Sycamore Road Amersham Bucks. Tel: 02403 – 22307	Bits and P.C's Computer Products Ltd. Leeds Computer Centre 60/62 Merrion Centre Leeds LS2 8NG Tel: 0532 – 458877	Electrovalue Ltd. 28 St. Judes Road Englefield Green Egham Surrey TW20 OHB Tel: 07843 – 3603
EV Computing 700 Burnage Lane Manchester M19 1NA Tel: 061 – 431 – 4866	Henry's Radio 404 Edgware Road London W.2. Tel: 01 – 402 – 6822	MDW (Electronics) 47/49 Woodbridge Road East Ipswich IP4 5QN Tel: 0473 – 78295
Off Records Ltd Computer House 58 Battersea Rise Clapham Junction London SW11 1HH Tel: 01 – 223 – 7730	Skytronics Ltd. Computerama 357 Derby Road Nottingham Tel: 0602 – 781742	Sktronics MBM 80 Bristol Street Birmingham B5 Tel: 021 – 622 – 6436
	Target Electronics Ltd. 16 Cherry Lane Bristol BS1 3BG Tel: 0272 – 421196	

Nascom & Gemini USERS

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

FEATURES:

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

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

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

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

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

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

FULLY BUFFERED ADDRESS, DATA AND CONTROL SIGNALS.

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

AVAILABLE NOW!

PRICES:

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

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

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

cheques and P.O.s to:

MICROCODE (CONTROL) LTD.
41a MOOR LANE, CLITHEROE, LANCS. BB7 1BE.
For further information Phone (0200) 27890

Please supply me with the following: Price

Total enclosed £

Address:

POST CODE

HENRY'S INCREDIBLE CP/M UTILITIES DISK.

All the things you ever wanted: The disk cataloguing and file dating suites. File compare and string and byte search utilities. Text file compression (and it's complementary expansion) programs. Universal disk repair utility (works on Nascom and the Gemini SD, DD and QD formats). Revised and correctly working SUBMIT.COM with interactive line input (no more XXX.SUB files unless you want to). The revised DRI PIP.COM including all the official fixes published a couple of months back. And many more. All are standard CP/M utilities, not system dependant. Available in Nascom and all Gemini formats. The price of this gift ? A mere £15.00 + VAT (£17.50 + VAT in Gemini SD format as it's too big to fit on one disk.)

Richard Beal's SYS overlay BIOSes for Gemini disk systems. SYSN7 the latest and last for Nascom/Gemini G805 systems. SYSB11 universal version for both Nascom/Gemini G809 and Gemini Multiboard systems (not Micropolis drive versions). £10.00 + VAT. Please state version and system configuration required.

Maths pack double precision for Nascom Basic	£13.00 + VAT
Maths pack handler for Nascom Basic	£9.95 + VAT
Command extender for Nascom Basic	£9.95 + VAT

HENRY'S RADIO phone 01-402 6822
404, Edgware Road,
London W2 1ED E & OE

32K NASCOM ADVENTURE

The REAL Adventure begins here, with this full-spec 32K version of the original classic mainframe Adventure, at last available for your Nascom in fast efficient Z80 code. No discs, no graphics needed -- just 32K RAM and NAS-SYS.

Somewhere nearby is Colossal Cave, where others have found fortunes in treasure, though some who enter are never seen again... Explore a vast underground labyrinth of caves, twisty passages and exotic rooms; outwit fierce enemies; collect all the many treasures and become a Grand Master!

Using the most advanced text compression techniques the authentic mainframe messages have been squeezed into 32K.

Supplied on high-quality C60 cassette with full instructions. Only £20 or a large gold nugget! Add 55p post and packing. Discount for Adve 16K owners. Send cheque or PO stating tape format (N1, 300), or SAE for further details on this and other software, to:

* SYRTIS SOFTWARE *

23, QUANTOCK Rd, BRIDGWATER, Som.

AMERSHAM COMPUTER CENTRE

80-BUS HARDWARE

GM811	Z80A CPU Board	£125.00
GM813	Z80A CPU +64K RAM Board	£225.00
GM812	Z80A Video Controller Board	£125.00
GM802K	16KRAM Kit (64k Board)	£ 80.00
GM802	64K Dynamic RAM Board	£125.00
GM809	Floppy Disk Controller	£125.00
GM803K	Eprom/Rom Board (Kit)	£ 55.00
GM816	I/O Board (RTC,CTC,3*PIO)	£125.00
RB32KC	32K Cmos Batt/Backed RAM	£170.00
IO824	8ch - 8bit A/D Board	£120.00
PLUTO	High-Res Colour Graphics Board, 2 Screen Memories (640h*288v) 3 bit Pixels	£399.00
BABY PLUTO	(320h*288v) 3 bit Pixels	£299.00
NAS AVC	Colour Graphics Board 80col (320h*256v) Pixels	£185.00
WT625	Teletext Colour Graphics Board,40 col* 24 rows	£136.00
WT910	Nasbus Sound Board	£ 49.50
NAS I/O	Kit Not populated	£ 45.00
GM810	8 Slot Motherboard 5A PSU	£ 69.50

MONITORS

9" AVT	High Res,Green or Amber	£ 99.00
12"GEM	High Res,Green or Amber	£130.00
14"BMC-1401	Colour Monitor,R G B (ideal for baby Pluto)	£299.00
14"HANTAREX	Hi-Res Colour,R G B This unit is one of the best monitors on the market today.Ideal for PLUTO.	£600.00

PRINTERS

EPSON MX80T/3	Dot/Mat/Tractor	£349.00
EPSON MX80FT/3	Dot/Mat/Friction/Trac	£389.00
EPSON MX100T/3	132col (as above)	£499.00
NEC PC8023BC	Dot/Mat/Friction/Trac	£389.00
SEIKOSHA GP100A	Dot/Mat/Tractor	£215.00
SMITH-CORONA TP1	Daisywheel/12cps	£485.00

NASCOM SOFTWARE TAPES

LEVEL 9	Extension Basic Tape	£ 15.00
CCsoft	Nas-Graphpac Tape	£ 20.00
NASCOM	Pascal Compiler Tape	£ 45.00
SIGMA	Zeap 2.0 Assembler Tape	£ 20.00
GEMINI	Nas-Dis/Debug Tape	£ 20.00
L-SOFT	Logic Soft Relocater Tape	£ 13.00

LEVEL 9 NASCOM GAMES TAPES

5 Games (Gunner/Wumpus etc)	£ 6.00
Missile Defence	£ 8.00
Asteroids	£ 8.00
Space Invasion	£ 7.00
Bomber	£ 5.00
Fantasy	£ 6.00
Nightmare Pork	£ 5.00
Colossal Adventure (32K)	£ 10.00

NASCOM FIRMWARE

LEVEL 9	Extension Basic (4*2708)	£ 25.00
NASCOM	Pascal Compiler	£ 75.00
POLYDATA	Polytext Text Editor	£ 35.00
GEMINI	Naspen Text Editor	£ 20.00
GEMINI	Nas-Sys 3 (2708/Nascom 1)	£ 20.00
GEMINI	Nas-Sys 3 (2716/Nascom 2)	£ 20.00
GEMINI	Nas Dis/Debug (4*2708)	£ 30.00
GEMINI	Imprint (For Imp Printer)	£ 20.00
SIGMA	Zeap 2.0 (2716)	£ 37.50
BITS & PC's	Programmers Aid (N/Sys1)	£ 20.00
BITS & PC's	Programmers Aid (N/Sys3)	£ 20.00

DISK OPERATING SYSTEMS FOR NASCOM

POLYDOS 1	for use with Gemini GM805	£ 90.00
POLYDOS 2	for use with GM815 + GM809	£ 90.00
CP/M 2.2	for use with GM815 + GM809	£100.00

GEMINI RPM/CPM SOFTWARE

	COMAL 80	Stuctured Basic	£100.00
*	GEM PEN	Text Editor / Formatter	£ 45.00
*	GEM ZAP	Assembler(screen editing)	£ 45.00
*	GEM DEBUG	Debug/Disassembler	£ 30.00
	COPY SB	Superbrain to Gemini (DDDS)	£ 30.00
	LIST/REPAIR	Recovers Lost Data Etc	£ 25.00
	DATAFLOW	Information Processer	£125.00
*	G BASIC	Graphics Basic (IVC)	£ 25.00
	GEM GRAPHPAC	Links to Mbasic (IVC)	£ 25.00
	COM-PAS	Pascal.Generates M/Code	£120.00
	QUIBS	Integrated Business Package	£400.00
	CP/M 2.2	Disk Operating System	£ 90.00

* Tape also available.
When ordering disks please specify the format.

MEDIA

			EACH	BOX(10)
SCOTCH C10	Cassettes		£ .60	£ 6.00
DYSAN 1042D	D/S disk	(M or P)	£ 5.00	£43.00
GEMINI 2D	D/S disk	(M or P)	£ 3.25	£30.00
DYSAN 1041D	S/S disk	(P)	£ 4.75	£40.00
WABASH 1D	D/S disk	(P)	£ 3.20	£30.00

(M) Suitable for Micropolis Drives
(P) Suitable for Pertec Drives

80-BUS SYSTEMS

QUANTUM	QM 2000 System (2.4M Bytes)	£2250.00
GEMINI	Galaxy 1 System (800K Bytes)	£1450.00
NASCOM 3	48K with Nas Sys 3 & Graphics	£ 499.00
*GEMINI	GM815 1 disk system (single)	£ 325.00
*GEMINI	GM815 2 disk system (dual)	£ 550.00
GEMINI	GM805 1 disk system (single)	£ 375.00
GEMINI	GM805 2 disk system (dual)	£ 580.00

* requires GM809 controller board

ORDERS ACCEPTED BY MAIL AND PHONE.
ACCESS AND VISA CARD HOLDERS WELCOME.

WE ARE OPEN MONDAY TO SATURDAY 9-5.30. PERSONAL CALLERS WELCOME.

Amersham Computer Centre Ltd., Oakfield Corner, Sycamore Road, Amersham, Buckinghamshire, HP6 6SU. Tel: 02403 22307. Telex 837788