

# inmc news

## issue 7

SPECIAL REVIEWS ISSUE

April/May 1980

### CONTENTS

Page 0	This is it.
Page 1	Chairman's Page.
Page 2	Letters to the Editor.
Page 7	Thought For The Day.
Page 8	Doctor Dark's Diary -5.
Page 10	ROM Graphics Set.
Page 11	Hardware Reviews.
	Nascom IMP Printer.
Page 14	Keyboard Case.
	ROM Graphics.
Page 15	Colour Graphics.
Page 16	Special IMP.
Page 17	Nas-Sys Mystery Program.
	Nascom 1 and 8K BASIC.
Page 18	BASIC "PRINT USING".
	Memory Mapping.
Page 19	ZEAP 2.0 Modifications.
Page 20	Nas-Sys Naughties.
Page 22	Memory plague - be gone!
Page 23	RAM B - welcome !
Page 24	Cheap Nascom 1 Expansion.
	B-BUG and BASIC.
Page 25	Moohlanding.
Page 27	Planning Your VDU Display
Page 28	Software Reviews.
	Nas-Dis 1.0.
Page 29	Super Debug.
Page 30	Xtal BASIC V2.2.
Page 33	Book Review.
	Z80 Instant Programs.
Page 35	Software Library.
	Nas-Sys Minimum System.
Page 36	Submitting Programs.
Page 37	The Money-Making Bit.
Page 39	Lawrence, Wherefore Art Thou ?

### INMC Services.

The INMC News is our main offering. Back issues are available for 70p each + 15p p & p per order. (40p Overseas.) Issues 1 & 2 count as one issue. We also run a program library - see inside for details.

Our address is:  
INMC, c/o Nascom Microcomputers,  
92 Broad St., Chesham, Bucks.

90p

PLEASE NOTE: WE ARE NOT NASCOM !! This address is purely a postbox and we cannot arrange to send you leaflets or answer all your sales and technical queries. Contact your Nascom distributor for these please.

CHAIRMAN'S BIT  
=====

Well, here we are again, yet another action packed issue, and only a month late (as usual). The Editor said, "Final copy first week April please." and as usual, we all ignored him until the last minute. Still we've got it all together, and it all goes off to the printers on Monday.

Now to the financial bit, members pay 5.00 per year for six issues, and what with rapidly rising postage and print costs, we are actually running at a loss. We aren't broke yet, but we soon will be. Part of the expense is letter writers who expect replies and don't send stamps for return postage. NUDGE. If you want a reply, 12p please, or you won't get one. Now we have no intention of foisting a surcharge on members (so you can put your cheque books away), but to try and make a bit of extra loot, we are going to start accepting advertising. Not a bookful with a couple of pages of text lost somewhere in it, but we reckon that if we sell a maximum of 5 pages per issue, we can then afford to bring the newsletter up from 40 to 48 pages, and we'll virtually cover our deficit. Maybe we'll make a modest profit. Who knows, we might be able to reduce the subs next year.


To start with, we've twisted Nascoms' arm, and another distributor. So if you sell goodies for Nascom, and you want to reach about 2,500 "Nasnuts", then approach the INMC. We bet we reach more Nascom owners at once than all that expensive newstand advertising. Get a slice of the action!!! Our rates are very modest.

Commercial over.

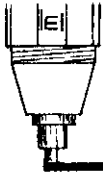
As you read this issue, it won't escape your attention that there are an awful lot of (perhaps that should be lot of awful) reviews. It didn't start out that way, but like "Topsy", it "Just grewed". Still, with this lot in print perhaps we can collect some loot from those people we didn't slate too badly, as well as would be advertisers. So we decided to turn this into a bumper reviews issue, and get them all out of the way in one go. Hope you like it. A lot of the reviews and articles have been submitted by members, and this is very encouraging. Our thanks, keep them coming.

Back to the subject of readers letters. Paul has landed me with a fair number this month, and a large number of those display a very severe shortcoming of some Nascom owners and embryo programmers. They can't read !!! Nascom manuals take a fair ammount of getting used to, but it is all there, and the vast majority of it is all correct. I have been tempted to write 'read the \*\*\*\*\* manual' over some of these letters and send them back like that. My natural politeness and modesty forbade. But I promise, next month I will. Its a lot quicker than writing a letter.

We've still got a lot of material left over for issue 8, so we'd better get on with it, in the meantime, happy programming.



D. R. Hunt.



Read, Try and Learn  
=====

Dear Sir,

There have been several letters from people who have built their Nascoms and have yet to acquire the "knack" of programming. Perhaps my experiences would hold out some hope.

Before last April when I bought my Nascom I knew very little about computers, in fact I wasn't even sure what Hexadecimal was. After building the kit and getting "hands-on experience" through the Software Manual things became a little clearer, but it wasn't until I bought the "Z80 Instruction Handbook" published by Scelbi that I really began to understand what it was all about. The Scelbi book is pocket sized and explains all the Z80 instructions in simple language, assuming that the reader knows very little, which was certainly true in my case.

It was about this time that my children were doing tables at school and so I thought of the "Table Tester" which times their answering the table of their choice, in random order, and if all answers are correct, compares their time with the previous best.

I have enclosed the program for consideration for the library, if only as an example of what can be done by a beginner. I can now see that it is not very well compiled, having, for instance, no entry verification. But IT DOES WORK, if only the correct keys are pressed.

Now to hardware.

No doubt there are others beside myself who have Nascom 1's and would like 2's but do not have the necessary. Apart from that, part of the reason for my buying a Nascom was that it was easily and, relatively, cheaply expanded. Apart from Nas-Sys and the Basic ROM, what else is required to bring the Nascom 1 to Nascom 2 standard (apart from speed)? Is there to be a conversion kit or is it to be done piecemeal?

My thanks to yourself and the INMC committee for such an enjoyable and informative magazine.

Yours faithfully,  
A. Marshall  
Huddersfield.

Ed. replies:

Well congratulations on "getting it together". Your program will be considered for the software library.

We are afraid that expanding an N1 to an N2 is piecemeal as apart from the items you mention, you will also require expansion RAM and buffer, an Econographics kit from NM, the Cottis Blandford cassette interface from Newbear, not to mention sundries like the "snow plough" and "Reset jump" multiplexer.

CUNNING PLOT (for frustrated virgins)  
=====

Dear Sir,

I think that I have discovered a plot cunningly devised by the computer fraternity to enlist new members of the right calibre. The code name for this plot is "Nascom". Realising that all good programmers enjoy a tough puzzle, they have designed a kit with built in frustration value. The price is low enough to attract any fool with a vague interest in finding out what the "micro-revolution" is all about, but many will not overcome the first hurdle of actually getting the machine to work properly. This cuts out most normal people with wives (or husbands), families, demanding jobs and other interests besides computing. With the hard core who now try to put the Nascom to use, natural selection again plays a part, in the form of the Nascom Programming Manual. This gives a few random clues about how to write programs. More clues can be found in the pages of the INMC magazines, but the whole plot cleverly ensures that only a devoted intellectual elite get to join the fraternity.

May I make a plea on behalf of thousands of virgin Nascom users who are struggling to rediscover the wheel? Nascom are missing a marvellous educational opportunity, unless they really don't want people to use their machines! Personally, I have some scientific background and a little general knowledge of computers, but still my experience has been of constant frustration. How anyone can survive with no previous knowledge is beyond me. Nascom should write a simple graded manual and distribute it to all Nascom owners. INMC is helping, but in a very piecemeal manner. I don't think the reply to Mr. Hewitt's letter (Issue 5) was very helpful. Would you tell someone who wants to know how a radio works to get a circuit diagram and find out what the symbols mean? Talk about jumping in the deep end! Surely a structured program can be broken down into small units which are simpler to understand? There must be someone out there who could get a kick out of dropping down to our level and really explaining the common routines used in programs.

I am sorry if this seems like just a long moan. I do feel that Nascom are particularly bad at the art of communication and I have tried to be constructive in my criticisms.

Yours faithfully,  
Brian Ward  
Meopham, Kent.

Editor's reply:

Well, is there anyone out there willing to write a 'Beginner's Guide to the Nascom'? If there is, then get writing and send it to us (we'll vet it). Take a look at books on the market, most of them are pretty useless for the beginner, but they still sell. So if you are any good at all, your're almost guaranteed a market.

STICK TO IT  
=====

Dear INMC Committee,

Thank you for your work in putting together an ever improving newsletter.

Twelve months ago I was in the same position as Mr. Hewitt (Letters, INMC5). After playing around with machine code and Zeap, I have now reached the stage where I have written a quite effective control program nearly 2K long, which sends and receives "Tonebursts" via a radio link, to identify the outstation, and cause certain operations to occur.

My message to Mr. Hewitt and people in his position is to stick at it, and particularly the section in the Program Manual on building up a simple program and stepping through it.

But I really do not think it is on to advise a beginner to try to start on such a complex program as Lollipop Lady.

By the way, I suggest all newcomers try to learn about machine code as well as Basic, as m/c is much more satisfying and can do so much more.

A couple of requests - probably impractical - but who knows?

- 1) Is it possible to publish or make available in the library, more information on NAS SYS? I have in mind perhaps some simpler flowcharts and more data on what each routine does to registers etc. This would make it easier to use the many routines available.
- 2) Could the software library offer, at some reasonable cost, a cassette option for longer programs?

Yours sincerely,  
C. Bowden  
Truro, Cornwall.

Ed. replies:

We have had quite a few requests for more info. on Nas-Sys, so we'll see what we can do. Any offers? As to cassettes, we have discussed this, but we feel it will be some time before we can offer this service.

STUCK BITS  
=====

Dear Sirs,

Thanks for publishing my letter in issue 5 and attempting to help. However shortly after writing to you I purchased a "Line graphics generator" from Comp Shop and after fitting this to my Nascom 1 all my problems became perfectly clear - 3 rows of my V.D.U. had bit 7 stuck permanently high.

This meant that most programs would run alright, but as Mastermind and Lollipop Lady both pick up data from the V.D.U. ram, this data would be read as incorrect even though it would be printed correctly on the screen.

After purchasing a new 2102 memory chip and fitting this in place of the existing IC20 all my troubles were over. Incidentally this fault also produces errors on the "L" command.

As "Doctor Dark" states in Issue 4 - just because the bit isn't displayed it still matters.

Thanks anyway for your help.

Yours  
R. Milton.  
Folkestone, Kent.

#### FLOATING BITS =====

Dear Sir,

Many thanks for INMC News - keep up the good work!

An addendum to the "Memory Plague" report - I hit a couple of memory problems with the STATIC memories. First was a 2708 which "flipped bits" when it got warm, changing say 5A to DA in the Load program. Second was a 2102 which started to output a few bits of its own about 20 minutes after switchon. The result in my case was continuous printing of a random character. Changing this chip for one in the VDU RAM position gave a regular pattern of dots on the screen when the chip warmed up.

Both faults took ages to track down, as the whole system worked fine from cold; trouble shooting tool which finally solved the problem? - the wife's hair dryer! The bright lads at Nascom were right about the 2708 but way off the beam on possible causes of continuous printing. Moral, keep changing the chips around - I had changed just about everything on the board except the RAM, as it worked perfectly in the memory tests!

Yours faithfully,  
H.R. Thornton.  
Largs, Ayrshire.

#### MIXED FEELINGS =====

Dear INMC,

As a paid up member, I would like to praise the very worthy efforts of the committee in producing the INMC News. Every issue a gem and this is encouraging more people to contribute, even me. I must admit to mixed

feelings on seeing my FRUIT MACHINE in Issue 5 but I am only too happy for other members to benefit. (Eddie Pounce is the common factor). This was the first program I wrote for our NASCOM some 12 months ago. The coding is not very elegant and the documentation is hand written and covers some 18 sheets so I do not think it worth putting in the library, especially now everyone has the object code. For those who like to fiddle, the three barrels are 16 bytes long each and occur starting from 0C96. As set up, the player makes a steady profit if cherry equivalents (✓) are held whenever possible. The jackpot odds are 1:4096 with no hold, 1:256 with one hold and 1:16 with two holds. Bon chance!

Yours,  
Gordon Alabaster  
Berkhamsted, Herts.

#### BASIC FACTS =====

Dear INMC,

Now for some flattery. I think that INMC news comes close to providing the best personal computing magazine going - no doubt because it only deals with Nascoms and not the plastic horrors available elsewhere.

The reason I am writing, however, is concerning the Nascom 8K ROM BASIC. I recently received my expensive piece of plastic and silicon and have been playing with it ever since. In the the course of this experimenting I have come up with a few facts that the grotty documentation ignores. For all I know I may be reinventing the wheel, but as I am an assiduous, not to say compulsive reader of computing magazines, and have not seen this info. published elsewhere, it may be helpful to pass on to our beloved readership.

1) As is well known, the BASIC reserved words, such as READ, DIM, PEEK etc. are stored as single bytes, using hex values of 80 and above. How to find what byte represents what word?

Examination of the BASIC text storage area shows that it always starts at address 10FAH and each line of text has the following format:

Bytes 1 and 2 : Address of next line, or free memory if line is the last line. Least significant byte first.

Bytes 3 and 4 : Line number in hex. L.S.B. first.

Bytes 5 to N-1 : The text string.

Byte N : 00H, which acts as line terminator.  
For example, the BASIC program,

```
10 PRINT "HELLO"  
20 END
```

maps to the following

```
10FA: 07 11 0A 00 9E 22 48 45 4C 4C 4F 22 00
1107: 0D 11 14 00 80 00
1114: Free memory.
```

Where 9E is equivalent to PRINT and 80 to END.

Having established this it is simple, but tedious to dissect BASIC programs and find the single byte representation of the reserved words. A much better way is to use the computer to give a list of byte values and corresponding words. This is readily done by writing a short assembler program whose output goes to the BASIC text storage area and, in effect, writes a BASIC program. This way you can work out the magic reserved words and their decimal equivalents.

What is the point of all this? Well, the BASIC allows BASIC programs to use machine code sub-routines - all very well. But for machine code freaks, how about a machine code programs calling BASIC subroutines? Or, how about writing a BASIC program which writes a BASIC program and then executes it? (I'm not sure if that's recursion or incest.)

2) The BASIC documentation differentiates between commands and statements, but in fact you can use commands in statements. For example try this program:

```
10 INPUT "DO YOU WANT TO SAVE THIS PROGRAM"; A$
20 IF LEFT$(A$,1) = "Y" THEN RUN 1000
30 END
1000 INPUT "WHAT FILE NAME"; F$
1010 PRINT "START YOUR RECORDER AND WHEN"
1020 INPUT "YOU'RE READY HIT NEWLINE"; A$
1030 CSAVE F$
1040 END
```

I hope that these ideas will be of some use to somebody and look forward to seeing some interesting programs using them. Keep up the good work.

H.E. Gilhespie.  
Bexley Heath, Kent.

Ed. replies:

As the BASIC manual details, the single byte representations of reserved words can also be used when writing programs. e.g. using ? instead of PRINT. There is a list in the BASIC manual of equivalents, but in the INMC's "Basic Programs Book 1" there is a more detailed list.

---

THOUGHT FOR THE DAY  
=====

An optimist is a programmer who writes in ink.

---



# DOCTOR DARK'S DIARY -5

## KEYBOARD REPEATS-AN AWFUL WARNING.

The article on keyboard repeats in INMC news 5 was most welcome, and I have made use of the routine provided in several recent programs of mine. Programs such as the "Road Race" and "Walled Chase" seem to contain very similar routines, for they certainly have keyboard repeat action. Someone I know has made a very significant discovery about the behaviour of these programs when loaded into a Nascom which has had a hardware repeat added.....

There's a user of Nascom in Street,  
Who thought Road Race would be a real treat.  
But it just wouldn't run,  
Which ruined his fun,  
This was caused by his hardware repeat.

There are three morals to be found here:

- 1) Add-ons should have off switches.
- 2) The software way is the cheapest, and often best.
- 3) Poetry should be left to experts!

## MORE ADD-ONS: GLOWING PRAISE.

I have just added a Bits and P.C.s dual monitor board and control keypad to my Nascom 1, so now I can switch between Nasbug T4 and Nas-Sys, and don't have to remember which five keys to press at the same time when I want Nas-Sys to do something fancy. Oh alright, three keys then. Delivery was very rapid, and the person who answers their telephone is very helpful. They did have problems with one of their suppliers however, and it was necessary to wait an extra week for the keycaps.

The dual monitor board is a small single-sided PCB, on which one assembles two normal 24 pin sockets, two of the wire wrap type with long legs, four resistors and a DIL switch. Guess what? The pins of the wire-wrap sockets plug into the sockets where your monitor was, and you plug two monitors in on top. Simple ideas are often the best, and this one has given me a good idea for a modification to their graphics board! (Anyone think of a good name for an add-on to an add-on?)

The control keypad is more complex, and of the same high quality as the dual monitor board. This PCB will also carry the keys of a Hex keypad, if you want one, I just didn't have space in my keyboard box. When it is assembled and connected up as shown in the instructions (which are also clear) the keyboard has ten extra keys, and Nas-Sys copes happily with them. T4 ignores some of them, as does my Microdigital bleep device. I am very satisfied with these additions, and consider the money well spent. My main worry is how long my 3 Amp power supply will be able to cope!

# FEEDBACK AT LAST!

Just when I was beginning to wonder whether anyone was reading my articles, the INMC forwarded a letter to me from R.L.D. Boxwell. Master Boxwell, who is fourteen, has been trying out unknown opcodes, and has found a remarkable total of eighty-one of them, which I shall pass on now. Since (presumably) the Z80 chip is only tested for the instructions that appear in the book, they may not all work on your Nascom. He doesn't count these in his total:

CB30      CB31      CB32      CB33      CB34      CB35      CB36      CB37

(Ok, everyone knows those are shift logical lefts gone wrong, but look at this lot!)

DD24		INC IXH					
DD25		DEC IXH					
DD2C		INC IXL					
DD44	FD44	LD B,IXH	or IYH	DD7C	FD7C	LD A,IXH	or IYH
DD45	FD45	LD B,IXL	or IYL	DD7D	FD7D	LD A,IXL	or IYL
DD4C	FD4C	LD C,IXH	or IYH	DD84	FD84	ADD A,IXH	or IYH
DD4D	FD4D	LD C,IXL	or IYL	DD85	FD85	ADD A,IXL	or IYL
DD54	FD54	LD D,IXH	or IYH	DD8C	FD8C	ADC A,IXH	or IYH
DD55	FD55	LD D,IXL	or IYL	DD8D	FD8D	ADC A,IXL	or IYL
DD5C	FD5C	LD E,IXH	or IYH	DD94	FD94	SUB A,IXH	or IYH
DD5D	FD5D	LD E,IXL	or IYL	DD95	FD95	SUB A,IXL	or IYL
DD60	FD60	LD IXH,B	or IYH	DD9C	FD9C	SBC A,IXH	or IYH
DD61	FD61	LD IXH,C	or IYH	DD9D	FD9D	SBC A,IXL	or IYL
DD62	FD62	LD IXH,D	or IYH	DDA4	FDA4	AND A,IXH	or IYH
DD63	FD63	LD IXH,E	or IYH	DDA5	FDA5	AND A,IXL	or IYL
DD67	FD67	LD IXH,A	or IYH	DDAC	FDAC	XOR A,IXH	or IYH
DD68	FD68	LD IXL,B	or IYL	DDAD	FDAD	XOR A,IXL	or IYL
DD69	FD69	LD IXL,C	or IYL	DDB4	FDB4	OR A,IXH	or IYH
DD6A	FD6A	LD IXL,D	or IYL	DDB5	FDB5	OR A,IXL	or IYL
DD6B	FD6B	LD IXL,E	or IYL	DDBC	FDBC	CP A,IXH	or IYH
DD6F	FD6F	LD IXL,A	or IYL	DDBD	FDBD	CP A,IXL	or IYL

Now there are 75 codes there, all of which must have uses. Then we have:

DD/FD 26 nn	LD IXH,n	or IYH,n
DD/FD 2E nn	LD IXL,n	or IYL,n
DD/FD CB dd 36	SLL(IX+d)	or (IY+d)

The function of the following is not known, other than the fact that they definitely do something:

FD24	FD25	FD2C	FD2D	DD2D	ED4C	ED4E
ED54	ED5C	ED5D	ED64	ED65	ED6B	ED6C
ED6D	ED74	ED75	ED7B	ED7C	ED7D	ED70

(Whoever gets the job of re-typing that lot has my sympathy, and if they get it all right they will also have my undying admiration!) (I'd rather have a drink, and give my NASPEN a rest! -Ed.)

## IS MARVIN A NASCOM 1 OR A NASCOM 2?

I have just added 8K ROM BASIC to Marvin, and have come to the conclusion that I am now a Nascom 2 owner, apart from a small problem with my graphics character set, which should soon be corrected. I had designed a board for the Basic chip to go on, in preference to hacking my memory board about. It didn't quite work, and I am not sure why? If there are any hardware experts reading this, who can tell me what sort of fault would prevent such a board from running a program, but still allow its contents to be tabulated correctly, I should like to hear from them. If the board can be made to work properly, I intend to make the design available to you all, almost free (photo-copying charges only).

In the meantime, if you are thinking of using the EPROM sockets on your memory board for the Basic, and like me do not want to cut the board about, use a 24 pin wire wrap socket to hold the ROM, and bend pins 18, 19, 20 and 21 out sideways. (I hope that you mean the wire wrap socket and not the ROM! -Ed.) Connect these with wire links to the plated through holes nearest to the places mentioned in INMC 5, page 12. Connect a suitable capacitor across pins 12 and 24 of the wire wrap socket, and there you are. (If it gives trouble such as hanging up for no apparent reason, use a National Semiconductors 81LS97 for its buffer.) This does prevent the use of the other three EPROM sockets, I hasten to add.

Sorry there are no useful bits of machine code here this time, but I seem to spend a lot of time either soldering, or wondering what has caused my perfect program to print "?SN Error". Nothing to do with plum wine, honest.....

ROM GRAPHICS SET - See page 27.

From V.P. Lipton

=====

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6																
7																
8																
9																
A																
B																
C																
D																
E																
F																

# Impact Matrix Printer

## HARDWARE REVIEWS

by D. R. Hunt

=====

### NASCOM IMP PRINTER

Printed using an IMP printer.

=====

The Nascom IMP printer is a comparatively cheap dot matrix printer, printing 80 characters per line on plain paper using an imported American mechanism and electronics designed by Nascom. As Nascom do not have an electronic assembly facility, the IMPs are constructed by a sub-contractor in Scotland. The mechanism is designed for bi-directional printing (more about that later), and will accept friction fed unperforated paper up to 8.5" wide, or tractor driven perforated paper up to 9.5" wide.

#### The Mechanism

The mechanism chassis is a trough shaped 16s steel pressing, cadmium plated to avoid corrosion. Although not very rigid without the remainder of the mechanism, the ingenious use of the 12s aluminium platten and the head guide bar as stretchers produces a remarkably stable box like assembly. Paper and ribbon guides do not form a structural part of the chassis, and are simply laid in slots in the main chassis stamping. Two screws through the chassis side-plates secure the head guide bar, and two screws at the rear secure the back of the platten, the front of the platten being supported in slots in the side-plates towards the front. These four screws pull the whole chassis together, trapping the paper and ribbon guides at the same time. By constructing the mechanism in this fashion adjustment of the critical head to platten distance is particularly easy to achieve, and the whole idea leads to a very compact structure.

The 7 needle print head is transported across the paper by a peg which engages with a three turn continuous spiral cut in what appears to be a nylon (or delrin) shaft, approx. 1" in diameter, driven by a stepper motor via a miniature toothed belt. The head transport motor is secured to the left hand side-plate. Head to paper registration is maintained by the precision ground steel guide bar, whilst the back of the head is supported by a pressed steel guide which is part of the chassis pressing. A nylon gear driven take-off at the right hand end of the head spiral shaft drives the ribbon feed.

Paper feed is derived from a second stepper motor mounted on the right hand side-plate, via nylon reduction gearing which drives the aluminium forward friction feed roller directly and the tractor feed via another miniature belt drive. The tractor assembly is a separate part of the mechanism and is secured to the chassis by two large screws. The tractors have levers which friction lock on to the tractor bars, and it is possible to position the tractors at any point across the paper width. Friction paper feed is made possible by a rubber roller which may be engaged with the front friction feed roller by a small lever mounted in front of the ribbon cartridge. The rubber roller is held in place by strong springs and are adequate for most purposes, but have been found to slip when 'interleaved carbon' telex rolls have been used (an unfair test perhaps?).

The ribbon cartridge is about 2" wide, by the full 10" width of the platten. The ribbon is in the form of a continuous loop, and is driven by a small capstan and pinch roller inside the cartridge. Drive to the ribbon is by a small tongue which engages in a slot in the ribbon capstan, and is driven from the head drive spiral. The ribbon cartridge appears to hold an enormous amount of ribbon, probably about 50 yards which should last a few million characters. Don't open one and spill the contents on the floor, it takes hours to wind back. The ribbon cartridge is slightly 'skewed' in its mounting so that the left hand side is lower than the right, this is so the head traverses the ribbon diagonally, thereby avoiding ribbon wastage.

### The Electronics

The electronics are quite a surprise, as they are loosely based on a Nascom, using a Z80 processor, 1K RAM, 2K ROM, PIO and UART (add a video RAM and a keyboard port and you could very well have a Nascom 1). A proper crystal controlled baud rate generator is fitted, and this also provides the system clock. Power transistors driven by the PIO feed the needle solenoids and stepper motors. The whole of the circuitry is mounted on the double sided pcb of the usual Nascom standard, with the main smoothing capacitor mounted on a bracket over the pcb.

All the printer timing functions are software derived and controlled by the CPU which is being used in an interrupt driven mode. Incoming data is stored in the RAM, which has about 950 bytes free as buffer storage. The buffer is organised as a 'first in first out' file and a handshake signal goes active low when the buffer has only ten bytes free. Parity and carriage return, carriage return/line feed options are link selectable, as are the input BAUD rates between 75 and 9600 BAUD. Interestingly, the baud rate generator also outputs a clock signal at 16 times the selected baud rate so that the IMP can drive the sending device UART at speeds higher than may normally be possible.

Internally the IMP's appearance is functional but untidy. The mechanism chassis and the pcb both being mounted on a heavy gauge black anodized aluminium pressing. The connecting cables to mains transformer and function switches are secured by 'double sided-sticky' to the tops of components on the pcb. Whilst the print head leads are allowed to drape across the top of the pcb. Not exactly professional practice, but adequate.

The IMP is cased in an attractive (although slightly too 'bright') blue moulded plastic cover, which whilst flexible appeared robust enough for the purpose. It is unfortunate that Nascom have succeeded in making this very cheap and functional printer look cheap as well. The metalised stick-on plastic strip right across the front, bearing the legend 'Nascom micro IMP' in letters 1.5" high completely ruins the otherwise neat appearance. A small discreet badge would have been a much better idea.

### On Test

On unpacking the IMP, it was immediately apparent that the packing would not be adequate in the hands of the Great British Post Office, and we suggest that the paper roll carrying arms be removed

before shipment. The documentation was accurate, and contained all the information required to get the IMF up and running, with the 100 odd sheets of fanfold perforated paper also provided. The documentation also contained complete software listings of the operating system, and circuit diagrams, although the style of the circuit drawings left a lot to be desired. The documentation also gave a listing to use the handshake with a Nascom, and this was tried and worked. Close examination revealed that the mains fuse fitted (500mA) did not tally with the circuits (250mA), and cases of fuses blowing without reason have been traced to this cause. Internally one screw (on the head guide bar) was found to be only finger tight. Everything else was in order.

A bit of Nascom text was tried, and the result was most acceptable, the character set being well chosen, given the restrictions of the 7 x 7 dot matrix with no descenders. The only confusing characters being the lower case 's' and 'g'. However, it was noted that the printer was only printing in one direction. This seemed odd until it was discovered that that software counts the characters per line (it doesn't count spaces) and if there are more than 40 printable characters on one line, then the printer does not print in the reverse direction. This allows time for the print head to cool down between lines.

Next a bit of Basic listing was tried, and, as the lines were shorter, bidirectional printing occurred on almost all lines. It was then that it was noticed that the characters did not line up in columns, but that the characters printed in the right - left direction were displaced about half a character width to the right. The cause was found to be the right hand limit microswitch. It seems that the software starts counting the character positions as soon as the switch is released by the print head, and as this is a mechanical device with a fiddly and critical adjustment, this was incorrectly set at the factory. It took about ten minutes to set this switch correct (which probably explains why it wasn't done properly in the first place), but the result was well worth it! Perfectly regular columns.

The IMF will take rolls of paper (telex rolls are very cheap), sheets (which are little awkward to feed), and perforated paper (which is trifle more expensive). The results were impressive at all times, and the printer averaged about 50 characters per second, bearing in mind that the print head had to do a full traverse even if only one character was printed on a line. The noise level was acceptable in a domestic environment, although aggravated by a resonant living room table.

At the price, the printer is probably beyond the reach of the pockets of many who would like one, but at the price should make some of the big boys sit up and take notice. The IMF is small, light and mechanically and electrically sound (although a little tweaking won't go amiss), and certainly very good value for money.

As the whole thing is software controlled perhaps we can look forward to software selectable character sizes, and perhaps even graphics. All of which could be achieved with suitable rewriting of the software. Nascom warn users against having a go at the software, and we endorse this, as any mistakes in the software are likely to be very expensive.

---

# Cheap add-ons

WE'VE SEEN IT SO WE KNOW IT EXISTS  
=====

Nascom 1 is two years old about now, and when it was announced, a keyboard cabinet was announced at the same time. Well Nascom 1 is still a firm favourite (although sales of Nascom 2 are going even better), but what ever did become of the keyboard cabinet ?

We don't know why Nascom have been so slow in producing this item, and by now it's probably a bit too late as all those keyboards are glued immovably inside wooden boxes, or balanced on small flower pots or something. Still, we've actually seen it !!! The same attractive blue colour as the Imp, the one piece plastic moulding is pre-cut for a Nascom 1 keyboard and is moulded with 'knock-outs' for the extra keys on a Nascom 2. The top has a rake of about 1 in 6 which is just right for keyboards with sloped keys, and not too much for the earlier flat topped keys. There is no bottom to the box, but this is hardly necessary. The keyboard is fixed from the inside of the case by screws which enter blind moulded projections on the inside, there are no screws visible on the top side.

In a fit of unnatural embarrassment Nascom have announced that the price will be the same as first advertised, and at about 4.00, only a modified cornflakes box comes cheaper.

ECONOGRAPHICS BOARD  
=====

Another late arrival from Nascom is the graphics add-on kit for Nascom 1. This small board adds the Nascom 2 graphics facilities to the Nascom 1.

The card is about 5" by 2" and is intended to be mounted immediately above the existing character generator. The character generator comes out of the main pcb, and along with the PISO, is fitted to the new board which already contains the other chips needed to decode bit 7 which enables the graphics. A few wire links are required under the Nascom 1 pcb, and two 24 pin plugs and a 24 way cable assembly connects the Nascom 1 to the graphics card. The kit is supplied with a plastic bracket which is bolted to the graphics card, and attached to the Nascom board by 'double-sided sticky'. This latter led to doubts as to how long the card would stay in place, but these doubts were dispelled when we tried to remove the card. It wouldn't come off !!!

The graphics set is the same as supplied for the Nascom 2, with 64 'useful' characters and 64 block characters which in conjunction with the 8K Basic form a block graphics set with a resolution of 96 x 48 points. It was interesting to note, that for the first time the little 'robot' characters in the set, actually have feet. This is because a Nascom 1 uses 16 TV lines to display a character, whereas the Nascom 2 only uses 14 lines.

The above items haven't got to your dealer yet, but will do soon. Ask him about prices and availability, or if he doesn't know, have a go at Nascom Sales.

---

# Colour Board

## WILLIAM STUART COLOUR GRAPHICS REVIEW =====

We have received independant reviews of this product from both Mr. P.Sanders and Mr. D.Jay. We reproduce both here as they do complement one another.

### Colour Graphics Review -----

by P.Sanders

Manufacturer: William Stuart System Ltd.

Price: 45.00 inc. vat and p&p.

What it does: Each character block is divided into 4 pixels, either 2x2 or 1x4 (chosen by a link), making either 96x32 or 48x64 pixels on the screen. Each block can be 1 of 8 colours (red, green, yellow, blue, magenta, cyan, white, black) and the whole background can be the same 1 of 8 colours. They are programmed per block as:

```
bit 7      = 0 for normal characters
bit 7      = 1 for graphics
bits 0-3   = 1 bit per pixel
bits 4-6   = 1 to 8 colours
```

Background colours are programmed from 3 bits of an output port.

How it comes: As a kit of 2 pcbs (why not 1 ?). IC sockets for most of the ICs. The 2 pcbs are Colour Modulator and Graphics Generator. Instructions are clear but assume a knowledge of kit building. Two useful subroutines are included, Line plot and Point plot. One demo program as well.

What it's like: It plugs in the aerial socket and connects to the Nascom pcb by soldering wires onto the back of the pcb; these connections break easily. Easy to set up. Colours are good, except for blue (slight ghosting). I didn't like the background colours (hard to read characters on some colours). Resolution too low for a lot of uses, but very good for boxes around messages, bar charts, and for similar uses.

### Review of William Stuart Graphics -----

by D.Jay

I had just fitted my Nascom 1, plus buffer board, plus 8K RAM board, plus psu, plus auto tape load board, into a Vero box and, as you can imagine, there was not much room left inside. THEN I saw the William Stuart advert in the computer mags.

The address of the company was at a place some 20 miles away, so I went along and took a look, no harm in that I thought ! Upon arrival I was greeted with a demo of the system on a 26" colour television. It was quite impressive and when I was shown a kaleidoscope program I was hooked and began to think of how I could cram it into my computer case.



The colour graphics is in the form of a kit. It consists of two boards, one is a colour modulator (originally designed for TV games), and the other a micrographics board. All components including the 12 chips, connecting wires, etc. are included, as are the instructions, programming notes and demo programs.

The kit provides six colours (red, green, yellow, blue, magenta, cyan) plus of course black and white. Each character on the screen is sub-divided into four "pixels". Any combination of pixels may be displayed as any colour, therefore the screen has a resolution of 3072 "cells". Also the background may be selected as any of the colours by interfacing with the PIO.

Construction is very easy, the instructions being quite good. I had no problems although there are 17 connections to the Nascom board and 3 to the PIO socket. This tends to make it look like a "bird's nest", but with a bit of thought it can be neatened considerably. When connected "bread board fashion" the kit worked extremely well, but I did not have enough room in the case to mount the two boards as directed - so, I mounted them vertically by the side of the Nascom 1. There my problems began... Firstly the modulator and PAL encoder board is not screened at all, and picks up the timing pulses from the graphics generation board, resulting in a crosshatch pattern being visible on the screen. This I partially solved by careful adjustment of the modulator trimmer capacitor, and by altering the video drive resistor to lower the black level.

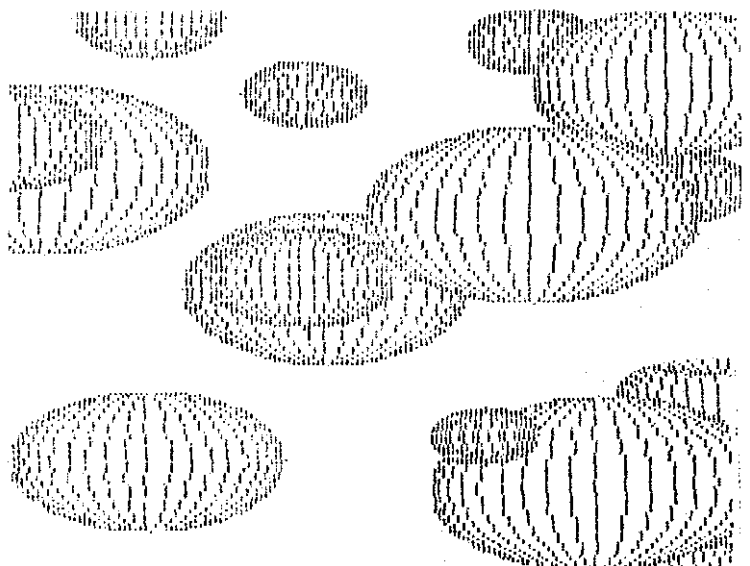
Software has been relatively easy to write in machine code and Basic. Some amazing and beautiful displays are possible.

I would recommend the kit to anyone thinking of CHEAP colour graphics, as it is, in my opinion, excellent value for money. The only disappointment being that the modulator does not give as good definition and clarity as that of a screened modulator such as the Aztec colour module.

One last word; as the software is designed for optimum use of colour and graphics in 4 pixels per character, and Nascom's graphics option being 6 pixels per character, using the SET, RESET and POINT commands in the 8K Basic gives some very peculiar results. However, a SET routine using the USR(0) command is easy enough to write.

#### SPECIAL IMP =====

In our Imp review David Hunt mentioned that by writing your own control software the Imp becomes capable of all sorts of special tasks. Here is an example of some work done by Graham Rounce. The program is called "Super-blots".



## 2 mysteries ~ 1 solved

### MYSTERY PROGRAM FOR NAS-SYS =====

Type in the mystery program (use the 'L' command with the checksums for error free loading), don't contract the NOPs at the beginning, they are important. Execute at 0D00H, and when you get fed up with it, just RESET.

```
T D00 DB1
 0D00 00 00 00 00 00 00 00 00 0D    0D60 6C 65 66 74 20 61 6C 6F 74
 0D08 00 31 00 10 21 00 00 22 99    0D68 6E 65 2E 0D 0D 20 20 57 27
 0D10 7E 0C 21 FF 0F 22 23 0C 27    0D70 68 6F 20 61 72 65 20 79 45
 0D18 3E 0D 32 25 0C EF 0C 20 EE    0D78 6F 75 20 61 6E 79 77 61 A9
 0D20 20 49 20 61 6D 20 74 68 80    0D80 79 3F 0D 00 DF 63 1A FE AC
 0D28 65 20 4E 61 73 63 6F 6D 1B    0D88 25 20 16 13 DF 79 38 11 A4
 0D30 20 44 65 6D 6F 6E 2E 0D 8B    0D90 DF 60 7C B5 20 07 21 00 55
 0D38 20 20 50 6C 65 61 73 65 DF    0D98 00 22 23 0C C7 22 A5 0D 91
 0D40 20 64 6F 6E 27 74 20 74 DD    0DA0 E9 2A A5 0D E9 A7 0D EF FE
 0D48 6F 75 63 68 20 6D 65 2E 24    0DA8 48 65 6C 70 21 20 00 18 97
 0D50 0D 20 20 49 20 6C 69 6B 53    0DB0 F6 00 00 00 00 00 00 00 B3
 0D58 65 20 74 6F 20 62 65 20 D4
```

---

### NASCOM 1 and 8K BASIC -- a bit more =====

In the last two issues we have published details on how to fit the Nascom Basic ROM to a series 1 memory card. We have since learned the a good few hundred are working successfully, but that a couple of dozen did some inexplicable things. We tended to ignore this as 'finger trouble' until last week when one of the committee's Nascom 1 suffered an unfortunate accident. (It was discovered that the PIO input protection is not proof against 240V AC !!!).

As only about three chips were still working, it was decided to re-chip the Nascom from square one. As the Nascom was pretty well fully expanded this was expensive, but all worked well except the Basic which normally wouldn't initialize, or when it did, only worked a few minutes without crashing. We poked around and found the only difference made when re-chipping was that IC9 on the buffer was originally a 74S04 (see memory plague notes in INMC 3), and as we didn't have any spare S04s, a 74LS04 was used instead. Well we never believed that the extra 5nS the S04 gave to the leading edge of MREQ was of any significance, so how about the extra capacitance an S04 imposed on MREQ ?

We tried zapping 150pF across pin 1 of IC9 to ground, and lo !!!, working Basic. Just to prove this wasn't a flock, we got in touch with a dealer we knew had three boards supposedly not working, borrowed them, and without the capacitor the Basic didn't work, with the capacitor it worked.

So if you have a Basic ROM fitted to a Nascom Series 1 memory, and its giving you trouble try 150pF between pin 1 and earth of IC9 on the buffer board. Don't ask us what it's doing, we haven't dared look (something horrible to MREQ we bet), but its worth a try.

---

## BASIC extra & Memory mapping update

### BASIC "PRINT USING" STATEMENT

=====

The Nascom 8K Microsoft Basic does not support any kind of 'Format statement', such as 'PRINT USING', a fact which can be annoying if you wish to produce a column of figures. For example, if you wish to produce a list of items and prices and the prices are say ... 12.00, 6.56, 145.02, 0.32 ... then using the straight forward 'PRINT' command this would appear as:-

	rather than:-
12	12.00
6.56	6.56
145.02	145.02
.32	0.32

One way to obtain the column on the right is to use a subroutine to do the formatting for you. The Basic supports a good set of string functions that makes the implementation of such a subroutine a fairly simple matter. An example of such a routine is given below. The variable A (amount) has to be set to the amount (in pounds) to be printed before the subroutine is called. It returns with the formatted output in A\$. It works for both positive and negative numbers.

```
1000 A$=STR$(INT(A*100+0.5)) : A=LEN(A$)-1
1010 A$=RIGHT$(" "+A$,9)
1020 ON A GOTO 1050,1040
1030 A$=LEFT$(A$,7)+". "+RIGHT$(A$,2) : RETURN
1040 A$=MID$(A$,2,6)+"0."+RIGHT$(A$,2) : RETURN
1050 A$=MID$(A$,3,6)+"0.0"+RIGHT$(A$,1) : RETURN
```

### MEMORY MAPPING - more information

=====

Following the article on memory mapping in the last INMC newsletter (No. 6), here are some brief details of memory usage in the area 0C00H - 0CFFH, which is intended for use as workspace for the operating system software and your own programs if you are desperate.

- 0C00H - 0C7FH NAS-SYS work space. See NAS-SYS manual for details.
- 0C80H - 0CFFH Possible extended system workspace. 0C90H - 0CFFH is used by the repeat keyboard routine which is listed in INMC newsletter No. 6.
- 0D00H - 0DFFH Workspace for possible future system software, is also used for Naspen stack and 'Find string' space.
- 0E00H - 0EFFH Workspace for NAS-DIS and D-BUG, including D-BUG stack.
- 0F00H - 0FFFH Workspace for ZEAP, followed by the user stack (0F80H - 0FFFH).

There is no reason why areas of RAM should not be used for different purposes by different pieces of software. The above usage of 0D00H - 0FFFH is applicable in the machine code/assembler programming environment. Once in Basic, 0D00H - 0F80H is free for you to POKE or DOKE your machine routines into. We recommend that you use 0D00H upwards for these - we doubt you will run out of space since most of these routines are only about 30 bytes long and we have seen one which was two bytes long!

# ZEAP 2.0

## MODIFICATIONS TO ZEAP 2.0

=====

Here are details of some "unofficial" modifications to ZEAP which we have found useful. These cure two minor bugs, and provide a valuable new feature :-

Bug 1: After an assembly, the margin to the right of the bottom line is corrupt, which enables the cursor to be moved off the screen, and can cause the top line to move when editing the bottom line. This is cured.

Bug 2: When using the H command to set the number of lines, and then repeatedly using the Z command for editing, it becomes necessary to press a key to display the line to be edited. This is cured.

Additional feature: Type a colon (:) and press ENTER. The message "Command?" is displayed, and the system waits for you to enter a NAS-SYS command, which is then obeyed. On completion of this command, control remains within ZEAP. This makes it very easy to use the Read and Write commands to save the source file or object program. It can also be useful to use the Tabulate and Modify commands to examine memory locations. To provide this feature it has been necessary to remove the Q command from ZEAP, but this command has been superseded by the J and K commands.

## MODIFICATIONS TO ZEAP 2.0 RAM VERSION

=====

```
1057 3A
1058 CC
1059 1F
1493 FF
182B 00
182C 20
1BB0 CD
1BBE B5
1BBF 1F
1BE9 CD
1BEA C2
1BEB 1F

1FB0 1D C3 DD 1D A3 32 FE 0F
1FB8 21 BA 0B 06 10 77 23 10
1FC0 FC C9 32 FF 0F 3A 14 0F
1FC8 32 77 0F C9 EF 43 6F 6D
1FD0 6D 61 6E 64 3F 0D 00 DF
1FDB 63 01 2B 0C 1A FE 20 C8
1FE0 FE 41 3B 0D FE 5B 30 09
1FEB 02 32 0A 0C 13 DF 79 30
1FF0 03 DF 6B C9 DF 60 DF 5C
1FFB C9 00 00 00 00 00 12
```

## MODIFICATIONS TO ZEAP 2.0 ROM VERSION

=====

```
D057 3A
D058 CC
D059 DF
D814 00
D815 E0
DBA6 CD
DBA7 B5
DBA8 DF
DBC F CD
DBD0 C2
DBD1 DF

DF98 C3 DD 13 00 00 00 00
DFA0 00 00 00 00 00 00 00
DFAB 00 00 00 00 00 00 00
DFB0 00 00 00 00 00 32 FE 0F
DFB8 21 BA 0B 06 10 77 23 10
DFC0 FC C9 32 FF 0F 3A 14 0F
DFC8 32 77 0F C9 EF 43 6F 6D
DFD0 6D 61 6E 64 3F 0D 00 DF
DFDB 63 01 2B 0C 1A FE 20 C8
DFE0 FE 41 3B 0D FE 5B 30 09
DFEB 02 32 0A 0C 13 DF 79 30
DFF0 03 DF 6B C9 DF 60 DF 5C
DFFB C9 00 00 00 00 00 00
```

# Get it RIGHT !

## NAS-SYS NAUGHTIES

Over the last few weeks we have become aware of an appalling ignorance about NAS-SYS, not only by users like you and me (who should try reading the manuals even if they are hard to understand at times), but by people who hope to sell software products for Nascom computers. For instance, we were sent the drafts of ammendments to a book, which incorporated an update for use with NAS-SYS. It said in effect, 'Throughout the book, whenever you see a reference to one of the following, change it thus:

NASBUG		NAS-SYS
CD 3E 00 (CHIN)	=	CD 08 00
CD 3B 01 (CRT)	=	CD 4F 01
CD 44 02 (B2HEX)	=	CD 19 03
CD 32 02 (TBCD3)	=	CD 00 03
etc.'		

Now this is wrong, WRONG, WRONG !!!!! And one of them is doubly WRONG. If you can't see anything wrong with that, then you should reread the manuals (go on, grin, but we bet there are a few red faces out there). The author has missed the whole point about NAS-SYS. This is not the only instance, we'll be giving a list of known 'Goodies' and 'Baddies' at the end.

Now one of the major features of NAS-SYS is the fact that to remain compatible with special versions, and, may be, later revisions, all calls to internal routines are handled through a table of addresses. This allows the software writers freedom to re-assemble NAS-SYS as required, yet still maintain compatibility with software written for an existing version. It is even possible to turn NAS-SYS inside-out and for the using software to be unaware of it.

Here are the rules, they are simple enough:

- 1) Never ever make absolute calls or jumps of any kind to NAS-SYS.
- 2) The major routines use the Z80 restarts, use them properly.
- 3) Always use the restarts or SCAL (RST 18H) to gain access to all NAS-SYS routines. STMON is the ONLY exception.
- 4) If calling routine 'R' using SCAL, make sure you put 52H into ARGX and that ARGN is set to 00H before the call.
- 5) Only use routine call numbers 41H to 7CH for full compatibility.
- 6) If in doubt read the listings and the manuals to understand how the SCAL routine works.

There, that wasn't difficult was it.

One point, if you are writing software which makes monitor calls, and you don't know what monitor it is to be used with, make all your CALLs to a table of your own, using three bytes for each call. Then, to give an example, if you want to call CHIN in NASBUG, the three bytes will be JP CHIN:

C3 3E 00

Whereas in NAS-SYS two of the three bytes will become RST RIN, RET, the last is a 'don't care' byte:

CF C9 XX

Or in another instance, using B2HEX, the three bytes become JP B2HEX:

C3 44 02

in NASBUG, whilst the NAS-SYS equivalent is RST SCAL, B2HEX, RET:

DF 68 C9

A nice example of this approach is XTAL BASIC, which although it appears in our list of 'Baddies', uses just this approach and so, although it makes absolute calls to NAS-SYS, it is very simply corrected (see the review in this issue).

Now for our list of 'Goodies' and 'Baddies'. Richard wrote a special NAS-SYS with all the addresses changed, so we are sure the following do or do not work with other NAS-SYS's (NAS-SYSES, NAS-SYSii ??).

Goodies	Baddies
Nascom ROM and Tape Basic	Xtal Basic (but see the review)
ZEAP 2	Nas chess (to be reviewed next issue)
Nas Pen	Memory Tests published in Liverpool
Nas-Dis	Software Gazette No. 3
D-Bug	

We know of other bits of software coming but haven't yet been able to lay our 'sticky mits' on them, so we don't know if they obey the rules or not. So if you are writing software with a view to selling it, DON'T FORGET THE RULES !! Better still, send us a copy, we'll review it, and tell you if any changes are necessary.

All this comes about by failing to understand the manuals, and Nascom must accept some of the blame, there are times when a thorough knowledge of cryptography (and a magnifying glass) would be useful in deciphering some parts of them. Here are some genuine 'howlers' that we have heard uttered by users recently.

Nascom 1 & 2 with 8K Basic:

"Isn't a pity that Nascom Basic always LISTs from one end to the other without being able to break it 'mid-stream'."

Answer:

Use the 'ESC' (shift newline) once to stop the list at any point (any other key will restart the LISTing), and 'ESC' again to break the list. The same applies whilst running a program.

NAS-SYS:

"How on earth did you open up that line and insert those extra characters?"

Answer:

Shift cursor right and shift cursor left keys will open and close a line.

"I didn't realise that you could make the Tabulate command tab in blocks of say ten, nor that you could escape halfway through a tab."

Answer:

Try T 0 FFFF A, newline, then hit any key a few times followed by an 'ESC'.

Anything except NASBUG T2:

"I'm a typist, and I'm not used to computer keyboards. I find normal lower case is better for me, can I change my keyboard for a lower case one please?"

Answer:

Type K1 then carry on. That will cost you the 50.00 I've just saved you on a new keyboard, for the information please. Ta !!

---

## The End of 'PLAGUE' & a NEW Board.

### THE FINAL DEFINITIVE MEMORY PLAGUE NOTE

=====

In the last issue we promised that we would collect together all thoughts about 'Memory Plague', and publish them. We are indebted to many members of the IMNC for writing to us detailing various symptoms and possible causes. For more recent members of the INMC let us first define 'Memory Plague': this was a euphemism coined to cover the small percentage of Nascom Series 1 memory cards (that's the one with the four EPROM sockets) that proved to be unreliable through causes not otherwise due to faulty or slow chips. 'Plague' soon became apparent after the release of the Series 1 card, and the symptoms are unreliability when working machine code programs in the expansion memory. 'Plague' is not usually revealed by Tiny Basic or the memory test programs as these represent data stored in memory and not op-code. 'Plague' usually affects the op-code fetch (M1) cycle as the timing is more critical, and results in the misreading of the op-code byte, causing programs to crash. The probable cause is noise generated on the Nascom 1 busses, although this has never been conclusively proved. A number of 'cures' were published, which consisted of basically three things:

- 1) Griding the back of the memory card to reduce power supply noise.
- 2) Pulling up the outputs of the RAMs to increase operational speed.
- 3) Adding a time constant in the form of a capacitor to the RAM pullups to damp any noise on the output buffer.

On a Nascom 1 these work. On a Nascom 2, step 3 represents overkill, and can cause poor operation at 4MHz, the time constant being calculated for 2MHz.

Over that intervening period other possible causes have come to light. One is the 33R damping resistors in series with the address and CAS lines are too low to be properly effective. Another possible is the DBDR signal coming off too early. Yet another is the MREQ signal (applied to IC31) arriving too late, related to the system clock which latches it.

With the advent of Nascom 2, which is buffered on board, bus noise has been much reduced, and plague is almost nonexistent. However, Nascom 2 has revealed that the memory card will not run reliably at 4MHz without 'WAIT states'.

We therefore now recommend the following:

#### Nascom 1

- 1) For each row of logic chips, grid the back of the pcb, connecting the +5 volt rail from the termination on the logic half of the pcb to the +5 volt termination of the 100n decoupling capacitor immediately opposite. For each row of logic chips connect the 0 volt rail likewise; where a row is adjacent to pin 12 of each of the EPROM sockets, incorporate this into the 0 volt grid.
- 2) Change R7 - 14 and R17 to 68R, and RAS links LK1 and LK2 should also be 68R resistors.
- 3) Where the Basic ROM is fitted, add a 150pF capacitor from pin 1 to 0 volts of IC9 on the buffer board, this will slightly increase MREQ, and thereby increase DBDR.

#### Nascom 2

- 1) Try the board and see if it runs Basic reliably. If so go to step 3. Don't bother to investigate further unless the board reliability is suspect.
- 2) Grid the back of the pcb and change the resistors as for Nascom 1.
- 3) If you want the board to run at 4MHz without 'WAIT states':
  - a) Cut the clock line from the bus connector, 5, to pin 3 of IC31.
  - b) Connect a wire link from the clock side of the cut track to pin 9 of IC35. Connect a 1K pullup between pins 8 and 14 of IC35.
  - c) Connect a wire link from pin 8 of IC35 to pin 9 of IC34. Connect a wire link from pin 8 of IC34 to pin 3 of IC31.

A number of boards have been fitted with the above, and all have behaved perfectly. If you have a board which has been fitted with the earlier Nascom 1 mods, and is working correctly, then leave well alone. If for some reason you wish to incorporate the above, then remove any earlier mods first. All this brings us to another review.

#### NASCOM RAM B

=====

Nascom RAM B is part of the new System 80 and is a RAM card designed to replace the Nascom Series 1 Memory card. It incorporates a number of novel features, and as far as we can tell has been fully debugged. The RAM is organised as three blocks of 16K, using the now popular 4116, 16K x 1 dynamic RAM. This gives the board a total capacity of 48K, there being no onboard EPROM sockets, unlike Series 1. The RAM is fully buffered in and out, and active delay lines are used to get the timing 'just so'. There should be no problems of the sort experienced with Series 1.

The novel part is in the way the board is controlled. When purchased, the RAM is just like any other RAM with a maximum capacity of 48K (you can always buy the 16K version and fill the rest up later). However, an option pack will be available which when added to the RAM gives two important features:

- 1) Selectable 'WRITE protect' on each of the 16K blocks, controlled by miniature switches mounted on the front of the pcb.
- 2) The ability to turn the RAM on and off under software control in one of four 'fields' (or blocks, or whatever) by the use of a port. This means that up to four cards could be co-resident at the same address, and controlling software could select each as required. This gives the Nascom system a total of 208K of addressable memory (4 x 48K + the minimum system). Nascom claim this as a 'first' and it certainly seems a very clever idea, although we wonder what you could do with all that RAM if taken to extremes. Certainly faster than disc, but what do you put in it?

We haven't played with a production version yet, but the production prototype worked faultlessly (4MHz and no 'WAITs'). We understand the price will be about the same as the Series 1 RAM.

---



MORE RAM and a B-BUG note  
RAM IN THE SECOND 1K  
=====

by G. Alabaster

Minimum Nascom 1 with Nasbug T2 ? Then you NEED more memory and have an empty EPROM socket.

Well, there is a nice static RAM chip, the MK4118, which is pin compatible with the 2708 EPROM. I should say ALMOST pin compatible. This is a way to put the chip in, leaving the board and socket unadulterated in case you wish to put an EPROM back there.

Here are the pin out differences:

Pin No.	2708	4118
18	Program	CS
19	VDD (+12V)	L (RD)
20	CS	OE (RD)
21	VBB (-5V)	WE (WR)

The pins on the MK4118 would object to being bent out, so put a socket with flexible pins between the RAM and board socket (See Dr. Dark's Page - Ed.). Bend out pins 18, 19, 20, 21. Connect leads to these pins and connect to the printed circuit as required. The points I used were these:

Pin 18 to E select line which has a through plated hole near pins 12 and 13 of IC36.

Pins 19 and 20 connected together to the inverted Read Data line which has a through plated hole just out from pin 9 of IC44.

Pin 21 to the inverted Write Data line which has a through plated hole near IC50 going to pin 3 of IC50.

As an extra safety measure, put insulating tape over the board socket holes for pins 18 to 21 and double check the voltages at the intermediate socket without the RAM in.

At the time of writing there is a worldwide shortage of MK4118s, but this should ease off once the backlog of NM's order for Nascom 2 requirements has been met. Ring around the suppliers, you could be lucky as you only need one. With time the price should drop quite a lot from the present twenty pound level.

---

B-BUG & 8K BASIC NOTE  
=====

Mr. F. Whitby wrote in to inform us that when using the 8K BASIC (or any other program), you get "Escape" by holding down "@", then pressing "Shift" and holding it down, then pressing "3".

All other control keys are different with B-BUG, as the B-BUG manual explains.

---

# Moonlanding

FIFFLOSO hits the lunar surface  
=====

by C.Webster

When the NASCOM users' newsletter announced its competition for an entertaining program to run on a 1Kbyte machine it seemed immediately apparent to two certain people that an entry developed on their machine should most certainly be submitted. Sadly, though, they were revising at the time for finals (congratulations, incidentally, to them both) and had not the time required to squeeze any of their amazing programs into the requisite tiny space. Not for nothing, though, had they spent so much time in the company of Arthur Norman (see previous articles in DATABUS for samples of his work!) - with a pair of malicious grins almost worthy even of him, they remarked "Arthur and Chris can write some code for it".

This suggestion did not fall on entirely stony ground. Both Arthur and I have a fairly long record as hackers of the PDP-7 and he has additional vast experience of entertaining software on many assorted machines. We both felt that the time had arrived to get fairly close to the machine code of a modern microprocessor, and what better machine could there be than the Z80? Furthermore, what could be a more amenable introduction than the challenge of writing a fun program to meet a deadline?

I must admit that we were both quite strongly taken aback to discover that the aforementioned deadline was less than two weeks away, but with mutual cries of encouragement we nerved ourselves for the fray, carrying the poor innocent NASCOM off to the fifth floor of the Computer Laboratory Tower and denoting ourselves the FIFth FLOor Software company.

One of the factors influencing our choice of program to write was the shortage of available time, not only to implement it, but even just to decide what it should be. Some of the considerations influencing our choice of a moonlander were

- 1) We each had some experience of looking at programs for this, respectively on the PDP-7 and the GT40.
- 2) It seemed to us that the problem could be expanded or shrunk more or less at will, and according to how well we were getting on with it.
- 3) The majority of moonlanders are BAD and BORING and we wanted to write a half way competent one.

Having determined this fundamental point, we turned our attention to the machine itself. A number of differences in philosophy from the machines with which we were familiar brought themselves forcibly to our attention. Probably the most significant of these was the bewildering variety of available instructions: the PDP-7, for example, has only a four bit opcode in its eighteen bit word, and there are no prizes for guessing how many instructions that gives it. By comparison with this, eight bits of Z80 opcodes, most of the combinations of which are legal, appeared to mean a great many more possibilities to consider.

Offset against this, we had occurrences of the bewildering variety of the

SAME instruction, if I may express the concept in such an opaque manner. Serious contemplation of the amazingly different functions represented by the one assembler mnemonic LD almost gave one cause to doubt one's own sanity:

```
LD A,B
LD E,D
LD (HL),A and even
LD A,(HL)
were sort of comprehensible, but do we have,
LD HL,BC
and if not, why not?
```

The ability to manipulate literal data provided another startling contrast with most older machines. Being able to go

```
LD A,12
makes for a machine with a very friendly feel, and makes the dreaded
NOVA for which, even at assembler level, one would have to write
something like
LDA 0,M12
```

```

.
.
.
M12: 12
```

look really pretty silly by comparison.

Of the other points that made a strong initial impression, the only remaining one really worthy of mention seems to be the stack: those who have written code to organise stacks for themselves must surely be appreciative, as we were, of this great facility being provided in the hardware, though I do admit that this is hardly exceptional in modern architecture.

So much for the machine. It has been asserted that "All machine codes are the same when you've worked out what the registers do", and with this thought (and a bottle of port purchased from one of the more respectable Cambridge Colleges) to sustain us, we set about our first night of Z80 hacking in tolerably high spirits.

It is amazing what optimism and alcohol can achieve between them. By the time that we adjourned at about 3.00 a.m., there were clearly two pieces of code which were approaching one another with every appearance of being on the brink of joining together and making something approaching a program.

Panic mounted during the succeeding days as the deadline approached nearer. Lunchtimes and precious weekend hours were all sacrificed to the cause, but by about day 5 one intrepid pioneer had made a controlled landing, to general rejoicing among the onlookers.

All conceivable arms were twisted to provide manpower for the project ("Finals? You don't need to revise for finals"). All praise is due to those who originally landed us in the frying pan for preventing our further descent into the fire. The program, however, got written.

Whether or not it is a great piece of software, and whether or not it can win a competition (It did! -Ed.), the conclusion to this narrative,

gentle reader, is as follows: one can learn about Z80 code in one week to write an entertaining and non-trifling program, and that makes the Z80 a good thing by anybody's standards.

Chris Webster  
(c) FIFFLOSO 1979.

This article originally appeared in the September 1979 issue of DATABUS, the Cambridge University Processor Group magazine, and has been reproduced here by kind permission of the author, Chris Webster, and the DATABUS Editor, Philip Gladstone.

## VDU

### PLANNING YOUR DISPLAY

By V.P. Lipton

1. The chart below has the decimal equivalents of the Hex addresses of the screens for Nascom 1 & 2. When using Basic, one has to POKE data into the decimalised addresses on the screen, so this chart will help all you Basic people to get your little men into the right positions easily.

(Ed.'s Note: Any position on the screen can be calculated by the Basic using the equation  $P = 1993 + X + 64 \times Y$ , where X is the column (1-48), Y the line (1-16), and P the decimal address of the desired location.)

2. On page 10 is a listing of the Nascom Graphics ROM. If a lot of key-bashers have been scratching their heads trying to get their little men to appear, then this ammended chart is sure to relieve their dandruff, as the chart printed in the original Nascom 2 handbook was far from useful!

1	3018	3019	etc...	3042		3065	TOP ROW(uncrolled)
2	2058	2059	etc...	2082		2105	2
3	2122	2123	etc...	2146		2169	3
4	2186	2187	etc...	2210		2233	4
5	2250	2251	etc...	2274		2297	5
6	2314	2315	etc...	2338		2361	6
7	2378	2379	etc...	2402		2425	7
8	2442	2443	etc...	2466		2489	8
9	2506	2507	etc...	2530		2553	9
10	2570	2571	etc...	2594		2617	10
11	2634	2635	etc...	2658		2681	11
12	2698	2699	etc...	2722		2745	12
13	2762	2763	etc...	2786		2809	13
14	2826	2827	etc...	2850		2873	14
15	2890	2891	etc...	2914		2937	15
16	2954	2955	etc...	2978		3001	16

MID SCREEN

L.H. Edge R.H. Edge

-----48 LOCATIONS-----

## SOFTWARE NAS-DIS/DEBUG/XTAL

### SOFTWARE REVIEWS =====

by R. Beal

Two new items of software are about to be released and these have been reviewed by the INMC committee to see how good they are.

#### NAS-DIS 1.0 =====

NAS-DIS is a disassembler, which means that it takes any machine code program and produces an assembler listing from it. This source code, if assembled, is certain to generate the original machine code program again. Disassemblers are very useful in examining machine code programs so that you can work out what they do and make any changes required.

NAS-DIS is a direct descendent of REVAS, and has been written by the same author, David Parkinson. However, it is a vast improvement over the original, and this review can only describe some of its many features. It comes in two versions, either on a tape which can be used to produce a copy of NAS-DIS located at whatever address you choose, or as three EPROMs located at C400H - CFFFH. It is 3K long and runs only under NAS-SYS.

It is most convenient on the Nascom 2 to have a separate memory board for RAM, and to use the eight sockets on the CPU board for 2708 EPROMs. In this case, four can be used for ZEAP 2 (Nascom assembler), and three for NAS-DIS. This makes sense, as NAS-DIS is fully compatible with ZEAP, which means the output of NAS-DIS can be fed into ZEAP without alteration. In fact the assembler output from NAS-DIS can be generated in several ways. It can be simply listed on the screen, or printed, or output to tape, or can even automatically become a ZEAP source file in memory. Printed output can be split into numbered pages with titles, and the tape output is also compatible with ZEAP so that it can be fed directly back into ZEAP.

An option normally selected is the automatic generation of labels. All locations addressed within the program are given labels, making it easy to read the code. Furthermore, a complete cross reference table can be output, showing every address where each label is referred to. This is very valuable when tracing backwards through the logic of a program.

It is also possible to specify that some areas of memory are data areas, so that these are not converted to spurious assembler codes. In this case DEFB instructions appear.

NAS-DIS can, optionally, take account of NAS-SYS restart features, namely SCAL, RCAL and Print String (PRS). With SCAL and RCAL, the next byte is output as a DEFB. With PRS, NAS-DIS generates as many DEFB lines as are needed for the message.

If you don't want to spend any time thinking which options to use, you can simply execute NAS-DIS and specify the start address to disassemble at. This means that any piece of code can be disassembled almost instantly.

Other options in brief:

- Disassemble a program which is not at its normal execution address.
- Change lines per page and page size.
- Display the output from only part of a disassembly of a large program
- NAS-DIS incorporates a subroutine called REVAS (!) which you could call from your own program.
- Good validation of user input when selecting options.

We have tested NAS-DIS and can verify that it disassembles all machine code correctly, with no nasty quirks. It is an excellent piece of software.

NAS-DIS is available from Nascom through your dealers. Ask your dealer about it. If he doesn't know anything about it, nag Nascom Sales Department for prices and ask them why it isn't available.

#### SUPER DEBUG =====

SUPER DEBUG (we'll call D-BUG) is a 1K program which runs under NAS-SYS. In order to work, it requires that NAS-DIS also be installed. D-BUG lives at C000H - C3FFH, D-BUG can therefore fill the eighth 2708 socket on the Nascom 2 CPU board, so that with NAS-DIS the whole becomes a 4K package.

The aim of D-BUG is to make it easier to examine the workings of other programs. The principle feature is a comprehensive and labelled display of all the CPU registers, as well as the eight bytes pointed to by each of the main registers. The example below gives an idea of the display:

```

SP 1000          FF 00 FF 00 FF 00
IX FFFF DE FD C3 B1 E0 31 00 10 D7 08   IFF2 0
IY FFFF DE FD C3 B1 E0 31 00 10 D7 08   I B0
HL 0000 FD C3 B1 E0 31 00 10 D7 08 C3   HL'FFFF
DE 0000 FD C3 B1 E0 31 00 10 D7 08 C3   DE'FFFF
BC 0000 FD C3 B1 E0 31 00 10 D7 08 C3   BC'FFFF
AF 0000          ↑   AF'FFFF
PC 0000 31 00 10   LD      SP,#1000   ;1..

```

As you can see, the next instruction to be executed has also been disassembled. This means that you can single step through a program and read each instruction in assembler at the same time. This also works with breakpoints.

D-BUG uses NAS-SYS editing in an unusual and advanced way. The cursor can be moved up and the lines of display edited, one line at a time. The memory locations pointed at by the registers can be altered, and the values of SP, IX, IY, HL, DE, BC, AF, and PC can also be altered. If the code after PC is altered, the disassembled source code appears at once. The values of IFF2, I, and the alternate registers cannot be directly altered in this way. A character representation of the flags is output when the flags are set, and these can be altered by editing the F register.

D-BUG has several commands. These are entered by typing a colon followed by the command letter. An excellent feature is the alternate video RAM command, :0. This specifies a 1K block of RAM which is spare in your system. It can be swapped with the NAS-SYS/D-BUG display by using the command :A, pressing Enter returns the display to normal. The two pages of the display are automatically swapped over during execution (E or S commands), so that use of the screen for debugging does not affect the display used by the program. This is invaluable for testing programs which output information to the display and read it back. (Don't forget to clear the alternate display before running your program! Use :A then clear the screen).

D-BUG works by trapping all output via the UOUT jump, Since this means the the UOUT jump has been used up and is no longer available, for example, for driving a parallel printer, D-BUG provides a command (:C) to set the address of your own U output routine.

A further feature is the :F command, which is used to find a string of up to eight consecutive bytes. Even more, one or more of the bytes may be specified as 'wild', for instance, if D-BUG were asked to Find 2A - 0C (the 'dash' means 'wild'), D-BUG would find every location where HL was loaded from a location between 0C00H and 0CFFH.

The only slight problem you might encounter is if D-BUG is activated while running a program which outputs a colon as the first character of a line. D-BUG instantly grabs control and obeys the "command"!

D-BUG is activated by typing the command E C009 and can be de-activated by the N command. A Nascom 2 can be set to power up at C000H, and in this case D-BUG is activated on power up or on pressing Reset. It is necessary to re-activate D-BUG after using ZEAP.

Remember, you must have NAS-DIS and NAS-SYS in your system before you can use D-BUG. We have tried D-BUG thoroughly, and have found it to work well, we recommend it as a 'must' for all machine code addicts.

D-BUG was written for Nascom by Mick Scutt of CC SOFT, and is available from Nascom through your local dealer. Contact your dealer for details, and again if he doesn't know anything about it, have a go at Nascom Sales Department for prices and availability.

#### XTAL BASIC V2.2 =====

Some time ago Crystal Electronics announced an 8K Basic for Nascom 1. They have not stopped at that but have continued work on improving it, and recently announced a completely new version Basic 2.2, which is for Nascom 1 or 2 using T2, B-BUG, T4 or NAS-SYS monitors. Since it has several major improvements over the previous versions, we have decided to review it and compare it with the Nascom 8K Basic.

XTAL is a standard floating point 8K Basic, although in fact it is only about 7.25K long. Crystal jokingly describe it as a 16K Basic

with 9K left spare, and while it can't compare with a good 16K Basic, it is a lot better than the arbitrary standard '8K Basic'.

#### Compatibility

=====

XTAL Basic uses the minimum number of calls to monitor routines, and all these are made through a short table of jump instructions. This means that these are easily changed to cope with any monitor changes. The tape is issued with a NASBUG version recorded in Nascom 1 format on one side, and a NAS-SYS version recorded in Nascom 2 format on the other. However the manual describes how to change one version to the other, and this would only take a few minutes, and the changed version stored on your own tape. One criticism here is that the NAS-SYS version does not follow the NAS-SYS 'rules', and uses absolute jumps instead of the NAS-SYS restart instructions. However, this is easily corrected as shown in the tabulation of the jump table (below) which starts at 2BDDH. The tabulation shows the correct values for NAS-SYS, and ensures that upward compatibility is maintained.

#### T2BD8 2C00

```
2BD8 70 C0 24 18 F6 E5 DF 4E
2BE0 E1 C9 00 00 00 DF 5B C9
2BE8 00 00 00 FF C9 00 DF 6F
2BF0 C9 DF 66 C9 F7 C9 00 DF
2BF8 62 C9 CF C9 00 DF 5F C9
```

#### Comparison of commands

=====

XTAL Basic has the following extra commands:

- EDIT** This allows a program to be edited line by line. Characters can be changed, inserted and deleted. Users of 'SUPER TINY BASIC' will notice a resemblance to the MCE comand. The XTAL Basic does not support the convenient NAS-SYS editing, although this method does have the advantage of allowing long lines (about two lines on the screen). The cursor movement keys are not used in the same ways as in NAS-SYS or Nascom Basic. Also CS is used instead of ESC.
- CSAVE & CLOAD** are similar to the Nascom Basic but allow names of up to six characters. The CLOAD routine stops when the first error is encountered, so your cassette recorder must work well if you have long programs.
- PRINT @ X,Y,"ABC"** prints ABC at the X,Y coordinate. This is similar to using the Nascom Basic SCREEN command, followed by a PRINT command.
- POP** Deletes the return of a GOSUB, so that an abnormal exit can easily be made from a subroutine.
- SPEED up.** Sets the output speed to allow the Basic display to be slowed up.



- PI        This gives the value of PI
- INCH     This waits for an input character. This is extremely useful in many programs.
- KBD      Checks for an input character and returns the value 0 if there was no input. This makes it easy to write games programs where fast reactions are required, and has many other uses.
- ON ERR GOTO or GOSUB This is a remarkable addition to the Basic, and allows any error during execution to be trapped and the appropriate action taken. The variable ERR supplies the number of the error, and the CMD\$ function gives a full text description of the error.

The XTAL Basic has another astonishing feature which is extremely useful. You can add your own commands to the Basic, specifying the name of the command, and the address at which you have stored the machine code. This means that it would be easy for those familiar with machine code to add the Nascom Basic commands which have been omitted from the XTAL Basic. In fact the manual supplied gives the code for the addition of DEEK and DOKE commands as an example. Another feature is the backspace key allows one Basic statement to be executed at a time, which is useful for debugging.

The XTAL Basic does not have the Nascom Basic WIDTH command, which surprisingly, is an improvement, as the Nascom Basic command has the annoying habit of putting out carriage returns when you don't want them. DEEK, DOKE, and the graphics commands, SET, RESET and POINT are also missing, but with the ability to extend the command table, could easily be added.

#### Summary =====

The XTAL Basic is a good Basic with several useful enhancements to the arbitrary standard '8K Basic'. It is not compatible with Nascom Basic, but the missing commands could be added. It would be better if it used the NAS-SYS editing, but still had its own EDIT command, which had advantages for long lines, using the cursor movement keys in the standard way.

#### Cost and availability =====

If you already have version 2.1 XTAL Basic, send your tape back with 50p, and Crystal will send you the new version. Otherwise, XTAL Basic is available for 35.00 + VAT from :  
Crystal Electronics,  
40, Magdalene Road,  
Torquay, Devon.  
(Tel. 0803 22699)

---

## Book Review

BOOK REVIEW  
=====

by D. R. Hunt

Z80 Instant Programs      J. Hopton      Sigma Technical Press.

This paperback volume of 180 pages is published by the Sigma Technical Press and costs 7.50. Sigma Technical Press, it seems, is no relation to Sigma Accounting, the authors of ZEAP, the Nascom Z80 assembler. The book sets out to describe a number of Z80 based machine code programs, many with little or no system monitor dependence. The summary says that there are 36 programs listed, but having read the volume I imagined there were more. The Programs cover most aspects of machine code programming relevant to the home user, ranging from simple delay loops, through simple arithmetic, to simple text handling routines.

The book is primarily based on a Nascom 1 fitted with a NASBUG T2 monitor, and an appendix of the main monitor routines is given, the implication being, that users of other Z80 based systems adapt the programs given in the book to their system. Unfortunately, for instance, the location of the screen memory (0800H - BFFH) is implicit within the monitor routine, and therefore unlikely to be of help to anyone not owning a Nascom fitted with NASBUG T2. Likewise, the keyboard routine given is the routine for the Nascom software scanned keyboard, and as Nascom is the only company using this particular keyboard, it is not likely to be of much use to other Z80 system users. These routines were however beautifully commented, in a particularly easy to follow fashion, so whilst not much use to anyone other than a Nascom owner, they provide a useful insight into the workings of these monitor routines.

It is difficult to tell who the book is aimed at, as it certainly is not at 'beginner' level, there being nothing but the sketchiest explanation of what the various registers are supposed to do or how the Z80 works. No explanation is given regarding memory mapping or how a certain area of the Nascom memory is set aside for the screen, etc. There was a very simplified explanation of what an op-code is, but the word 'operand' did not seem to figure at all. To the more experienced programmer, a lot of the detail in the explanations would be dismissed as trivia. Although the programs were adequately explained I feel the beginner would be left with the feeling that the machine code instructions themselves were even more like black magic than when he started.

Some interesting hardware examples were given, driving LEDs from the ports, D - A and A - D converters, attaching a speaker to play a music program, etc. Although these were not tried, there is every reason to believe that they would work.

Many of the programs are long, and as an aid to correct loading, a tape is available from the publishers. The tape is in Nascom 1 tape format and is unsuitable for Nascom 2. Be warned, the first few minutes of the tape are a spoken commentary, and of course, will not load into the Nascom. Although I did not have the tape for the review, those programs I typed in and tried worked.

Many of the programs themselves were amazing (in the sense that I was amazed). Although programming techniques are very much a matter of opinion, and I am the first to admit that I am not a very clever programmer, some of the programs were mind boggling for their tedium and longwindedness. In many instances a pointer (HL) was set to a screen location, the C register loaded with the character to be displayed, a LD (HL), C instruction performed, HL was then incremented by one and the operation repeated for each character of the string. Occasionally spurious NOPs would be thrown in for good measure. This produces an overhead of at least 3 bytes per character displayed. I wonder if Mr. Hopton has forgotten that the Z80 has an LDIR instruction which only produces a total overhead of 11 bytes regardless of the length of the string. Or as the routine PRS was included in the appendix, why was this not used as the total overhead would then be only two bytes. A couple of programs are given to display geometric figures, which work by setting HL to each location in turn, then using what seems to be the author's favourite instruction, LD (HL), C, placing an asterisc at the location pointed to by HL. No attempt is made to calculate the HL locations within a loop. Whilst this style of programming cannot be condemned as bad programming, programming being such a matter of opinion, it is at least inelegant and inefficient.

Perhaps the most infuriating thing about the whole book is the total omission of instruction mnemonics and labelled routines within NASBUG, which time and again had me thumbing through the pages of my Z80 reference book to disassemble instruction codes that were unfamiliar, or the NASBUG references for whose labels and purpose I knew, but whose absolute address I could not remember. The commenting throughout was good, but without the mnemonics and labels, it was very difficult to follow which register was doing what, particularly as jumps were consistently referred to as just 'jump', regardless of whether it was a jump relative, jump absolute, conditional or not.

To summarise, the book is an attempt to fill the gaping void between the absolute beginner at programming and the programmer who 'has got it all together'. A brave try, as no-one else has attempted it. Unfortunately, the book falls between the two stools, in that inadequate information is provided for the rank beginner, and the programming techniques used are liable to provoke mirth rather than serious reflection on the part of the experienced. The commenting is very thorough, but the absence of instruction mnemonics and labels makes following the programs (by single stepping) almost impossible. It is difficult to know who to recommend this book to, as it would be of little use to the beginner; hard going and poor programming practice to those with a little experience; and rib tickling to those who know what it's all about anyway. The only saving grace is perhaps the very detailed explanations of those monitor routines listed in the appendix. In conjunction with the listings given in the Nascom manuals (to pencil in the missing mnemonics), it becomes a relatively simple piece of logical thought to see what the monitor is up to. I only wish that the monitor listings given with the Nascom were commented in this fashion in the first place.

I note that the book has "Approved by Nascom" plastered all over it, it makes me wonder who approved it, and exactly what purpose he approved it for.

---

# Library

SOFTWARE LIBRARY  
=====

Below is a further list of programs that have now been added to the library. These are all for minimum system Nascom 1 or 2 with Nas-Sys. Most of them have been reassembled from the Nasbug library and therefore, although they have all been checked, some of them do contain portions of code that would be written in an entirely different manner if the programs had been written for Nas-Sys in the first instance.

Where the program has appeared in the Nasbug library, no description is given, as this can be obtained from Issue 6. Prices appear alongside each entry. Additionally there is a charge of 15p per order for postage and packing (Overseas, 40p).

FRENCH MEMBERS please note that British banks do not accept cheques in Francs.

## Nas-Sys Minimum System Machine Code Assembly Listings.

=====

S2	Reaction Test	0C80-0FAD	0.60
S3	Bouncing Beastie	0C80-0CFC	0.20
S4	Reverse	0C80-0F99	0.70
S5	Double Mastermind	0C80-0FB9	0.75
S7	JJ	0C80-0E0F	0.40
S8	Unizap	0C80-0ED9	0.75
S9	Random Buzz-word Linker	0C80-0F3E	0.50
S10	Quiz Program plus Italian File	0C80-0DCB 0DD0-0FD5	0.65
S11	Cockney Rhyming Slang File for S10	0DD0-0FD6	0.20
S12	Currency Types File for S10	0DD0-0FC0	0.20
S13	Capital Cities File for S10	0DD0-0FE1	0.20
S14	Ship Game Blow up ships using your torpedoes. Ships come at random speed, distance, and direction. This program requires the real-time interrupt mod. to the Nascom.	0C80-0FA4	0.70
By P.Sanders			
S15	Hangman Can you guess the word before you are hung ? Ability to add own library of words. Commented teletype assembly listing. By P.Grant. Data by J.Waddell. Load routine by J.M.Stevenson.	0C80-0F79	0.55

S16	Memory Test 1 Puts address related pattern throughout RAM and then checks it. Test thus reveals addressing errors.	0F00-0F6F	0.15
S17	Memory Test 2 Tests that 00-FFH can be written to each location of memory. Test thus reveals bit faults.	0F00-0F61	0.15
S24	Octal to Hexadecimal Convertor	0C80-0CE8	0.15
S28	Decimal to Hexadecimal Convertor As T28 but with improved input validation.	0C80-0D2B	0.20
S34	Random I-Ching Character Generator	0C80-0D25	0.15
S36	Sub-Search	0C80-0DA3	0.50

There are a lot more programs awaiting entry into the library, and we hope to include several new ones with each issue. If you have any programs of your own that you feel could interest others, please send them to us. Thank you.

---

#### SUBMITTING PROGRAMS TO THE LIBRARY

=====

We are pleased to receive programs in any shape or form for consideration for the library. All writers of programs added to the library receive a 5.00 voucher for use when purchasing library programs. The philosophy behind the library is to provide programs at as low a price as possible, so the only other reward offered is that of putting the writer's name in print !

To assist us in adding the programs to the library as quickly as possible it would be helpful if the following guidelines could be followed.

1. Include a statement of the program category. e.g. Assembler program for unexpanded Nascom with Nas-Sys, Super Tiny Basic program for 8K Nascom, etc.
2. Include a clear description of what the program does, and how to "drive" it.
3. Include the source text of the program, with comments.
4. Include a cassette of the program. (This will be returned if requested.)

Ideally (3) should be in Zeap format and (2) in Naspen format. This will particularly speed up the process, but failing this neat, legible listings will do very nicely thank you.

We will attempt to check the programs for bugs, but cannot guarantee finding them ! Sometimes programs may be modified by us, but generally they will be as sent. It is assumed that by sending in a program you are giving us permission to reproduce and publish that program and its documentation.

---

# Ads Pages

## CLASSIFIED ADS.

=====

1 NASCOM 1 built c/w keyboard	100.00
1 Buffer Board built	25.00
2 8K RAM Boards built c/w RAMS	60.00 each
1 Set Tiny Basic Eproms	15.00
1 40 column, 7x5 dot matrix printer, uses 3" aluminised paper. 64 chars. 6 bit ASCII parallel input. c/w 8 rolls of paper.	150.00

Phone: D.Watson. Rochdale (0706) 56824.

### Expanded Nascom System:

Nascom 2, 4x48K RAM boards, 4 double density, double sided mini floppies with control card, colour programmable high resolution graphics, I/O board, 8A psu, keyboard case. All professionally built and tested and installed in a rabbit hutch. Apply at any Nascom dealer and watch him cringe.

Nascom 32K extended memory card, compete with 16K of Dynamic RAM. Brand new and tested by Nascom, free of any bugs. Plugs straight into Nasbus. 110.00

Phone Vince : Walton-on-Thames 21078

Computer paper tape. 7/8" wide. 8" diameter reels. As new. Offers. Tape Punch (100 cps) and Reader (300 cps). Rack mounted. Offers. Phone Derek : 01-449 1690

# interface components

## AVAILABLE NOW - EX-STOCK !!

You've read about them,  
Now come and see them !

### CPU BOARDS

Nascom 1 Kit  
Nascom 2 Kit  
Nascom 1 Built  
Nascom 2 Built

### SOFTWARE

8K BASIC Tape for N1  
ZEAP 2.0 Tape for N2

PLUS - We also stock, or can obtain at short notice, an extensive range of Nascom spares; ICs, DIL switches plugs, sockets, cables etc.

### FIRMWARE

Nasbug T2  
Nasbug T4  
Nas-Sys 1  
Naspen VT  
Naspen VS  
Nas-Dis 1  
Zeap 2.0  
8K BASIC  
Tiny Basic  
Super T.B.

### POWER SUPPLIES

3A PSU Kit  
8A PSU Built

### OTHER

I/O Board  
Plus PIO kit  
CTC kit  
UART kit  
Keyboard Case

### GRAPHICS

ROM Graphics Board for N1  
Graphics ROM for N2

Because of the expected high demand for some of these items please ring us before ordering.

### PRINTER

IMP Printer  
Spare Ribbons

### MEMORY BOARDS

RAM A 8/16/32K  
RAM B 16/32/48K

RAM A can accept  
4 x 2708 EPROMs.  
RAM B can accept  
Page Mode option.

INTERFACE COMPONENTS LIMITED,  
OAKFIELD CORNER, SYCAMORE ROAD, AMERSHAM, BUCKS HP6 6SU  
TELEPHONE: 02403 22307. TELEX: 837788  
Write, telephone or call. Access or Barclaycard accepted

# **NASCOM**

## **I/O BOARD**

### **BOARD**

A standard NASBUS 8" x 8" PCB: twelve may be fitted to a system to use all 256 available ports, each board with three parallel input/output controllers, one counter timer circuit and one universal asynchronous receiver-transmitter. Each board may be fitted only with the options necessary, though later updates are simple. Boards are supplied with documentation covering all options.

### **PIO**

Each device carries two 8-bit read/write parallel data ports with separate control signals; applications for this type of device include direct interfacing with many digital-to-analogue and analogue convertors, the control of relays or electronic power controllers and uncomplicated EPROM programming.

### **CTC**

The use of this device saves complex programming when the system is required to count or time internal or external events; essential for instrumentation, process control and experimental robotics, it permits a higher program speed without sacrificing control.

### **UART**

Supplied with its own crystal clock and monolithic baud rate generator, this device frees the system computer's serial port to allow simultaneous operation of, for example, a cassette mass storage system and a printer. A V24/RS232C serial interface is also provided for direct connection to terminals and VDUs.

## **NASCOM EXPANDS**



Nascom Microcomputers Limited : 92 Broad Street : Chesham : Bucks HP5 3ED  
telephone (024 05) 75155  
telex 837571

# LAWRENCE THE LONG HAIREO WEIRDO

=====

We haven't heard from Lawrence this issue, but we did hear a rumour that he was assisting the police with their enquiries. (Central Records Computer seems to have crashed !)

We did, however, receive this communication from Doctor Dark. Is Lawrence too embarrassed to show his snout ?

