

# INMC NEWS

# 80

Issue: 2

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

September 1980  
- January 1981

'BIGGEST YET' ISSUE

## CONTENTS

Page 1	This is it.
Page 2	Chairman's Letter.
Page 4	Letters to the Editor.
Page 12	PASCAL Special Interest Group.
Page 13	Disks and the Nascom.
Page 16	Hardware Review. Screen Control Unit.
Page 17	Doctor Dark's Diary - 7.
Page 21	New Programs and Back Issues.
Page 22	Adding extra keys to an NI. Correction to I/O Board Review.
Page 24	BASIC Holiday.
Page 25	Classified Ads.
Page 26	Teach Yourself Z80 - Part 2.
Page 31	Free Progs. Assembler - LOAD and TAB. BASIC - Calender.
Page 34	Nascom RAM with Schizophrenia.
Page 36	Protecting Program Variables.
Page 37	Software Reviews. Integer PASCAL.
Page 40	DATRON PASCAL. MAPP 1-4Z. Parkinson BASIC Toolkit.
Page 42	IMPERSONAL Column.
Page 43	Book Reviews. "Practical Microcomputer Programming"
Page 44	"Z80 and 8080 Assembly Language"
Page 46	Software Review. Two BASIC Toolkits.
Page 47	'PRINT USING' Routine.
Page 48	Software Review. Watkins BASIC Toolkit.
Page 49	INMC80 Subscription Form.
Page 52	Free Programs. Machine Code - SIMP.
Page 54	Machine Code - SPACE INVADERS.
Page 56	Advertisements.
Page 57	
Pages 53/54/55	

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

PLEASE ALSO NOTE. INMC80 is run by a voluntary committee on a part-time basis, and it is not possible for us to become involved in technical correspondence. Please contact your NASCOM distributor for this.

# HIM again

Chairmans' bit

=====

WE ARE BACK !!!! My apologies on behalf of the committee for the very late arrival of this issue. The last issue came out before we all went on holiday, and now I'm sitting typing this looking at snow gently falling past the living room window. Either there is something radically wrong with the weather, or the intervening interval between typing this chairmans' letter and the last is a lot longer than it feels. Truth to tell, Paul and I have been extremely busy to the exclusion of virtually anything else. Admittedly we work for different companies but it seems our respective employers have both conspired against us getting this newsletter to press. Most of the copy for the October/November issue was in hand before the end of September and that was where everything stopped. We picked up the pieces a couple of days ago, and we have set the final copy date for tomorrow. I don't know what page count Paul has hidden on tapes and disks, but we intend this to be the biggest issue yet.

The first thing to notice is the the change of print style. To date most INMC newsletters have been prepared on NASPEN and then printed using a rather arthritic IBM printer. Progress has struck. We now have access to a Qume Sprint. This has not only meant that the printing process has been speeded by about three times, but that we can cram a lot more print per page. Our old page format of 72 characters per line, 56 lines per page has given way to 87 characters per line by 60 lines per page. This should help reduce our printing bills a little by using less paper.

For once I can not complain about lack of material for publication, we seem to be awash with it. There must be enough for this issue and most of the next already. Our thanks to all those who have contributed, even if we have not printed it in this issue. However, can I make an appeal for a Nascom owner somewhere near North London who has a Nascom 2 fitted with NASPEN and could put in a few hours transcribing letters, articles etc, prior to editing as typing time is now becoming our biggest stumbling block. Whilst asking for help, would a capable technical draughtsman please stand up and be identified. As you may have noticed we don't go overboard with illustrations or technical drawings. This is because none of us can draw. We have a number of hardware articles which we would like to publish, but our printer complains that he can't make litho plates from drawings on the backs of envelopes. Both jobs would only take a few hours every couple of months, and the reward is to see your efforts in print. Please reply to Amersham, for my attention.

Dodos are not extinct

=====

Thank you all out there who replied to our 'Prove a Dodo Exists' appeal in the last newsletter. May I thank particularly those I haven't had time to write to yet. Up until a few days ago we had identified our ideal dodo, but alas, extinction struck, and the hunt starts again (more about that later). Just as well I kept all the letters of application. One way and another we have managed to meet many prospective dodos, a few were even clever enough to ferret out an unofficial INMC committee meeting in the bar at the Cunard Hotel at a computer exhibition. We had decided that these budding 'Sherlocks' were really too clever to be dodos, as they had the wit to find and identify us. But as the dodo hunt is on again, who knows?

If you have been watching your magazine adverts, you will have noticed that a group of Nascom dealers have clubbed together to promote Nascom related products. More strength to their elbows, as this is one way in which lesser known products for Nascom will see the light of day. It is really most encouraging to see this kind of movement, as with the future of Nascom still uncertain (didn't you know, then keep reading),

consortia like this can do a lot to help Nascom users world wide. We have twisted their collective arms for an advert which covers many products you may not have previously heard of, and we hear there is yet more to come.

On the score of collectivity (sounds a nice socialist, this fella), we've been thinking that this newsletter is exclusively about Nascoms. Yet when you get to the end of this 'Chairman's bit', you will realise that the likelihood of more Nascom (Nascom meaning from Nascom) goodies appearing is disappearing as fast as fried snowballs. Yet other goodies are arriving thick and fast from other sources. We already have the Winchester Technology colour graphics board and Interface's EPROM card, both Nasbus compatible. We have heard rumours about a new Nasbus based Z80 computer being planned for the middle of next year (shades of Nascom 3), and the rumours about a mini programmable graphics is certainly more than a myth. Anyway, to the point. Whilst we think that this newsletter should remain an organ for the enlightenment of Nascom owners, we also feel that more emphasis should be put on Nasbus itself. What do you think? The more manufacturers who support Nasbus the better? Or should Nascom and Nasbus be allowed to sink into oblivion, and simply remain a name known to a few.

Now to the bad news. (No not that, not yet). Richard Beal, who has been with the committee from the start has resigned. He seemed reluctant to do so, but as he's about 16,000 miles away now, keeping in touch is a little difficult. His excuse has been that his job has taken him to the Solomon Islands (right and up a bit from Australia), and although he protests that he is working very hard, we can not remove the picture from our minds of him sitting on some sun scorched sandy beach under a palm tree sipping coke. Thank you for all your time and help in the past Richard. Just one thing, when you come back in March, please don't bring back NAS-SYS 4, 5 and 6 with you !!!

Now the really bad news. You may have seen in the press that Nascom had been rescued by Alteck Technology Initiatives headed by one Peter Mathews. Having met Peter and his associates I can say that they seem 'nice people'. Peter, an avid Nascom owner, and Managing Director elect of Nascom volunteered to be our dodo. That killed several birds with one stone, giving us just the type of dodo we needed, and one that would listen to the INMC when it came to grass roots requests regarding Nascom products. Sadly it seems it has all fallen through. I had a phone call from Peter only a few days ago to tell me the news, as he felt it important that the INMC should be 'in the know'. I must admit I was flattered by this, and yet saddened by the news. I don't know why the deal didn't go through, and it would be unjust to speculate. I just feel sorry for Peter who I think would have made a great success of Nascom.

All this puts the future of Nascom in question again. The receiver is continuing to manufacture Nascoms and they are being supplied to the dealers. However, dealer quotas are now falling further and further behind the delivery requirements, and if I may get a personal 'gripe' in here in the hope that the receiver will read this; the profit on the quantity of Nascoms supplied last month did not cover the cost of my company's advertising. We could have sold three times as many Nascoms as were supplied. If this situation continues much longer, then from a pure economic point of view, my company can no longer afford to advertise Nascom, and that would be one more nail in the coffin. I think we are all agreed that Nascoms are very good products at the right price in the market. Please Mr. Reciever, make more of them to keep it alive.

D. R. Hunt.

# Letters

Machine Code Programming - made easy ?  
=====

Dear Sir,

Firstly, to those Nascom owners who read with trepidation every article purporting to teach machine code programming the 'EASY' way (apologies to Dave Hunt). When I bought my Nascom 2 early this year I found that excellent though it is, the Basic very soon lost its challenge as familiarity made programming a fairly easy matter and I became interested in machine code routines as being so much faster and more versatile. So in my innocence I went out and bought the SARGON book (a chess program listing in 8080 ASSEMBLY language (TDL MNEMONICS)). Naturally, I was utterly lost immediately, although I understood what 'Hex' numbers were and what various instructions did to the Z80 registers. It was then that I discovered Assemblers, essential to comfortable machine code work. I sent for the V & T Assembler as reviewed in your pages (issue 5/17) and began the laborious process of translating the TDL mnemonics to Z80 mnemonics using the assembler to store the precious listings on tape. Purely by coincidence I came across the Liverpool Software Gazette feature on converting Sargon for Nascom 1. Suffice it to say that eventually, by assembling Sargon in two halves and spending a week or two debugging I got a working program! Worse than that, it beat me every time!

Now, I had read articles ad nauseam on machine code programming and understood each step as explained, but still I never knew how to get off the ground, what registers to use for which functions, etc. It was not until I had spent a month immersed up to the elbows in Sargon and an assembler that I began to see what it was all about. I started by assembling blindly and finished by understanding how the program works, how an assembler works and how to use the Z80 instructions in a practical way. I have even written an alternative routine for Sargon to have it play itself at different levels, a feat I could never have considered 2 months ago! I think the point is that to pick up the ideas of machine programming, it takes TIME to become familiar with the concepts concerned, a practical program to work on to provide the motivation to 'get things right' (I never found that slogging through examples of 16 bit multiplication interested me enough for the details to sink in) and good software to lend a hand (in this case V & T's assembler).

Incidentally, I recently bought the Bits and PC's toolkit which has brought back the excitement to Basic programming by eliminating tedious processes such as numbering lines, re-numbering entire programs, appending programs onto each other, listing contents of variables used after a run, finding strings, listing lines in compressed form, finding lines containing errors, single stepping with display of variable contents, converting hex numbers to decimal, repeating keyboard inputs, and much more, by providing Direct-mode commands to do it all for you. (Ed. See reviews elsewhere in this issue.) The only drawback it has is that it uses the input and output tables - excluding their use by Basic machine code routines (I'm sure there's a way round this), and to run under the toolkit 'old' Nas-Sys programs in BASIC may need minor input modifications. A pity Toolkit has "find string" without "change string" but this is just a quibble. For those who have entered Super Startrek or similar programs with the XO command so as to get 72 characters per line, Toolkit has trouble with finding strings in these lines, but other functions seem to work OK.

For two Eproms (the contents of which are relocatable) it's a bit heavy on the pocket, but a dream to use, and all things considered, good value for the 'Keen Programmer' in Basic.

FREE! Here is a simple idea to control the speed of the CRT display by using the User output routine.

In Basic my version is as follows:  
(The routine starts at ODOOH and is 12 bytes long)

```
10 DATA 3328,1781,3330,-255,3332,-752,3334,-13839
20 DATA 3192,3328,31187,1918
30 FOR I = 1 to 6
40 READ X,Y
50 DOKE X,Y
60 NEXT I
```

LINE 10 contains the routine.  
LINE 20 contains the addresses to be changed.

To vary display speed, POKE 3330,X where X can be 1 to 255, X=1 is Fast, X=255 is Very Slow ! This can be done at any point in a program.

In Machine Code, it looks like this:

```
ODO0 F5  PUSH  AF ; SAVE AF
ODO1 06 01 LD B, 01 ; 01 = DELAY CONSTANT
ODO3 FF  RDEL   ; RST 38
ODO4 10 FD DJNZ -2 ;
ODO6 F1  POP   AF ; RESTORE AF
ODO7 C9  RETURN ; Return to sender !
```

To EXECUTE FROM NAS SYS:

```
Load Routine address into OC78
Load Table address (OC7E) into OC73
```

These are the Output Routine Address (UOUT) and Start of 'table of output routines' (OUT) respectively in the Nas Sys workspace.

Here is a routine which will save the trouble of entering these addresses by hand each time:

```
21 7E 07  LD HL,077E ; Table address (077E)
DF 71    SCAL NOM   ; Call change address routine
21 XX XX  LD HL, ROUTINE ADDRESS = XXXX
22 78 0C  LD (OC78),HL ; Set jump to user routine
DF 5B    SCAL MRET  ; RETURN to monitor.
```

This can be tacked on to the end of the routine itself (or onto the beginning) and should be executed before trying to change display speed. To vary speed, load (Routine address +2) with the value required. Remember, '0' will give you a very slow display, but '1' will give almost normal speed. This is why I have set the initial speed at '1'.

Yours byteingly,  
DAVE LORDE  
Pontyclun  
S. Wales.

Ed. There is now a tape version of the Sargon chess available. It comes complete with the Sargon book, a graphics EPROM containing a Chess set, and a small piggy-back board to allow switching between the normal graphics and the chess set. This is produced by Bits & PCs, costs 35.00 plus VAT, and is available from various Nascom distributors.

D.J.Software ?  
=====

In the last issue of INMC80 News there is a very interesting article about converting Zeap 1.0 to Nas-Sys operation. I would like to obtain a copy of the patch tape mentioned in this article, but I cannot find any other reference to DJ Software who supply it.

Please can anyone give me their address?

J.B. HAWKES  
Aylesbury

Ed. - Sorry, we don't know. Any offers ?

Heaps of Praise  
=====

The last issue (INMC 80-1) of the newsletter exceeded all expectations (including the one that it might not come at all !). Personally, I would not complain if the subscription was increased, as it represents such good value. "Piranha" is worth the annual subscription alone. Come to that, so is the modification to get the Nascom 2 cassette interface running at 2400 Baud! (Eat your hearts out T\*\*dy, P\*t etc.).

Thank you.  
Graham Smith  
Stockport.

Plotting Mods.  
=====

Dear INMC80,

With reference to INMC 80 Iss.1, may I suggest the following changes to M.Taylor's plotting routine. In general it is convenient to have the origin on the bottom left hand corner. Also a MOVE not LINE command may be desired.

The following changes are proposed:

```
65507 B=ABS(B-44)
65508 IF T=1 THEN X1=A:Y1=B:RETURN
65509 IF T=1 THEN X2=A:Y2=B
65510 IF X1=X2 AND Y1=Y2 THEN SET(X2,Y2):RETURN
:
:
65516 SET(X+X1,Y):NEXT:X1=X2:Y1=Y2:RETURN
:
:
65519 SET(X,Y+Y1):NEXT:X1=X2:Y1=Y2:RETURN
```

This modification allows the user to move the cursor (or pointer) across the screen and draw lines from any two points. The following variables are used:

- A - X-coordinate (horizontal)
- B - Y-coordinate (vertical)
- T - (=1 Move to coordinate, =2 Line to coordinate)  
<First command must be a move>

So the program:

```
10 A=10:B=10:T=1:GOSUB 65507
20 A=80:B=10:T=2:GOSUB 65507
```

would draw a line from (10,10) to (80,10).

Using this method an array of coordinates could be plotted within a short loop. As an example the following program is suggested:

```
10 DATA 10,10,1,80,10,2,80,40,2,10,40,2
11 DATA 10,10,2,20,15,1,30,15,2,60,15,1
12 DATA 70,15,2,45,20,1,45,25,2,35,30,1
13 DATA 40,35,2,50,35,2,55,30,2,0,0,1
14 DATA 25,20,1,25,20,2,65,20,1,65,20,2
20 FOR I=1TO20
30 READ A,B,T:GOSUB 65507
40 NEXT I
50 END
```

For a clearer picture try GOSUB 65508 instead in line 30. I hope this mod. will be of some use.

Yours faithfully,  
K. Kishimoto,  
Manchester.

Reserved Words  
=====

Dear Sirs,

Firstly, let me point out yet another sinful error in the GREAT "NASCOM 2" book of WISDOM. Nowhere is there mention of the 'MO' error (MISSING OPERATOR) (There is in my manual, as 'MISSING OPERAND' - Ed.) which occurs when an argument is missing from an equation. Incidentally a list of error codes can be found inside the Basic Rom at locations E2B9 through E2DD.

Secondly, here is a program which will generate all the instructions and statements in the Basic language and place them on the line number equal to the instructions reserved character value.

```
OD00 DD 21 FA 10 LD IX,10FA
OD04 21 00 11 LD HL,1100
OD07 11 06 00 LD DE,0006
OD0A 06 FF LD B,FF
OD0C DD 75 00 LD (IX+0),L ;$ LOOP
OD0F DD 74 01 LD (IX+1),H
OD12 DD 70 02 LD (IX+2),B
OD15 DD 36 03 00 LD (IX+3),00
OD19 DD 70 04 LD (IX+4),B
OD1C DD 36 05 00 LD (IX+5),00
OD20 DD 19 ADD IX,DE
OD22 19 ADD HL,DE
OD23 10 E7 DJNZ,$ LOOP
OD25 DD 36 00 00 LD (IX+0),00
OD29 DD 36 01 00 LD (IX+1),00
OD2D DF 5A SCAL 'Z'
```

To run this program, first initialise the Basic interpreter with the 'J' command. Then return to the monitor and execute the above program by typing E DOO NL. This will generate the program in Basic and return control to the interpreter. Then type 'LIST' NL and the instructions will be displayed alongside their ASCII (CHR\$) values. Note that the line numbers are in reverse order. Basic uses a list of memory addresses rather than line numbers to determine which order the information is printed out.

Yours,  
B.E. Kelly,  
Yate, Bristol.

Money Wasters !!  
=====

Dear Sir,

If, as you claim, you are short of money then I suggest you stop wasting our subscriptions on publishing c\*\*p - like pages 4 and 48 of Issue 80-1.

Dr. G.R. Kelman.  
Leicester.

More Praise  
=====

Dear INMC80,

I was very pleased to hear that the INMC was back in business with the same committee. INMC(80) News is by far the best of the microcomputer journals if you are a Nascom owner. The only thing the others do better is get advertisements - good luck in your efforts. Provided that they are Nascom-related I don't see any objection to much more than the 5 pages you have suggested, provided of course that there is still some editorial matter left. Related possibilities that you have probably considered already are the insertion of separate advertising material when mailing the News, and using the mailing list for an independent mailing shot of advertising matter. I'm not usually all that keen on being on mailing lists, but would welcome it in this case; it isn't easy to keep track of all the hardware and software being offered for Nascoms these days, though it is certainly encouraging that other manufacturers are showing so much interest.

Could I suggest that a review of the ways of housing a Nascom would be useful? I have a Vero frame - Vero's own, not the Nascom version - but am not all that pleased with it, especially not with the accessories (front panels) which turn out to be unsuitable for Nascom 2 after they have been sold to you!

Dr. Dark  
I'm sure someone must have made the point, but as the statement has been printed twice can I correct the impression that ED6B is an unpublished op code. It is in the manuals as LD HL,(nn), although there is a shorter version: 2A. Similarly, ED7B, mentioned in INMC News 7 is listed as LD SP,(nn).

Yours sincerely,  
G. DAVIES,  
Surrey.

NASCOM CRISIS  
=====

Dear Sir,

The Nascom crisis is a disappointment - this is just to say I appreciate your work and hope that INMC80 can continue. I only joined a few months ago but the two newsletters were well worth the subscription (particularly as the contents are not available elsewhere).

MIKE WHITEHEAD  
Dundee University.

SPACE INVASION MODS ?  
=====

Just a few words of thanks for the trouble you are taking to keep the magazine going after all the developments at Nascom and lots of good luck for the future. Please keep the "KIDDIES GUIDE TO Z80 ASSEMBLER PROGRAMMING" by Mr. D.R. Hunt going for all us beginners (ME).

I purchased "SPACE INVASION" by G. CLARKE from the library, can you tell me what changes are needed to run on my N2 under NAS-SYS as this will save me much money at the local amusement arcade? (Ed. Mr Clarke?)

Yours, a frustrated programmer,  
R.K.HANSELL,  
Kilwinning, Ayrshire.

NEWBEAR CASSETTE INTERFACE and PASCAL  
=====

Dear Sir,

I have just received Issue 1 of INMC News 80 (or THE RETURN OF THE INMC!). I hope that this is not the start of a long series (Son of INMC, INMC rides again etc etc). On reading it, a few points could do with expansion or clarification, hence this letter.

Mr. Keneally's review of the Newbear/Cottis-Blandford cassette interface may serve to introduce others to the joys of high speed cassette loading. I have been using the C/B interface for 18 months now at 1200 Baud with astonishing reliability, and have built three units in total (two for myself, one for a friend) all of which work very reliably. There are one or two points which may help Mr. Keneally achieve reliability. In the ACC Newsletter Vol.16 No.5, dated December 1978, Bob Cottis points out that for high mains hum, either the input capacitor can be reduced to about 10nF, or a 1K preset placed between 5v and ground, with its slider taken to the amplifier side of the input capacitor. This preset should be adjusted so that with no input signal the schmitt trigger is in the middle of its hysteresis band i.e. midway between the two points at which the schmitt changes state. I have done this on all the boards I have made, set the frequency of the 555 i.c. with a frequency meter, and they can all read each others tapes very reliably. One other point on this board: The Nascom uses the same clock for receive and transmit, but the C/B board does not. The transmit clock from the C/B is selected by the rate selector option. The receive clock is slow running, and is pulled up by the phase locked loop to match the receive clock demanded by the incoming data.

If the C/B board is connected to the N1 with both clocks to the external clock pin, then the output clock will cause the receive clock to slave to its own frequency, and one might as well not have a phase locked loop at all. The clocks should be connected to the UART by a single pole double throw switch, which is thrown to the receive side to receive, and to the transmit side to transmit. Alternately, a bit of surgery on the board will allow a track to be cut which joint the two clock pins on the UART, and the correct leads attached to each pin. In my cases, I took all the connections I required to a connector I mounted on the N1 board, and made a connection through a multiple switch to allow either Nascom Tape Standard or 1200 CUTS to be selected. The reliability this modification (of seperate clocks) has given is unbelievable - I've almost forgotten about tape errors.

With regard to tape quality, I use Microdigital mostly, and for very large files BASF LH60 or FUJI FL30, with no problems at all. If available, use a frequency meter to set output clock from 555 to 76.8Hz, and if thought desirable, check the capture range of the PLL as described by Mr. Cottis in PCW December 1979.

Arising from my notes on implementing a Tiny Pascal, I have since obtained from LP Enterprises a copy of the 8080 runtime support and P code to machine code converter written in Basic. These, bound together with a copy of the three original articles so that they form a complete unit, cost 11.95, and would form a good starting point for someone wishing to get a Pascal up and running. Also of interest are articles in the LSG first four issues on the subject of getting this language running on a C\*mm\*d\*r\*P\*t (I don't use dirty words, unlike Dr Dark). As my own machine has packed up, and is currently having a holiday in Nascom's repair Dept, I have not been able to progress any further with the development of the Pascal. I have however discovered that there is available from Springer Verlag: "A Concurrent PASCAL Compiler for Minicomputers" by A.C. Hartman, which describes a seven pass compiler for Concurrent Pascal in detail, and from McGraw Hill: " the BYTE Book of PASCAL", with "two versions of a Pascal Compiler, one written in BASIC, and the other in 8080 assembly language". I have both of these books on order, and will report on them when they arrive.

General Points:

Weller's assembler has a nice feature - it can read source from a paper tape, and assemble it, which is a nice trick for very large programs. I must have a go at that. It could be implemented on Zen fairly easily.

Has anyone given any thought to the inelligence of designing for the Z80 as if it were an 8080 with a few extra instructions? In my view the Z80 is a very different machine to the 8080. For example, in the case of a Nascom, which is not running under interrupt control in general, the ancillary bank of registers is there and empty most of the time. Zeap uses this to good advantage. The indexing instructions are very rarely used, and yet they can very easily lead to a nice neat compact solution to a problem. The relative jumps can allow position independant code to be written, which can have advantages if you run out of memory on a big assembly. The Nascom BASIC seems to be an 8080 Basic with input/output patches all over it, which are concerned with deciding which monitor is available, and even emulating it (?). I have discussed the disassembly with a fellow Nascom fanatic (a trusted associate) and we are of the opinion that were it written in Z80 code from the beginning, it could be made about 20 o/o smaller and about 15 o/o faster. For example, it could on initialisation look to see which monitor it had available and set a flag which it would look at each time it wanted input or output, instead of doing all the tests it seems to do. As I am working towards a Pascal compiler, I am not inclined to have a go at rewriting it - I think I'll save the 35.00 for the XTAL BASIC and put it towards the price of CP/M V2.2 and a floppy disc controller. Can anyone help with details on I.B.M. 33FD 8" floppies - I have two on indefinite loan, and would like to get them running, but have no further details other than that they are in working order. I understand them to be single sided single density, rather old-fashioned and outmoded really, but still 250K bytes on each !

Finally, remember the original definition of a committee - a body of people appointed to sit in judgment on a lunatic to decide his sanity!

RORY O'FARRELL  
Co. Wicklow,  
Ireland.

LOCAL CLUBS  
=====

Dear Sir,

Upon joining the INMC a few months ago, I was taken by how comprehensive and useful the newsletter is, considering the low budget you must have to work within. I would however like to see more details of local clubs and club activities made available, if there are any !

Every club depends on the interaction and communication of its members with the H.Q. With this end in mind if there are no local group meetings being held then why not ? I personally would like very much to meet fellow owners of NASCOM's (perhaps over a pie and a pint in a local hostelry) so we can lurch out of the door satisfied in the knowledge that we are members of an elite club who have exchanged ideas (visions ?) of the system expansions to come!

With this purpose in mind might I suggest that any willing members or non-members (are there any) who would like to meet with the aforementioned in mind, please contact me at the address below so that we can arrange something, sometime, somewhere in the locality of St. Albans. Looking forward to hearing from someone out there.

Bye for now.  
JOHN MARJORAM  
3 Blenheim Road,  
St. Albans.

Ed. Any others who would like their addresses published with a view to starting a local club ?

HELPING THE HANDICAPPED  
=====

Dear Sir,

During recent months I have been made aware of the activities of a number of people involved with applications for microprocessor based products for the physically handicapped.

In particular I have read with interest an article that appeared in a recent issue of Computer Age written by Patrick Poon.

Although I have left Nascom I would like to see if it is possible to gather together names of people who would be interested in both the specific project Mr. Poon was involved in, i.e. children with Cerebral Palsy, and also other aspects where the microcomputer could possibly aid the disabled.

I am prepared to offer the services of <sup>-12-</sup>Interface as a co-ordinating body and mailing address and I would very much like to know if any of the members of your club have had any involvement with, or would be prepared to become involved in a project or projects relating to this extremely important field.

My understanding of the situation concerning direct or indirect government support is that there is virtually nothing, and it is obviously going to require the pooling of resources of both private commercial organisations, private individuals, private charitable organisations, and research development capabilities which lie within the various places of advanced studies and teaching.

I have also been in touch with Peter Deacon who indirectly is linked with the Spastics Society and I would like to arrange a meeting of interested parties to discuss the matter further.

If you would be kind enough to publish this letter I know it would reach nearly two and a half thousand active Nascom users, some of whom may be prepared to assist in this project.

Yours sincerely,  
J.A.Marshall,  
Managing Director,  
Interface Components Ltd.

## PASCAL

PASCAL Special Interest Group  
=====

Object of the INMC80 special interest group is to bootstrap a subset of Pascal to run on Nascom. Bootstrapping is the process whereby a very small subset of a language is implemented, and used to compile itself. This small compiler is then used to compile expanded versions of the subset until a satisfactory result is achieved.

Because of the amount of work in implementing a language, as much use as possible should be made of existing work. The Byte Tiny Pascal in Basic (1) or 8080 machine code (2) forms a good starting point, as the source is readily available, and an implementation for Nascom is under way (3) or commercially available (4). This program compiles the Pascal source into a series of subroutine calls to runtime support routines.

As a working aim, the intention is to implement a Pascal compiler for Nascom running under NAS-SYS, with floating point arithmetic to at least the accuracy of 8K BASIC (5), but possibly better, supporting the subset of Pascal implemented in the Byte Tiny Pascal, extended to include the Pascal 'type' constructs. The object machine is a 32K Nascom, with ideally runtime support in Eprom, and tape I/O for storage. For efficiency, the runtime routines will be written in machine code, and treated as system calls by the compiler. The program when finally completed will be inserted in the INMC80 program library, as will all the major intermediate steps.

CONTACT POINT : Rory O'Farrell,  
Tinode,  
Blessington,  
Co. Wicklow,  
Ireland.

(Stamp or International Reply Coupon please).

Refs:

- (1) BYTE Nybbles LS100, "Tiny Pascal Compiler". L.P. Enterprises 11.95.
- (2) BYTE Book of Pascal, McGraw Hill. 15.00.  
This also contains reference (1) ... much better value !
- (3) INMC80 No.1
- (4) Datron Micro Centre, Sheffield, 35.00 + VAT.
- (5) Z80 Gourmet Guide and Cookbook, Scelbi 8.90. MOI.

## Disks

DISK SYSTEMS  
=====

It had to happen, three competing disk systems for Nascoms. Two from the States, one via our old friends at Barnett, one via Airamco in Scotland, and the third is a British designed one available from a consortium of Nascom dealers, and originating from the nether depths of Farnborough, Harrow and Kingston. Now, at this stage, we know very little about the Comp system, nothing about the Airamco, and an awful lot about the Henelec/Gemini system. In fact the non appearance of the October - November issue of the INMC80 News is directly attributable to the Gemini Disk System. You see, the two main editor/typists involved with the INMC have been up to their eyeballs getting the thing ready for production. No excuses, but the laws of nature still only allow for 24 hours in a day (I'm afraid we're still working on that one, very tricky !!).

We don't intend to review the systems here, as that would be a trifle biased, but by Christmas there will be some 200 Gemini systems around (and who knows how many of the others), some points about disk systems ought to get a look in. We haven't heard from a single member owning either Comp or Airamco systems, and at the time of writing Gemini systems are only just going on stream, so we haven't had any user feedback on that either. But as we know a lot about the Gemini, and the majority seem to be going to INMC80 members, a few hints and tips won't go amiss.

Now the EBASIC supplied with the system works on ASCII disk files, so how do you get some of your nicer Nascom Basic programs onto disk without the tedium of typing them all out again. Easy. First LIST the Basic program to tape. How? Load the program under Nascom Basic, go into the MONITOR, type X0, and warm start Basic. Zap a cassette into the recorder, start it up, set the Basic WIDTH to 80, set the LINES to 32000, and then LIST. Your Basic program goes to tape as an ASCII file. If there are a number of programs to be transferred, copy them to tape at the same time, as changing from NAS-SYS to CP/M gets a bit tedious if repeated too often. Ok, so now we have a tape of the Basic program(s) saved as ASCII strings, how to get it onto disk.

Because the inimitable RB had more than a passing influence on the design of the software for the Gemini system, there is a routine hidden in it which scans both the keyboard and the serial input. Does that ring any bells? If not, you haven't understood how NAS-SYS and NASBUG work. Anyway, Under CP/M, PIP.COM is really quite clever. It's not just a disk copying program !!! In effect, it can copy anything from anywhere, and put it someplace else. We want the tape (which because of the scanning of the serial input), comes from the keyboard routine to end up as a disk file. In CP/M, the keyboard and video are referred to collectively as CONSOL so we want to PIP from CONSOL to disk. Snag, the stuff is coming in as a continual bit stream, so there will be no time for the disk system to stop and shovel that data away whilst the data is coming in. So effectively we must load it all into a buffer first. PIP has just such a command.

A>PIP FILENAME.BAS=CON:[B]

The [B] is a PIP option to buffer the input in RAM before sending it to disk. Give the disk system that command, wait until a prompt appears and start the tape. The ASCII strings on tape will be gobbled up into RAM, and when the data is finished, type 'ctrl/S' to tell it to write it to disk and 'ctrl/Z' to tell it it's finished. If you only have 16K of RAM and the program is a long one, you won't be able to load it all at once. Load about 8Ks worth, stop the tape, type 'ctrl/S' to write the first bit to disk, rewind the tape a bit, and having written the first half away, restart the cassette. The file will be a little screwed up in the middle, but having got it on to disk you will have to edit it a bit using ED anyway, so it shouldn't be too difficult to unscramble the overlap in the middle. Don't expect to be able to run a Nascom Basic program under EBASIC, they're syntactically different, and EBASIC is particularly fussy about spaces between reserved words. Unless it is the simplest program

imaginable you are going to have to edit it.

Although this little piece has been written with the Gemini system in mind, it applies equally to the other two (assuming that Comp and Aircamco have written the CBIOS correctly). We doubt that the serial input looks at CON:, but it should look at RDR:, so substitute RDR: for CON: in the above command.

Another little point which applies generally to CP/M 1.4 is that files are written and read in 16K chunks known as extents. Now what is not obvious is that if a program is greater than 16k, ie: more than 1 extent, then when reading a program back, CP/M has return to the directory to find the location of the next extent. Whilst its there, CP/M writes a flag into the directory to tell it its gone on to the next extent. If you write protect a disk then you won't be able to read files which are larger than 1 extent as CP/M won't be able to write the flag into the directory.

With the Gemini system, it can't have escaped your notice that the keyboard option defaults to lower case when using SYS.COM. We think Richard did it because he liked it, and after a time I now prefer it, however, it can be a pain at times. ED has a funny quirk with the 'I' command which even though the 'V' option has been selected to lower case, it will only input lower case if the 'I' command was given in lower case. A Digital Research bug ??? Another case in point is the Microsoft MACRO-80 assembler which will only accept commands in upper case, yet Microsoft's LINK-80 will accept either. Confusing ain't it. Dear old RB having decided to default the keyboard option to lower case, didn't leave any way of changing it. In the end I was forced to find out how it worked and came up with this stupid little .COM file. Enter it under DDT and then SAVE it.

UPPER/LOWER CASE SWITCH MACRO-80 3.35 Page 1

Title UPPER/LOWER CASE SWITCH  
Subttl Version 0.1 for SYS 1.1

.Comment \*

PROGRAM CC.COM  
=====

Purpose; to reverse the kbd option bit within SYS V1.1. For use with Henelec and Gemini CP/M disk systems.

Finds the position of KOPT by taking the known location of the warm boot jump and the known offset between WBOOT and KOPT.

D. R. Hunt 30/11/80 \*

```

0000' .Z80
      ASEG
      ORG 100H

0000 $WBOOT EQU 0000H ; Warm boot jump
09B6 KOPT EQU 9B6H ; Offset from BIOS

; Calculate position of KOPT.
0100 2A 0001 LD HL,($WBOOT+1)
0103 11 09B3 LD DE,KOPT-3
0106 19 ADD HL,DE
; Get KOPT bit, toggle it and put it back.

```

```

0107 7E LD A,(HL)
0108 EE 01 XOR I
010A 77 LD (HL),A

; Return to CCP
010B C9 RET

END

```

Not a bad little trick that, note the use of the warm boot jump to tell us what size the CP/M system is. This little trick has other uses. For instance how about a tape read routine in generalized Nascom format. The problems come thick and fast with that one. The main problem is to duplicate TX1 out of NAS-SYS. This is the one that goes away, displays the header and checks the checksum. Because of the length of the CRT routine it barely makes it at 2400 BAUD at 2MHz on a Nascom as it is. Now to do the same thing under CP/M takes even longer, not only does the poor little byte to be displayed have to fight its way through the CRT routine in CBIOS (thats already as tortuous as the one in NAS-SYS), but it has to go through all the CP/M FDOS checks first. Believe me, it only just makes it at 1200 BAUD at 4MHz with no wait states. So how to bypass the FDOS (cos we already know the data is correct)? Well it would be a simple matter to call CONOUT in the CP/M, the only trouble is, it moves about with different size systems, so we have to find out where it is first. We already know where the system is located because the warm boot jump at 0000H tells us, we also know that CONOUT is offset from WBOOT by a fixed amount, so we can calculate its address. Now to call it. We use what is known as a 'fake call'. Find out where the routine is to return to, push that location onto the stack and then jump to the routine, the address of which we have just calculated. Having completed the routine, the RET at the end pops the last address off the stack (as normal), and returns. Provided nothing happened to upset the stack, this will be the return address. The CRT routine looks like this:

NAS-SYS MACRO-80 3.35 Page 1

Title NAS-SYS  
Subttl CRT, display on screen

.Z80

GLOBAL CRT

.Comment \*

This routine displays A on the screen. It replaces ROUT in NAS-SYS.

D. R. Hunt 16/9/80 \*

```

0000 $WBOOT EQU 0000H
0009 CONOUT EQU 9 ; Offset from WBOOT

0000' F5 CRT: PUSH AF
0001' E5 PUSH HL
0002' D5 PUSH DE
0003' C5 PUSH BC

; Save return address
0004' 21 0011' LD HL,RTN
0007' E5 PUSH HL
0008' 4F LD C,A

; Calculate call to CONOUT
0009' 2A 0001 LD HL,($WBOOT+1)
000C' 11 0009 LD DE,CONOUT

```



```

000F' 19          ADD HL,DE
                ; Jump to it
0010' E9          JP (HL)
0011'          RTN:
0011' C1          POP BC
0012' D1          POP DE
0013' E1          POP HL
0014' F1          POP AF
0015' C9          RET

                END

```

Not a bad little trick, and by using it, you can get at all the legal calls within the CBIOS. The only snag is that I'm not sure it's entirely legal. I'll bet some person will decide that it shouldn't be used within CP/M.

Still that's all about disks for this time, except to remark that this issue of INMC has been prepared using the new DISKPEN with all its enhancements, and of course, the whole issue was saved on a couple of disks.

=====

SCREEN CONTROL UNIT REVIEW

=====

This is a review of the Screen Control Unit produced by:  
R. W. Electronics, 27 The Vineries, Acocks Green, Birmingham, B27 6SB. The kit costs 27.00 inc. P&P and VAT.

This control unit provides 5 functions :- reset, display on (normal), display off (blanked), white on black (normal), and black on white (reverse video). It is constructed on a single sided 8" X 5.5" fibreglass pcb that plugs into a NASBUS and has a wire connection to an existing Nascom IC socket.

The sample kit provided for review took under 2 hours to build on the well prepared board, the legends being very clear and informative. Over half an hour was spent in inserting the wire links required since the board is single sided.

No problems were encountered in building the kit. In fact this is one of the best presented kits our reviewer has seen. The only hiccup occurred because the kit was first configured for a N1 (someone else can't read simple instructions?) and could not be tested on the N2 available. The kit is not intended to be changable between N1 and N2 but with some careful soldering this was accomplished.

Anyone who is thinking of purchasing this kit should also acquire another edge connector socket since there is not one in the kit.

Testing of the unit presented no problems (it worked first time) although the suggested memory address for the unit (it uses a memory mapped port to control it) seemed a little strange. The handbook suggests using address D000H (selected by wire links on the board) but 0800H would seem a better alternative since this is not used by any known software and provides a record of the last code sent to the unit in the first location of screen RAM.

In summary, this would be a very good first project for someone who purchased a built NASCOM and now wants to learn a little about assembling kits. The unit offers only VDU on - off, and screen inverse, and as such would seem to be somewhat overpriced. Perhaps a NASBUS socket could be provided in the price since it would appear that all other Nascom hardware suppliers provide a socket in with their kits. However, if you consider the time and effort involved in going it alone with a 'DIY' job, this unit would probably work out cheaper.

# Doctor Dark's Diary

DOCTOR DARK'S DIARY -- EPISODE 7

=====

This could be a very long episode. I was going to call it the "bumper try-to-write-more-than-David-Hunt episode", but sadly it is now the "write-a-lot-because-I've-time-to-spare episode". Marvin is a very sick computer and will almost certainly be away from home for some time. Cheques of condolence will be gratefully accepted, of course...

-----

SPIRAL WIPE CONTEST.

-----

I hope lots of you have entered this brain-torturing competition. It wasn't my idea, I'm not that sadistic. I have not actually sent an entry, but there is no truth in any of the following rumours:

- (i) it is because my game "SCRUM" didn't win the Christmas games competition.
- (ii) there is no shorter or quicker way.
- (iii) I didn't want to re-invent the wheel.
- (iv) there is no rumour number four.

The fact is, at the time, Marvin was being rebuilt around several new parts, of which more later, and I wouldn't dream of sending in something I hadn't tested. (Unless I happened to feel like annoying every-one...)

-----

WHAT THE OTHER PAPERS SAY.

-----

The first quotation in this new section comes from Henry Budgett's series on microcomputers in the October issue of E.T.I. and shows that not everyone invents the wheel even once, or something....

"As an example of this, there are some 20 extra codes built into the Z80 that are not mentioned in the manuals. Apparently they are not all guaranteed to work on all Z80's. Anyone know what they are and what they do?"

Can it be that the normally well-informed Henry doesn't read INMC80 ?

There's an excellent article in Computer Age (September issue) entitled "Are Computers Alive ?" While the author doesn't prove that they are, he knocks down flat all the arguments I have ever heard people use against the possibility. The condition of Marvin is probably best described as comatose, with bursts of insanity...

Meanwhile in Personal Computer World, the anonymous author of "Chip Chat" continues to make strange remarks about Diego Rincon of Computing Today. Apparently, Diego has the same opinion as myself, when it comes to the hideous new binders now being sold by "O Prawn! Sell more product" (anagram 8,8,5). Their first binder was excellent, but under new management they sell a custard coloured thing in which the magazines are retained with string. I have considered "Gout Tidy, No Camp" second only to INMC80, ever since they started sending me free copies. All I have to do, to earn this privilege, is fill in a questionnaire each month, criticising or praising the various articles, and the next month's issue arrives like magic. (I make these anagrams up myself, you know. What is needed is a version of Kerr Borland's Corrector program to speed up the process.)

-----

QUICK HINT

-----

When you are testing a machine code program on your Nascom, and have a BASIC ROM that gets jealous, it is a good plan to put HALT (and you should know that is 76) in all the unused RAM above your program. Then when your perfect program runs away, you won't have the BASIC initialising itself in the area 1000H to 10D6H.

(Ed. - Alternatively put E7 in the unused RAM, this will give a register display should the program crash, so you'll know where you've gone to !)

"SORTING IT OUT" SORTED OUT (see last issue)

-----  
Line 100 should read as follows: 100 IF A(I)<X THEN I=I+1:GOTO 100.

SUBERSIVE THOUGHT

-----  
I can think of no reason why it should not be possible to connect a Nascom to one of the soon-to-be-legal Open Channel radios. If there is no reason (experts, please advise!) which channel shall we grab (sorry, allocate) to ourselves?

ISN'T BASIC WONDERFUL ? SECTION

-----  
Before poor Marvin became ill, I managed, with the talent I so often display, to produce an error message from the BASIC that doesn't appear anywhere in the documentation; it was an MO error. (Ed. Missing Operand.) There's a whole new field here, which goes far beyond unknown opcodes - FS error, for instance, is Forgot to Switch on error.

My younger brother, who has an M\*14 for sale (Richard Beal could make a Nascom 1-2-14; or any offers ?) whilst writing the ultimate starship simulation, used an amazing array, which at one point in the program was accessed as follows:

```
X9=A(A(1,2,A(1,2,3)),A(1,3,4),F(I,0,2))
```

No error message whatsoever was produced, and I am told that it is obvious what this does, but I didn't understand his explanation at all. He tells me that any M\*14 owner would immediately understand the full implications of such a thing with no trouble at all....

I propose a competition for users of ROM BASIC, to see who can enter the longest line of BASIC using single (i.e. three at a time !) key entry of reserved words. The line must make sense, should preferably do something useful, and must produce no error messages when run. A hint - RESTORE is the longest word you have available, so you'll need to use it a lot to win.

LATEST EXPENSIVE ACQUISITIONS.

-----  
I bought myself (or rather, Marvin !) an 8 amp power supply, and haven't seen our usual postman since he carried it up the hill. Any offers for the now redundant 3 amp unit? The 8 amp PSU is so heavy, it makes the Blue Oyster Cult look like Tiny Tim (not a programming language). I suspect that if you put two of these power supplies together, a black hole would form instantly.

Something to watch out for, which I spotted in time, when you get yours (not if, when. Computing is the most serious addiction known to mankind and Heloise Fortran). If the post office drop it on the end with the huge heat sink, the connections to the power transistors can bend until they touch the casing. The casing will spring back into shape, but if the thing is switched on in this state, I fear it is more likely to form a super-nova than a black hole. Look before you plug it in and save money!

Another recent purchase is an EPROM board from the Merseyside Nascom Users Group. This has eight sockets for 2708's and one for the BASIC ROM. It works perfectly at 2 MHz, is delivered fast, and costs 46.00. My only criticism is that it doesn't have the connections required between lines 16 and 17, 19 and 20 for the Nasbus, that form the interrupt daisy chain, but you can easily connect them yourself, or put it at the far end of the motherboard. I admit it is not as wonderful as the Interface Components board, which has bells and whistles, but that one does cost more....

I have not been able to test my other new board, an I/O board, due to the ongoing negative hardware situation. More in another episode, perhaps. In the meantime, would someone please send me a copy of any pages there may be in their documentation for the board, with a page number higher than 8.

IS PASCAL NECESSARY ?

-----  
I put that sub-heading in to stir up some correspondence in defence of the people who write long series of articles in the glossier magazines about structured programming. Their second article tends to be full of rules you must not break, or your programs will not be structured, and you won't be a "real" programmer. In fairness, it must be said that the first article in such a series is usually full of excellent advice such as design your program before you start to write it, make it modular and test each routine as you write it. This is good; and in my usual "do as I say, not as I do" style I recommend it. The rules that appear in the second article are odd though. Why must we never use GOTO ? Why is a "computed GOTO" doubly damned ? Will the sky really fall down if I continue to write self-modifying code using undocumented opcodes ? I think (and remember, I'm not getting paid for this !) that you get paid more for two articles than you do for one.

My greatest software achievement so far would have had to be about 6K in length if I had obeyed all those rules. It is in fact 1.5K, modifies itself when necessary, uses IX and IY as four 8-bit registers, and won't work if relocated carelessly; you have probably seen it advertised.

SOFTWARE IN PROGRESS.

-----  
Now that I have all this time, and can't sit zapping Klingons all night, I have several projects under way. One of the least useful is an extension to Nas-Sys, in the form of a Genuine People Personality. This should not be too tricky, all you have to do is put a new routine address table together, change the table address in the workspace RAM and add suitable messages before (after ?, during ?) the normal routines. To prevent this from becoming boring, they shouldn't always appear, and should vary. The counter-timer chip on the I/O board should come in handy here. It could also be used to produce a message if there is no input for, say, five minutes. "I'm not making you depressed, am I?" would be probable. I hope to make Marvin very much more like his Radio 4 namesake....

I have also restarted work on an improved Pilot interpreter, for use with Nas-Sys, but this is at present halted. I'm waiting for my copy of the Scelbi Z80 Software Gourmet Guide and Cookbook, so that I can adapt (pronounced "pinch") the floating point routines it includes. A friend showed me this book, and I recommend it for any Nascom user who needs a good introduction to programming in machine code. My original interpreter only had positive integer arithmetic; in case you think that would suffice, you will be glad to hear that I recently lent a copy of it to a friend in the hope that the thing would get properly written out and sent in for the library. The new version will also have a G: command for graphics, and some fancy screen formatting, if I can get it to work.

I liked the article in Practical Computing, August issue, about a program to play "Adventure". I have been giving some thought to how I would go about this. The printed version is a program in "pseudo-code" which uses several tables of information to play the game. To change the game you change the tables. My own, very slightly different, approach would be to define a new language, with statements of various types for the elements of the adventure: these would include place statements, artifact statements and so on. Then all (ha ha) that is needed is an interpreter for the language. Then I looked at how much data would need to be stored, and came to the conclusion that the thing would need disks to be worthwhile. The coward's way out, and if one of you proves me wrong, I'll be delighted. (Official receiver, please note that I am waiting for the chance to buy myself a disk controller board, not to mention the programmable graphics unit.)

THINGS SOMEONE SHOULD BUILD.

-----  
These are both light-years beyond my design ability, but I will write the software for them in exchange for one of each, should some genius manage to design them. Sorry, Heloise, but I offered first.

It must be possible to make a colour graphics board that will display more than two colours at a time in a single character space. And is there any reason why the user shouldn't have control over the brightness of each pixel ? Ideally the resolution should be as good as that of a colour television, of course. My rough calculations indicate that it will need about 500K of RAM, and digital to analogue converters capable of about 25,000,000 operations per second.

Down to earth a bit for the second idea ! In the July 1980 issue of Personal Computer World, R.M.Yorston described how he added a Z80 and 8K of RAM to his 6800-based system. A board like this, to Nasbus specifications, would really amaze the owners of lesser computers. To the Nascom, the board would appear to be (say) 4K of RAM. Data passed into this RAM area could be processed while the Nascom was doing something else. Several such boards in a system would make possible some really fast processing. I could run my star ship simulator in real time ! Oh, what a givaway...

WHATEVER HAPPENED TO ? SECTION.

Whatever happened to:

- (i) the Christmas game competition ?
- (ii) my entry for (i) ?
- (iii) all the maze solving programs I thought you would all have written by now, surely it's not too difficult ?
- (iv) my list with item (iv) on it ?

AND FINALLY, THE AREN'T COMPUTERS WONDERFUL ? SECTION

Here we see that a certain underwear manufacturer, from whom I will never buy a pair of gloves again, could use a little more care in the programming of their computer....

*Winter is just around the corner—once again—are you fully prepared for it?*

AD/PDD/CLA

DEAR MR [REDACTED],  
 REMEMBER LAST WINTER? HOW THE SNOW LAY FOR DAYS IN THE ILMINSTER AREA AND HOW THANKFUL YOU WERE FOR YOUR PANCY-KNIT LADIES VESTS. WELL - WINTER IS ONLY JUST AROUND THE CORNER AND IT WON'T BE LONG BEFORE YOU ARE WONDERING IF SNOW WILL BE FALLING

# Ex-stock

BACK ISSUES & INMC80 PROGRAM LIBRARY - NEW ADDITIONS

The following programs have now been added to the Program Library.

Address all orders to: INMC80,  
 c/o Oakfield Corner,  
 Sycamore Road,  
 AMERSHAM,  
 Bucks. HP6 5EQ.

PLEASE NOTE: Interface Components Ltd. are very kindly letting us use their address as a Post Box. They cannot accept phone calls on our behalf. If you have any queries please WRITE. Ta.

NASBUG Programs

- T44 Black Box V1.1 by G.M.Clarke 0.90  
A game based on the cloud chamber experiment to discover atomic structure.
- T45 Universal Chess Clock V1.1 by G.M.Clarke 0.60  
Provides a chess clock for Standard, Allegro, and Lightning chess games.
- T46 Factorial V1.1 by J.Haigh 0.50  
Evaluates factorial of N up to 900 digits precision on an unexpanded Nascom 1.
- T47 Powers of 2 V1.1 by J.Haigh 0.40  
Evaluates powers of 2 up to 2 to the power of 4250 on an unexpanded Nascom 1.
- T48 Chomp V1.1 by J.Haigh 0.90  
A NIM type game, played on a rectangular array of points.

2K TINY BASIC Programs

- TB1 Simple Line Editor V1.1 by J.Hill 0.20  
Assembly language routines to add line editing functions to 2K TINY BASIC.
- TB2 String Package V1.1 by J.Hill 0.70  
Assembly language routines to add string handling to 2K TINY BASIC.
- TB3 Stock Exchange V1.1 by J.E.Hawkins 0.15  
Play the Stock Market.

POSTAGE and PACKING

UK customers please add 30p for the first program, plus 5p for each additional program. For overseas orders these charges are 60p and 15p. All cheques to be payable to me, oops, INMC80.

BACK ISSUES

All INMC and INMC80 back issues are available ex-stock. Because of reprint costs we have reluctantly had to increase the price of these:

- INMC 1 50p
- INMC 2-7 1.00 each
- INMC80-1 1.00

Postage. UK orders add 35p for one issue, plus 10p for each additional issue. Overseas orders 70p and 20p respectively.

# Kbd mods

MODS FOR DEMON TYPISTS

by Derek Brough

Having, for the umpteenth time, cursed the lack of a left hand shift key on the N1 keyboard, I decided to do something about it. The mods were quite simple even though I only possessed two hands, and whats more it worked (second time) much to my surprise.

First the theory - briefly the keyboard works by having an 8 x 6 matrix of keys with eight driver wires which are pulsed in sequence under software control. Six wires sense which key (if any) is depressed (ready to prescribe some Valium no doubt). To fit a second parallel shift key we must wire the new key in series with both the drive and sense wires of the existing shift key. The wires must also be connected with the correct polarity otherwise the sensed pulse will be upside down (which is why my first attempt didn't work).

Now for the painting by numbers:

- 1) Obtain the spare Licon key and blank keytop (price 1.20 (I think) from Interface Components, Amersham).
- 2) Look at the underside of the keyboard pcb, placed with the space bar to the top. It will look something like diagram 1. (I have added the letters to the drawing. The capital letters refer to the key pins, the small letters to the tracks.)
- 3) Carefully drill 4 1mm holes immediately to the left of the 'Z' key, on the same horizontal line. Dimesions as per diagram 2.
- 4) The right hand hole will be too close to track 'p' for comfort so cut out a small section of this track with a sharp knife and replace it with thin insulated wire to avoid pin 'C', diagram 3.
- 5) Cut tracks 'a' and 'b' near pins 'A' and 'B'.
- 6) Connect pin 'B' to pin 'E'.
- 7) Connect pin 'E' to the free end of track 'b'.
- 8) Connect pin 'A' to pin 'C' and pin 'D' to the free end of track 'a'. These two wires should be lightly twisted together to reduce noise.

If you have done it all correctly, the new shift key should work.

Ed. Please note that Nascom keyboards carry a one year guarantee from Licon, which is nothing to do with Nascom. The above modifications will void the keyboard guarantee.

## SORRY.

CORRECTION : I/O BOARD REVIEW

In the review of the Nascom I/O board in INMC 80-1 (the last issue), it was stated that the UART did not have the ability to send and receive simultaneously at different speeds. This was incorrect. The UART speed select links allow different send and recieve speeds to be used. This was an oversight on the part of the reviewer, for which, we apologise.

Nascom have also pointed out the despite our INMC award for incomprehensible documentation, no complaints have been received. (Yet !!)

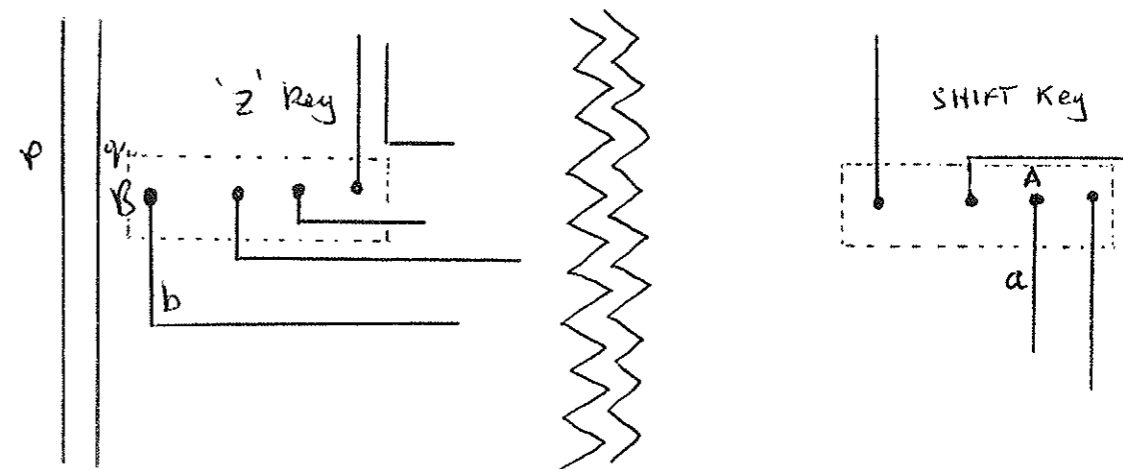


Diagram 1

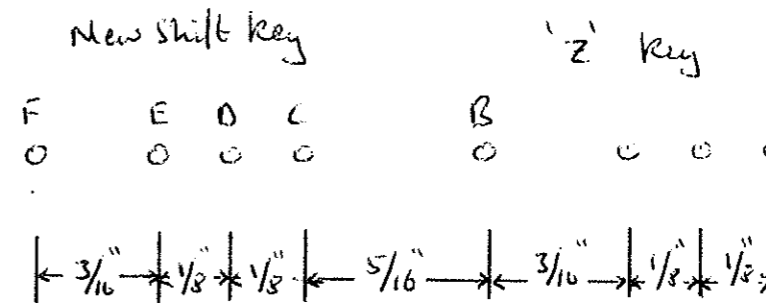


Diagram 2

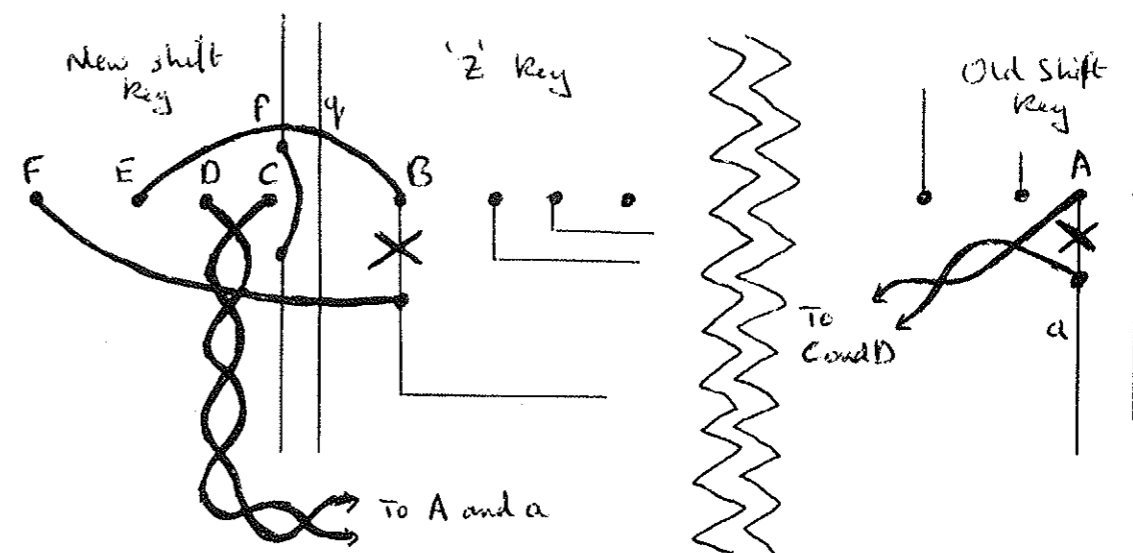


Diagram 3

(New wires shown thick)

## Back to Basics

The other evening an envelope containing an article (and a Naspen tape of it) was shoved through my letter box. The article was yet another review, but this time on a most unlikely subject. A computing holiday !!! So if you have already recovered from this years holiday, and are thinking about next years, this might be worth consideration.

On Holiday in Basic  
=====

by R. White

As a computer addict of some eighteen months standing, the thought of spending a week away from the keyboard for a mere family holiday was indeed hard to bear. When a fellow addict produced a brochure of 'Marlborough College Summer School 1980' all our holiday decisions were made in one fell swoop. The Summer School was offering a range of courses to suit all possible interests: Yoga, Bridge, Swimming, Painting, Walking, Languages, Philosophy, History and many others ... including Computer Programming. The children were soon convinced the 'Children's Activities' were what they always wanted, while my wife realised at once that the 'Pre-history of Wiltshire' had an irresistible fascination for her inquiring mind. So we went.

We arrived on a Sunday evening and installed ourselves in our twin-bedded room, the children being nearby in dormitories. The accomodation was adequate if a little spartan compared with a modern hotel. The food however was excellent. Breakfast, lunch and dinner were all served in the main college dining-hall on a self-serve basis, with an excellent menu at each meal. There were several bars including one within the dining-hall itself which was about the only extra expense we had during the whole week - absolutely everything else being provided, including the never-to-be-forgotten doughnuts served with morning coffee each day.

On the first morning the whole of the Summer School, about five hundred people, assembled for a few minutes talk introducing the various Tutors, many of whom were staff at Marlborough College itself. The computer programming course was to be in the hands of the College computer System Manager, J. Marcus Gray, and the language was to be Basic !

After introductions all round it became apparent that all sixteen members of the course were from widely differing age groups and interests. At least four were still in full-time study where they had picked up an interest in computing at school or college and now wanted to learn more. There were two medical men, both G.P.s, who were interested and even concerned as to how computers may invade their sphere. They had no computing experience whatsoever and had perhaps only read one of the books on Basic that had been mentioned in the 'suggested reading' list before the course. There were two or three hobbyists like my fellow Nascomite and myself, plus one or two who ran small businesses enquiring into the possibilities of the machine for them. Altogether we were a very mixed bunch.

In the lecture room itself there was no sign of any computer - just desks and a blackboard. There was one rather nasty moment when our tutor began to talk about the college computer itself and produced from his briefcase - horror of horrors - a ZX80 ! The thought of a week with a ZX80 between sixteen people ..... But this was only to make the point to the inexperienced of us how small(!) the home computer has already become.

The general pattern of the course was an occasional lecture with blackboard and notes etc., and then adjournment to the college's eight-terminal computer for practice on one's own with suggested experiments. There was always help available both from our tutor himself and from his assistant, a very enthusiastic and helpful sixth-former at the college. Thus there were two students per terminal and two sources of help when in trouble - a very satisfactory arrangement.

The first two days of the course took us through the simple statements - PRINT, INPUT, IF, READ etc., and a discussion of numerical variables and arrays. We two Nascomites found considerable difficulty in getting used to the rather intolerant disciplines of the multi-terminal computer after having been used to the screen editing and other short-cut luxuries of our micros.

Although the official lecture periods ended at 4.15 p.m. we were often still typing until dinner at 6.30 when we reluctantly broke off to eat and perhaps join in an evening activity. This could have been a concert of music, a walk on the downs, a guided tour of the College or, on the first evening, a sherry party. Generally the evenings tended to end in the bar, possibly still talking and thinking in Basic and knowing that if required we could always use the computer until 10.30, usually with help still available if needed.

By mid-week we were into strings and files. Here we had the great advantage of having access to some of the College files already stored on disks. Thus without having to type in anything except program instructions we could practice Bubble Sorts, make new files and investigate search routines.

All too quickly the week came to an end. By now we had filed away quite a few interesting short programs which we were able to print out and take home for further study. The very first thing we each had had to do on that first Monday morning had been to write down on paper our individual aim in taking the course. Mine had been quite simply to learn to think in Basic without worrying about registers, bytes and memory locations within the machine. At the end I did feel that I had achieved that aim and had done so in a most enjoyable manner.

As to the rest of the family, my wife is as hooked on Avebury and Sarsen stones and the like as I am on Basic, while the children said their 'activities' were just, "Brill"(iant). One thing we all shared at the end was tiredness but at least it was coupled with a sense of achievement.

Ed. We have since heard that Marlborough College MAY be running computing courses in different levels of Basic and in another language next year. For details, write to:

Marlborough College Summer School,  
Marlborough,  
Wilts. SN8 1PA.

## Ads.

FOR SALE  
=====

Nascom 2 with 32K, built and fully working in a micro case. Naspen. Disagreement with solicitor forces sale, Nascom hardly used. 400.00 ono. Phone ERITH 32835.

NAS-SYS 1 on ROM, straight out of Nascom 2. Have upped to NAS-SYS 3. Full documentation. No sensible offer refused. Phone PETERSFIELD 4059, evenings.

Nascom 2, 32K, ZEAP in EPROM, 4K Toolkit, runs 2400 baud. Manual, tapes and programs. Keyboard case. 470.00. Can deliver West Country or London. Phone 0326-72207, evenings or weekends.

Nascom 1 with NAS-SYS and PSU. Purchased ready built 5 months ago, total cost 195.00. Full documentation. Will accept 160.00 ono. Phone Brian Reece - Hornchurch 74388.

## Z80 made simple

THE KIDDIES GUIDE TO Z80 ASSEMBLER PROGRAMMING

D. R. Hunt

Part: The second.

Things to do, and what to do it to  
(Op-codes and operands)

By now, anyone who actually bothered to read part one of this series will (of course) be fully proficient in the doubtful business of counting in HEX. Those who are still wondering about the necessity, need only look at the object listings of programs which sometimes find their way into this newsletter to realize that they are missing out on something by not understanding what it's all about.

By the way, to digress (so soon already), object listings are the name given to those columns of HEX digits which appear when you use the Tab command on the Nascom. I don't know why it's called 'object', it doesn't seem important to know, and I've never exposed my ignorance (till now) by asking anyone. The term 'object listing' refers to the machine code instructions (in this instance displayed in HEX) tabulated in some form that the programmer can read (it may also refer to a form that the computer can read). Essentially, it is the machine code instructions to the compute. It's not confined to the Nascom form of display where each line starts with an address, followed by eight bytes displayed as two HEX digits each. For that matter, given that one byte (eight binary bits) only needs two Hex digits to express it, the space between each consecutive byte is not really required. Likewise, provided we know where the start address is, the address given at the start of each line can also be deleted. The result could be something like this:

```
4010 (start address)
DBCDF23F30F0C9DDB0217D0DB01C9CD5F
4030072AB04722AE47C92AAE472B22AE
etc.
```

```
instead of the more familiar
4010 D8 CD F2 3F 30 F0 C9 DB
4018 02 17 D0 DB 01 C9 CD 5F
4020 40 30 03 2A B0 47 22 AE
4028 47 C9 2A AE 47 2B 22 AE
```

The first example is equally good 'object code', a lot more compact, but a darned sight less readable. It's call 'Intel format' by the way, and is quite common in Stateside computer magazines. Thank goodness it's not used much in this country.

So what do these numbers mean? Well to understand that we must first look inside the Z80. Now for those who don't know, the Z80 is what is known as the CPU which means Central Processing Unit, and at the bottom of it, it's this chip which does the work. Despite the size and apparent complexity of the Z80, it really isn't a frightening device. For a lot of purposes it can be treated as a 'magic black box', we don't need to know how it did something, only the net result of what it did. With any CPU, real understanding comes through practice and not 'book learning'.

The second thing to realise is that up till now we have generalised, and talked of HEX and object code. From now on we are going to be specific, and deal with the Z80. Z80 instructions mean something to a Z80, much as english means something to me. (Don't believe you. Ed.) Portugese (any Portugese speaking persons amongst my readers will, I'm sure, forgive the reference) is totally foreign to me. The form of letters used is, in the main, the same, and a Portugese speaking person is a member of the human race (very much closer related to me in fact than the relationship between one CPU and another), but the language is different, and to me meaningless. In the same way, if Z80 HEX code were presented to, say, a 6502 CPU, the 6502 would be totally lost.

Not all CPUs are all that different to each other (isolated by language), there is some compatibilty in this world. An 8080 CPU and a Z80 for instance, may be likened to someone with a limited and simple vocabulary speaking english to me. I understand the words perfectly, but as the speaker only has command of roughly half the vocabulary I have, it takes longer to get the meaning across. Contrarywise, when a Z80 speaks to an 8080, its vocabulary must be restricted to the 'intellectual level' of the 8080. So beware not all object code has the same meaning. With some practice, it is possible to determine which is good Z80 code from code for some other processor or from rubbish.

Having said that this business is not 'book learning' stuff, I must follow on by saying that the Z80 Technical Manual is vitally important. Read and digested in little doses, it becomes quite understandable. If you read the Technical manual from end to end, once, and understand it all, then sell your Nascom and go and buy an IBM 370 instead, you obviously need something meatier to chew on.

The Technical Manual goes on about the registers and describes in great detail how they are interconnected. Think of the registers as railway sidings, some connected to the two main lines, others connected to one or other of the main lines. The Technical manual gives you a nice map to the whole 'goods yard'. On one main line there are eight main lines running parallel and on the other there are 16 main lines. Each siding has eight parallel tracks connected by points to the 8 wide main line, whilst pairs of registers may also be connected to the 16 wide main line. Trains in the form of eight wagons each, one on each parallel line, move along the tracks of the 8 wide main line parallel to each other, and may be directed through points from the main line into the sidings. Likewise, groups of wagons, 16 wide move along the 16 wide main line, and may be moved to a selected 16 wide siding, composed of a pair of 8 wide sidings. That's all the load (LD) group of instructions do. If we want to load the 'A' register, eight parallel bits are placed on the (8 wide) data bus, and are directed through switches into the 'A' register. Similarly, sixteen bits of data may be moved along the (16 wide) address bus to or from either the register pairs, or special purpose registers such as the Program Counter. There is one important thing to know about the load group of instructions though; when an instruction such as LD A,B (that means load the 'A' register with the contents of the 'B' register) is encountered, unlike trains, the contents of 'B' is COPIED to 'A', not moved to 'A', so the contents of 'B' remains unchanged.

There are quite a number of registers in the Z80, we'll start with just a few of them. Perhaps the two most important are the 'PC' (Program Counter) register and the 'A' register. Starting with the 'A' register, this is also refered to as the 'accumulator', for reasons I hope will become obvious in a few moments. It is here that it is all done, all arithmetic and logical operations are performed in the accumulator. For instance, if we wanted to add two numbers together, we would put one in 'A' and one in another register, say, 'B'. When the addition is performed, the answer will be in 'A' ('B' incidentally will be unchanged), hence 'A' accumulates the answer.

The 'PC' register is a special purpose 16 bit register, and its job is to keep an eye on the address where the processor is to get its next byte from. In many senses, the 'PC' register is purely automatic, every time the processor fetches a byte (as it must do to know what to do next), the 'PC' register is automatically incremented (increment means to add 01H to it) to the next byte. When the processor is ready for the next byte, the contents of the 'PC' register are placed on the address bus, and this forms the address of the next byte to be fetched.

There is another special purpose register in this case uniquely associated with the 'A' register, and this is the 'F' register, or 'Flag' register (also occasionally called the Status Word Register). Back to the trains. The 'F' siding is at the end of the 'A' siding (but also connected back to the main line). If, on addition, there is a 'carry' from the 'A' siding, then the extra wagon is shunted into a specific location in the 'F' siding reserved for the 'carry' from arithmetic and logical operations. So if our addition overflowed, then the 'Carry flag' would become set. There is also a flag which will be set if the result of the calculation becomes zero, this is known as the 'Zero flag'. There are four other flags as well, we'll deal with them another time.

An important point has been made here, it's the flags which give the processor its intelligence. By testing the condition of the flags a decision can be made. For example, suppose we wished to count down from 40H to 0 (remember the 'H' means HEX). We could load 'A' with 40H, and then go into a simple loop which subtracted 01H from the accumulator then tested the flags to see if the 'Z' flag had been set. If the 'Z' flag was still unset, then the program would go through the loop again, round and round until such time as the 'Z' flag did become set. Enough reading, let's prove it. We'll write a little program that does just that. I expect you to read the Software Manual and discover how to use the 'M' command, and I'll write the code in two ways, firstly as an object code listing, and secondly as an assembler (or source) listing. I'll explain a bit as we go along, but not too much (I believe in making my readers work, I had to learn the hard way, so why not you). We'll only count from 10H to 0 otherwise it will become tedious.

Object code  
 0C80 3E 10 3D 20 FD 3E 22 00

Source code  
 0C80 3E 10 LD A,10H Load the 'A' register with 10H  
 0C82 3D DEC A Decrement A, meaning subtract 1  
 0C83 20 FD JR NZ,-03H Jump no zero back three places, relative to the next  
 instruction (I'll explain later)  
 0C85 3E 22 LD A,22H Load the 'A' register with 22H  
 0C87 00 NOP Do nothing (for enthusiastic key pressers)

Now what we have said is load the 'A' register with 10H. Decrement the 'A' register by 1 (decrements are always by 1). Test the 'Z' flag, and if this says 'Not Zero' then move back three places from the start of the next instruction.

Why from "the start of the next instruction"? Well, when the processor meets an instruction the first thing that instruction does is to tell the processor how many bytes in that instruction. There might be up to four bytes in an instruction. The processor has to read each byte in turn, and by the time it has done this the 'PC' register has already been incremented so it is already pointing to the next instruction. Don't forget it's the job of the 'PC' register to 'keep tabs' on where the processor is to get the next byte from. So in the case of a jump like this, the 'PC' is already pointing to the next instruction by the time it discovers it should have in fact gone backwards not forwards. We didn't deal with negative HEX numbers in part 1, just take my word for it FDH is actually -03H. If the 'Z' flag said the result was zero, then the program would 'drop though' the jump instruction to the next instruction (which it was already pointing at), which in this instance is load the 'A' register with 22H. I chose 22H to prove that it was the next instruction, and not a magic number the CPU thought of itself. You might like to change the number just to prove I'm right. The NOP (No Operation) at the end was thrown in in case the 'Enter' key was pressed once too often. This program has no 'END', so immediately the processor found the next byte it would do unpredictable things, as we haven't programmed beyond this point. The memory would be full of garbage from the time the computer was turned on.

Now having got the program in using the 'M' command, go and read up the 'S' command. Notice how the register display is presented when using single step. It might help if you stuck a piece of masking tape across the top of the monitor screen, with the registers written in order across it, saves continually referring to the book. Away we go!! Type S 0C80, and provided you did it all right, then a register display will appear on the screen. What's more, the 'A' register will contain 10H and the 'PC' will be pointing at the next instruction (0C82H). Hit 'Enter' again, and the 'PC' will advance one, pointing to the next instruction, and the 'A' register will be decremented by 1. Also at this point the flags may have changed (they were in an indeterminate state when we started), now they definitely show the result of the decrement. Hit 'Enter' again, and the 'PC' will have skipped back to the address of the decrement instruction ready for the next time round. Keep hitting the 'Enter' key, watching the 'A' register and the flags. When the 'A' register goes to zero, notice the flags change, the 'Z' flag comes on. Take one more step, and notice nothing happens, but this time the 'PC' instead of pointing to the decrement instruction, now points to the next instruction, the load 'A' with 22H. Nothing happened because the processor still has to perform the test on the flags. You know they've changed, but the processor still has to test them and jump (or not jump as the case may be). One more step, and lo, the 'A' register contains 22H. There you are, your first assembler program. It didn't do much, and it's only safe to single step the program, as it doesn't stop when its finished, but you must have learned something. If you didn't understand what happened, perhaps a Basic analogy will help.

10 A=16  
 20 A=A-1  
 30 IF A<>0 THEN 10  
 40 A=34

Having manipulated one register and seen the effects. Two things remain to be explained in this episode. The first, a brief explanation of some of the other registers, and secondly, how the processor knows what to do.

The other main registers of the Z80 are the 'B', 'C', 'D', 'E', 'H' and 'L' registers. As mentioned previously these 8 bit registers are also organised so that they may also be loaded as pairs so that each of the 8 bit registers paired together give the following 16 bit registers, 'BC', 'DE' and 'HL'. The 'A' and 'F' registers are also paired together, although because of the special nature of the 'F' register, the 'AF' pair can not be treated as a single 16 bit register. Presumably there is no 'G' register because the name of the other of the pair has already been allotted to the 'F' register.

When treated as 8 bit registers, the registers act as temporary storage places for data or can be used as counters. They also have limited arithmetic capability, they may be incremented and decremented, and major changes of status will affect the 'F' register just like the 'A' register. There are perhaps more 8 bit registers than would be really necessary, (memory locations could be used in place of registers), but as it takes the processor markedly less time to get frequently used data from an internal register, using them speeds things up.

Perhaps more important is the 16 bit capability. Now addresses are 16 bits wide, and the 16 bit register pairs have access to the address bus. The 'HL' pair acts rather like the 'A' register, in that limited arithmetic can be performed using it as a 16 bit accumulator and the results affect the flags accordingly. This means that addresses can be calculated directly which saves a lot of time and effort. The remaining registers, the 'SP', 'IX' and 'IY', 'I' and 'R' all perform special functions and we'll make use of them as we come to them.

Just in case the above set of registers aren't enough for you, there is also a complete duplicate set of the main registers which may be switched in at will. Phewww !!!

Lastly we come the title of this piece. Op-codes and operands. How does the processor know what it is about? Now we've talked about the 'PC' register pointing at each byte of the program in turn. Provided the 'PC' is pointed at the first byte of a program, and provided that program is correct, it can't go wrong. It works like this: each instruction may be between one and four bytes long. The first byte contains important information for the processor, this byte is called the 'operation code' or op-code for short. Firstly, it tells the processor how many bytes there are in the instruction. Then back to our railway analogy, it sets the points. This first byte operates the switches which route the data round the processor. In some instructions (the 4 byte ones), the op-code is two bytes long as there is too much for the op-code (or instruction) decoder to handle at once.

The bytes that follow are known as the operand, that is the bytes that need to be worked on. In some instances the operand may be data, like the LD A,10H instruction (above), the 10H is data. In other instances the operand might be an address offset like the JR NZ instruction. In many instances there is no operand at all, as all the information is implied like the DEC A instruction, where 'A' is implicit within the instruction, and a decrement simply means subtract 1 from it. The processor is clever enough to handle all that in a 1 byte instruction.

It can be seen that if instructions are given to the processor properly, then by the time the processor has finished with one instruction it is already pointing at the next (or has been pointed at another by a jump or call instruction). More often than not, with novice programmers, things start going wrong when a jump or call goes to the wrong place, or the program is allowed to run on into a data or unprogrammed area. The poor little processor doesn't know something is wrong, it faithfully carries on and tries to interpret the instructions it finds. Suppose (because you programmed it wrong) a jump instruction sent the processor to the middle byte of a LD HL,0123H instruction. That looks like this:

21 23 01

Instead of loading 'HL' with 0123, it will first increment what is already in 'HL' because 23H is the instruction to increment 'HL', then, seeing 01H, it will think that that is the first byte of a LD BC instruction so the next two bytes will be interpreted as the operand for the LD BC instruction, which they are not. We needn't go on, the program is blown already. From then on the processor will run wild, actually, 'wild' is the wrong word. What the processor does is quite logical, if you could only find where it had gone. The processor was only doing what it was told, it can't know that that is not what was intended. Really it's the program which ran wild. The usual result of all this (determined by Sod's Law) is that if the program was not saved on tape, it will write rubbish all over it, and then try to execute the rubbish, and so on, and so on. This is what is known as a 'program crash', and one thing will be impressive, that is the speed at which it can all happen. After hours of painstaking typing putting a program in, there is nothing so humbling as to conduct a postmortem on the program, that, because of a single typing error the processor managed to completely 'scramble' in a few thousandths of a second. You wouldn't have had time to think about hitting the reset button, yet alone do anything about it.

So what have we learned. Object code listings. A glimpse at the insides of the Z80. How the registers are arranged, and op-codes and operands. In the next episode we will go on to discuss the workings of the other registers, and take a closer and more detailed look at some of the instructions.

One final bit of encouragement. This stuff is not easy, it's rather like walking through a fog, and not even knowing where you are supposed to be going. Even now, if you are still totally lost, just keep 'plugging away at it'. In my experience, and I am certain I am not unique, at some point, after one or two false starts, there will come a small glimmer of understanding. Suddenly everything becomes clear. All the facts fall into place, and although you may be wrong about some of the details, the 'core' or 'heart' of the problem is exposed and to your surprise, you will find that you had already learned most of what there is to know. You will kick yourself for not having grasped the concepts earlier.

# Assembler

## ZEAF Z80 Assembler - Source Listing

```

0010 ; *****
0020 ; *
0030 ; * LOAD AND TAB *
0040 ; * (with checksums) *
0050 ; *
0060 ; *****

0080 ; Intended for use with later revisions
0090 ; of NAS-SYS, where the 'L' command has
0100 ; been omitted, and the 'T' command has
0110 ; been amended such that checksums are
0120 ; not output.

0140 ; These routines are intended for use,
0150 ; where programs printed with checksums
0160 ; are to be re-entered using the 'L'
0170 ; command.

0190 ; These are revised versions of the code
0200 ; used in NAS-SYS 1, and are totally
0210 ; relocatable.

0000 0230 ORG 0

0000 0028 0250 ; Restarts used in NAS-SYS.
0000 0030 0260 FRS EQU 28H ; Print the following string.
0000 0030 0270 ROUT EQU 30H ; Print a character.

0000 005E 0290 ; Routines used in NAS-SYS.
0000 0066 0300 MRET EQU 5BH ; Return to monitor.
0000 0067 0310 TBCD3 EQU 66H ; Display HL and add to C.
0000 0068 0320 TBCD2 EQU 67H ; Display A and add to C.
0000 0069 0330 B2HEX EQU 68H ; Display contents of A.
0000 006A 0340 SPACE EQU 69H ; Output a space.
0000 006B 0350 CRLF EQU 6AH ; Output a carriage return.
0000 006E 0360 ERRM EQU 6BH ; Put out 'Error' message.
0000 0079 0370 RLIN EQU 79H ; Load line into ARGs.
0000 007B 0380 BLINK EQU 7BH ; Get an character in A.
0000 007C 0390 CPOS EQU 7CH ; Pick up first char on line.

0000 0008 0410 ; Symbols used by NAS-SYS.
0000 001E 0420 BS EQU 0BH ; Backspace symbol.
0000 000C 0430 ESC EQU 1BH ; Home cursor.
0000 000D 0440 CS EQU 0CH ; Clear screen symbol.
0000 000D 0450 CR EQU 0DH ; Carriage return symbol.

0000 0C0C 0470 ; NAS-SYS workspaces used
0000 0C0E 0480 ARG1 EQU 0C0CH
0000 0C10 0490 ARG2 EQU 0C0EH
0000 0C29 0500 ARG3 EQU 0C10H
0000 0C29 0510 CURSOR EQU 0C29H

0530 ;*****

```



```

0000      0550      ORG  0

          0570 ; The TAB routine is executed from TAB,
          0580 ; having activated the appropriate 'X'
          0590 ; routine. The 'from' and 'to' arguments
          0600 ; follow the execute address. The data
          0610 ; output scrolled on the screen.

          0630 ; Get 'from' and 'to' into HL and DE
0000 2A0E0C 0640 TAB  LD  HL, (ARG2)
0003 ED5B100C 0650      LD  DE, (ARG3)

          0670 ; Test if HL < DE, if so continue
0007 B7      0680 TB1  OR   A
0008 ED52    0690      SBC  HL, DE
000A 19      0700      ADD  HL, DE
000B 3806    0710      JR   C TB2

          0730 ; HL > DE, so must be end
000D EF      0740      RST  PRS
000E 2E0D00 0750      DEFB ",CR,0
0011 DF5B    0760      SCAL MRET

          0780 ; Initialize checksum
0013 0E00    0790 TB2  LD   C, 0

          0810 ; Output address
0015 EF      0820      RST  PRS
0016 202000 0830      DEFB " , " ,0
0019 DF66    0840      SCAL TBCD3

          0860 ; Output 8 bytes
001B 0608    0870      LD   B, 8
001D 7E      0880 TB3  LD   A, (HL)
001E DF67    0890      SCAL TBCD2
0020 23      0900      INC  HL
0021 DF69    0910      SCAL SPACE
0023 10F8    0920      DJNZ TB3

          0940 ; Output checksum
0025 79      0950      LD   A, C
0026 DF68    0960      SCAL B2HEX
0028 DF6A    0970      SCAL CRLF
002A 18DB    0980      JR   TB1

          1000 ;*****

0000      1020      ORG  0

          1040 ; The LOAD routine executes at LOAD, and
          1050 ; data is input from the keyboard in the
          1060 ; TAB format, terminated with a 'newline'.
          1070 ; Any error will result in an 'Error'
          1080 ; message. Correct entry will clear the
          1090 ; line leaving only the last line address
          1100 ; entered for easy reference.

          1120 ; Clear the screen
0000 EF      1130 LOAD  RST  PRS
0001 0C00    1140      DEFB CS,0

          1160 ; Get an input
0003 DF7B    1170 LD1  SCAL BLINK

```

```

          1190 ; Strip parity, then if ".", end.
0005 E67F    1200 LD2  AND  7FH
0007 FE2E    1210      CP   "
0009 2857    1220      JR   Z LD8

          1240 ; If a CR, then end of line, so to LD4
000B FE0D    1250      CP   CR
000D 281B    1260      JR   Z LD4

          1280 ; Validate input, allow BS, SPC and 0 - F
000F FE08    1290      CP   BS
0011 2814    1300      JR   Z LD3
0013 FE20    1310      CP   "
0015 2810    1320      JR   Z LD3
0017 FE30    1330      CP   "0
0019 38E8    1340      JR   C LD1
001B FE3A    1350      CP   "1
001D 3808    1360      JR   C LD3
001F FE41    1370      CP   "A
0021 38E0    1380      JR   C LD1
0023 FE47    1390      CP   "G
0025 30DC    1400      JR   NC LD1
0027 F7      1410 LD3  RST  ROUT ; Print the character
0028 18D9    1420      JR   LD1

          1440 ; Convert and check the line
002A 2A290C 1450 LD4  LD   HL, (CURSOR)
002D DF7C    1460      SCAL CPOS
002F EB      1470      EX  DE, HL
0030 DF79    1480      SCAL RLIN
0032 3828    1490      JR   C LD7

          1510 ; Checksum
0034 210C0C 1520      LD   HL, ARG1
0037 AF      1530      XOR  A
0038 0612    1540      LD   B, 12H
003A 86      1550 LD5  ADD  A, (HL)
003B 23      1560      INC  HL
003C 10FC    1570      DJNZ LD5
003E BE      1580      CP   (HL)
003F 201B    1590      JR   NZ LD7

          1610 ; Copy data to memory
0041 2A0C0C 1620      LD   HL, (ARG1)
0044 110E0C 1630      LD   DE, ARG2
0047 0608    1640      LD   B, 8
0049 1A      1650 LD6  LD   A, (DE)
004A 77      1660      LD   (HL), A
004B 23      1670      INC  HL
004C 13      1680      INC  DE
004D 13      1690      INC  DE
004E 10F9    1700      DJNZ LD6

          1720 ; Clear line except address
0050 EF      1730      RST  PRS
0051 1B00    1740      DEFB ESC,0
0053 2A0C0C 1750      LD   HL, (ARG1)
0056 DF66    1760      SCAL TBCD3
0058 DF6A    1770      SCAL CRLF
005A 18A7    1780      JR   LD1

```

```

1800 ; Bad data, put error message and scroll
005C DF69 1810 LD7 SCAL SPACE
005E DF6B 1820 SCAL ERRM
0060 18A1 1830 JR LD1

1850 ; End
0062 F7 1860 LD8 RST ROUT
0063 DF7B 1870 SCAL BLINK
0065 E67F 1880 AND 7FH
0067 FE0D 1890 CP CR
0069 209A 1900 JR NZ LD2
006B F7 1910 RST ROUT

1930 ; Return to monitor
006C DF5B 940 SCAL MRET

1960 ; END OF LISTING

```

ZEAP Z80 Assembler - Symbol Table

0C0CH 0480 ARG1	0C0EH 0490 ARG2
0C10H 0500 ARG3	0068H 0330 B2HEX
007EH 0380 BLINK	0008H 0420 BS
007CH 0390 CFOS	000DH 0450 CR
006AH 0350 CRLF	000CH 0440 CS
0C29H 0510 CURSOR	006BH 0360 ERRM
001EH 0430 ESC	0003H 1170 LD1
0005H 1200 LD2	0027H 1410 LD3
002AH 1450 LD4	003AH 1550 LD5
0049H 1650 LD6	005CH 1810 LD7
0062H 1860 LD8	0000H 1130 LOAD
005EH 0300 MRET	0028H 0260 PRS
0079H 0370 RLIN	0030H 0270 ROUT
0069H 0340 SPACE	0000H 0640 TAB
0007H 0680 TB1	0013H 0790 TB2
001DH 0880 TB3	0067H 0320 TBCD2
0066H 0310 TBCD3	

# BASIC

```

10 REM *****
20 REM * *
30 REM * C A L E N D A R *
40 REM * *
50 REM * By John Waddell *
60 REM * *
70 REM *****
80 REM
90 REM **
100 REM ** Initialise
110 DIM N(12)
120 FOR I=3328 TO 3336 STEP 2:READ J:DOKE I,J:NEXT
130 FOR I=1 TO 12:READ N(I):NEXT
140 GOSUB 730
150 REM **
160 REM ** Restart here and get year
170 N(2)=28:RESTORE 940
180 PRINT "What year do you want ";
190 IF F=1 THEN PRINT:PRINT TAB(3);"('ENTER' for next month) ";
200 PRINT "? ";
210 GOSUB 780
220 IF F=0 AND I#="" THEN 210

```

```

230 PRINT:IF I#<>"" THEN 260
240 IF M=12 THEN Y1=Y1+1:M=1:GOTO 380
250 M=M+1:GOTO 380
260 Y1=VAL(I#):IF Y1=1582 THEN 320
270 F=0:PRINT"Sorry, Gregorian calendar only -"
280 PRINT"after 1582 (continent), or 1752 (UK).";
290 PRINT:GOTO 180
300 REM **
310 REM ** Get month
320 PRINT "Which month (Jan = 1, Feb = 2 etc) ? ";
330 GOSUB 780:PRINT:M=VAL(I#)
340 IF M>0 AND M<13 THEN 370
350 PRINT "There is no month";M:GOTO 320
360 REM **
370 REM ** Reduce year by modulo 400
380 F=1:Y=Y1
390 Y=Y-400:IF Y>400 THEN 390
400 IF Y<>INT(Y/4)*4 THEN 460
410 IF Y=400 THEN 430
420 IF Y=INT(Y/100)*100 THEN 460
430 N(2)=29
440 REM **
450 REM ** Find where year begins
460 D=Y+INT((Y-1)/4)
470 IF Y<=100 THEN 510
480 Y=Y-100:D=D-1:GOTO 470
490 REM **
500 REM ** Find first day of month
510 IF M=1 THEN 550
520 FOR J=1 TO M-1:D=D+N(J):NEXT
530 REM **
540 REM ** Reduce by modulo 7
550 D=D-INT(D/7)*7
560 REM **
570 REM ** Read month, print headings
580 FOR K=1 TO M:READ M$:NEXT
590 GOSUB 730:PRINT TAB(14);M$;TAB(26);Y1:PRINT

```

Ok.

```

600 PRINT TAB(7);"Sun Mon Tue Wed Thu Fri Sat";PRINT
610 X=0:FOR I=1 TO 6:PRINT TAB(6);
620 FOR J=1 TO 7:L=X+J-D:IF L<1 THEN 670
630 IF L>N(M) THEN 700
640 IF L>9 THEN 660
650 PRINT " ";L;" ";:GOTO 680
660 PRINT L;" ";:GOTO 680
670 PRINT " ";
680 NEXT J
690 PRINT:X=X+7:NEXT I
700 PRINT:PRINT:GOTO 170
710 REM **
720 REM ** Clear screen and put up title
730 T$="CALVALVALVALVAL C A L E N D A R(hshhshhshhshh)";
740 CLS:PRINT:FOR I=1 TO LEN(T$)
750 POKE 3026+I,ASC(MID$(T$,I,1)):NEXT:RETURN
760 REM **
770 REM ** Get an input, validate & display
780 I$="";DOKE 4100,3328:FOR J=1 TO 5
790 I=USR(0):IF I=8 AND I#=""THEN 790
800 IF I=8 THEN 840
810 IF I=13 THEN:RETURN
820 IF I<48 OR I>57 THEN 790
830 PRINT CHR$(I);:I#=I#+CHR$(I):NEXT:PRINT:RETURN
840 PRINT CHR$(I);:I#=LEFT$(I#,LEN(I#)-1)
850 J=J-2:NEXT
860 REM **
870 REM ** Machine code input data
880 DATA 31711,-20665,3370,-5664,-13873
890 REM **
900 REM ** Lengths of months
910 DATA 31,28,31,30,31,31,30,31,31,30,31,31
920 REM **
930 REM ** Names of months
940 DATA January,February,March,April,May,June
950 DATA July,August,September,October
960 DATA November,December

```

See INMC80-1  
Page 18.

(hshhshhshhshh)

# Schizo RAM

NAS-SCHIZOPHRENIA

=====

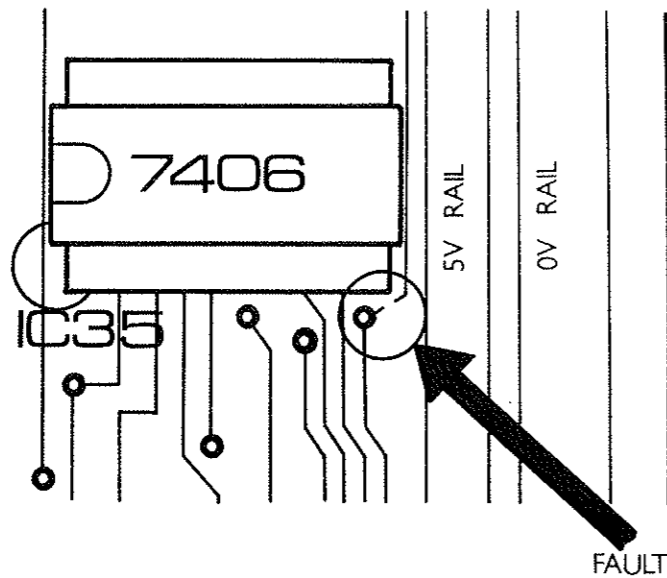
by M. Kuczynski

Having bought and constructed my Nascom 2, it worked first time without a single fault ... or so I thought. Basic gave me 16067 free, and all my small Basic programs worked as well as I wrote them. Then one day INMC 5 arrived, and in it was the program Eliza. About halfway through typing it in, the Nascom 'hung up', so I pressed reset, warm started Basic and proceeded to LIST the program. As soon as the program listing reached the point where it had 'hung', the screen filled with rubbish. "Ah", I thought, "The end marker has got lost, and the Nascom doesn't know where the program ends."

I always heed the advice of Lawrence the Long Haired Wierdo, so fortunately I had saved most of the program on tape already. I reloaded, and carried on from there. At exactly the same place, the same thing happened, time and again. "Damn, 'memory plague'." I thought, but it didn't behave like 'memory plague', as one of 'plagues' symptoms is that it happens randomly, and anyway, the thought of having to string resistors and wires over my memory did not please me. Hours later after trying everything from changing clock speed to swapping memory chips, I had got no further. So I did what I always do in desperation. I hit reset and 'Tabbed' the memory. The endless rows of Hex digits marching up the screen are very soothing to shattered nerves. Odd ?? Sections of code written in page 1 of memory were appearing further on in memory. Suddenly, all became clear, somewhere along the line the memory was being decoded as two parallel blocks of 8K instead of a single block of 16K. In other words my RAM had schizophrenia !!

Checking through the pcb revealed a track with a piece missing from it. I scraped away the green resist and carefully soldered a piece of fuse wire across the break. (Ed. Ecolit silver base paint is also very good, and no heat to damage the tracks.) On testing the board all was found to be perfect. I have since heard of two other Nascom owners who have experienced this trouble and in each case the fault has been a broken track adjacent to pin 7 of IC35. Perhaps a duff batch of boards?

The moral of the story is, to check your Ram card thoroughly for address 'wrap round' and other faults before believing it is in perfect working order. (Ed. The memory test in the new RAM (B) documentation does just this, but beware, in early issues there was a typing error in the object listing around address ODODH, so check against the source code given.)



# Strings

We have recieved the following article from Dr. Michael Hendry, he passes some nice remarks about us lot at the INMC (modesty never was one of our better virtues), endorses our comments about reading the manuals, and passes a message to those in Liverpool (LSG issue 8/1 p36) that, "POKEing isn't everything; the POKE they suggest is already there as the WIDTH command in Basic." He also comments that if anyone is throwing away a NASPEN and a golfball printer, he could find them a nice friendly home. He should be so lucky !!

The Variable Protector

by M. D. Hendry

=====

Nascom owners with Basic, who want to process string data will no doubt recognise the scenario.

The intrepid programmer has just typed in 200 names and addresses, and called the alpha-sort routine, which he has checked with a few names while writing the program. To his surprise, it won't handle 200 names, but he spots the bug and corrects it at once. He calls the alpha-sort routine, and congratulates himself on how fast it now is .... but because he re-RUN (ran) the program, Basic has destroyed all the data for him !!

Simply dumping the whole Basic program and data section of the memory on to tape does not help, as the Basic program can still not be modified without re-initializing the various pointers, and there is no provision for SAVEing string variables as there is for numeric variables. It is possible to use a subroutine to convert each string into a numeric array, which can be SAVEd and LOAded by Basic, but this takes a long time, and is error prone, because of the need to stop and start the tape under program control. You see, Basic indulges in a periodic 'garbage hunt', which may take ten seconds or so, and you don't know when that will happen.

The following short program illustrates another approach, which involved a little detective work into the Basic interpreter's use of workspace, and of available RAM. The user's Basic program is stored from 10F9H upwards, and is followed by a table of variables, numeric variables first, then string variables. The strings themselves are saved at the top of memory, and the string variable table comprises pointers to these strings. Between the table and the strings is a free section of RAM. As a program is entered, the variable table is shifted towards the strings, and as strings are encountered they fill the memory towards the variable table. Should the user try to make the two overlap, an OM message will be printed, unless Basic can make more string space by disposing of unwanted strings (a garbage hunt).

This empty RAM between the table and the strings provides the answer to the problem. The program copies the table to the top of RAM, just below the strings. The Basic program can then be modified. When the modifications are finished the program copies the table back down into the new position above the end of the user's program. Care must be taken now, not to use the RUN command, but to GOTO a point in the program after all the array dimensions have been declared. The former zeroes all variables, a mistake in the latter will result in a DD message, signifying that there has been an attempt to redefine the dimensions of a variable.

Once the various workspace pointers are known, it is not hard to use them in a program, and the source listing is (I hope) self-explanatory. Note the use of RST 18H, the NAS-SYS subroutine call facility to call the NAS-SYS 'I' and 'Z' commands (intelligent copy and Basic warm start respectively).

With this program to hand, the user can simply dump the whole Basic program and its associated string files to tape using NAS-SYS, knowing that he can later make modifications to the Basic program, or even manipulate the same data with another program.

(Ed. This program has been assembled using the MACRO 80 assembler, watch out, two byte address operands appear as you would read them. Don't forget that when entering code, two byte operands are entered low byte first, ie: in reverse order to that printed below.)

VARPRO the variable protector MACRO-80 3.35 Page 1

```
title VARPRO the variable protector
subttl Version 1
; by M. D. Hendry
; Re-assembled, MACRO 80, D. R. Hunt 06.09.80

; This program is for use with Nascom ROM
; Basic and runs under NAS-SYS.

; It allows the user to save variables in RAM
; and recover them after altering a Basic
; program, which must be re-entered by (eg)
; GOTO XXX, where line XXX occurs after all
; DIMensions have been declared.

; Entry at ODOOH for save
; Entry at OD30H for restore

; No check is made for memory overflow, so do
; X=FRE(AS) to force a 'garbage hunt' and
; PRINT X to find out number of free bytes
; for additions to the Basic program. Then call
; MONITOR, E OD30, and GOTO XXX.
```

```
0000          aseq
              .z80

0018          scal equ 18h          ; Subroutine call

10D6          var equ 10d6h         ; Ptr. to var. tab.
10D8          $var equ 10d8h        ; Ptr. to strings
10DA          ramlo equ 10dah       ; Ptr. first free byte
10C3          ramhi equ 10c3h       ; Ptr. last free byte
OD80          var1 equ 0d80h        ; New loc. of var. tab.
OD82          var1 equ var1+2      ; var. tab. length
OD84          nvar1 equ var1+4     ; num. var. tab. len.
OD86          ramhi1 equ var1+6    ; Top of var. tab.

              org 0d00h

OD00          save: ld hl,($var)
OD03          ld bc,(var)
OD07          sbc hl,bc
OD09          ld (nvar1),hl
ODOC          ld hl,(ramlo)
ODOF          sbc hl,bc
OD11          ld (var1),hl
OD14          ld hl,(ramhi)
OD17          ld (ramhi1),hl
OD1A          inc hl
OD1B          ld bc,(var1)
OD1F          sbc hl,bc
OD21          ld (var1),hl
```

VARPRO the variable protector MACRO-80 3.35 Page 1-1  
Version 1

```
OD24 2A 10D6          ld hl,(var)
OD27 ED 5B OD80       ld de,(var1)
OD2B DF              rst scal
OD2C 49              defb "I"
OD2D DF              rst scal
OD2E 5A              defb "Z"
OD2F 00              nop

OD30 2A OD86         rstr: ld hl,(ramhi1)
OD33 22 10C3         ld (ramhi),hl } Restore top of RAM PSILR7
OD36 2A 10D6         ld hl,(var)
OD39 ED 4B OD84     ld bc,(nvar1)
OD3D ED 4A          adc hl,bc } Restore PTRRT
OD3F 22 10D8         ld ($var),hl
OD42 2A 10D6         ld hl,(var)
OD45 ED 4B OD82     ld bc,(var1)
OD49 ED 4A          adc hl,bc } Restore PFOLRR7
OD4B 22 10DA         ld (ramlo),hl
OD4E 2A OD80         ld hl,(var1)
OD51 ED 5B 10D6     ld de,(var)
OD55 DF              rst scal
OD56 49              defb "I"
OD57 DF              rst scal
OD58 5A              defb "Z"

end
```

VARPRO the variable protector MACRO-80 3.35 Page S  
Version 1

Macros:  
Symbols:  
\$VAR 10D8 NVARL OD84 RAMHI 10C3 RAMHI1 OD86  
RAMLO 10DA RSTR OD30 SAVE OD00 SCAL 0018  
VAR 10D6 VAR1 OD80 VARL OD82

No Fatal error(s)

```
TOD00 OD60
OD00 2A D8 10 ED 4B D6 10 ED 2A
OD08 42 22 84 OD 2A DA 10 ED 0B
OD10 42 22 82 OD 2A C3 10 22 2F
OD18 86 OD 23 ED 4B 82 OD ED 8F
OD20 42 22 80 OD 2A D6 10 ED 1B
OD28 5B 80 OD DF 49 DF 5A 00 7E
OD30 2A 86 OD 22 C3 10 2A D6 EF
OD38 10 ED 4B 84 OD ED 4A 22 77
OD40 D8 10 2A D6 10 ED 4B 82 FF
OD48 OD ED 4A 22 DA 10 2A 80 4F
OD50 OD ED 5B D6 10 DF 49 DF 9F
OD58 5A 00 00 00 00 00 00 B2
```

Ed. It occurs to us that very little work need be done to make this program completely automatic. The answer to the 'string save' problem may be found in this way. Neither would it be too difficult to turn it into a Basic module that could provide a 'string save' and 'string load' within Basic.

# Reviews

Software Review  
=====

by R.O'FARRELL

INTEGER PASCAL:Initial impressions.

From: Datron, 2 Abbeydale Road, Sheffield  
Cost 35.00 plus VAT

This compiler may well be many microcomputer users introduction to the Pascal programming language.

It comes in a number of versions which are specific to the monitor you use. My copy was designed to run with Nas-Sys 1, and in consequence there may be minor differences in the monitor dependant routines - e.g. the Nas-Sys on screen edit, which is not available under the other monitors.

The program is supplied on a tape, recorded in my case at 300 Baud CUTS, using the Block routines rather than the Tabulate format. It is designed to load at 1000H, and takes some considerable time to load, occupying memory up to approx 3200H. It contains within it a text editor to allow you to build up the program which you wish to compile. This test editor is idiosyncratic (that is, it's not like any text editors I know), and takes getting used to. It allows a program to be built up, a line displayed, or deleted. It also automatically renumbers the lines of the program when instructed, and has the ability to output a program listing to the VDU or to a printer. It can also save a program on tape, and read it back in. Each line must have a number, but the number does not necessarily put the line in its correct position in the buffer - this is done by a location pointer. Any corrections have to be made by either retyping the line, or by using the Nas Sys on screen edit.

The editor needs each action to be preceeded with a command letter, and the location of the line pointer decides where the line you have just typed/replaced/deleted is (or was). The editor works fairly well, but I feel that more work could be done on it, to bring it more towards the way a Basic interpreter builds up a program, which I think would be more familiar to most users.

Having entered the program, the Pascal compiler having been started by the following command:

E1000 BUFFERSTART BUFFERLENGTH

then you can try a compilation. This you do by using the "C" command. When entered on its own with no ancillary parameter, this command causes a trial compilation to take place. As the program compiles, it scrolls up on the screen the lines of the program. Should an error be found, the compilation is aborted, and one of 28 hex values is displayed with an error message. Reference to the manual is necessary to find out what the hex value indicates to be wrong. If the compilation is successful, then you can enter the "C" command again, this time followed by a Hex address, and the Z80 code will be generated there. You can now either leave the compiler, and jump to the code with the monitor E command, or you can use the compiler "G" command to execute the program and return to the compiler when finished.

The compiler occupies Memory 1000H - 3200H, and uses 3200H - 3750H as workspace. Memory outside this can be used for the source buffer, which is defined on entry to the compiler by additional parameters for start and length. If these parameters are omitted on first entry to the compiler, an error message is printed, and you are put back in the monitor.

As can be seen from the above figures, a 32K system would be desirable, particularly if you wish to write a long program in Pascal.

The compiler contains the run time support routines, which reside from 2D63H to 318AH, and a table of the various addresses for these routines is stored at 318BH to 3200H. It would be possible to move these routines into EPROM, having made any necessary changes to addresses, and to change the position of the address table - I'll have a go in about a months time.

The compiler supports only Pascal reserved words in capitals, which I feel to be a pity, as the program is harder to read. It has a number of extensions to "Standard" Pascal such as "CALL" to call a machine language routine at a specified address - "MEM[I]" to access the value of the byte stored at I, and alsoto store a byte at that address, and an "ELSE" addition to the " CASE" statement.

Using a % sign indicates that the number following is in HEX or is to be printed in HEX, and a " indicates that the ASCII character having that value is to be printed. These are departures from standard Pascal, and as such ought probably to be frowned upon, but I feel that they have been introduced with some justification. After all, on a micro computer, the ability to specify addresses in HEX is something we all take for granted - it's the way addresses come ! Basic please copy ! - and the MEM[I] is the equivalent of the Basic PEEK and POKE.

This is a brief description of the compiler, after about a week of fiddling with it. I would like to run the BENCHMARK programs on it to give a comparison with Basic, but that hasn't been possible due to the indisposition of my Nascom, and its subsequent hospitalisation in Nascom's repair dept.

The parentage of the compiler would appear to be the Yuen and Chung TINY PASCAL published in BYTE. It offers one considerable advantage over that implementation - SPEED! and MEMORY. The Chung and Yuen takes about 16K of memory for the first pass, plus whatever the source program takes up, and about 9k for the second pass, plus whatever the object program takes up, in addition to the basic interpreter. Because it is interpreted, it is very slow. The INTEGER PASCAL takes up about 10K (for both passes) and the source buffer, and the object code space, but is very much faster to compile.

The documentation is some 14 leaves of typescript, describing briefly the various commands, and giving syntax diagrams, a sample program and a Hex listing. It could do with rewriting and considerable extension and elaboration. It would also be a good idea if there were given a source listing for the runtime routines, and instructions on how to relocate them to allow them to be blown in EPROM. This would obviate having to load a compiled code from tape at an address - say 8000H, - and the runtime routines at 2D63H to 318H. The other problem with this compiler is that it doesn't allow you to use certain areas - for example 1000H to 3200H, or the location of the source buffer, to store the object code it creates - and it is not possible to place the object code on tape, and then reload it after you have finished with the compiler.

Conclusions: An interesting if rather expensive introduction to Pascal, which could do with attention to a few points - tidy up the editor and extend the documentation in particular.

Implemented are the following

CONST ... = ...;	PROCEDURE...(...);...;
VAR ... : INTEGER;	FUNCTION ....(...);...;
... : ARRAY [...] OF INTEGER;	MEM[...]:=...
BEGIN .... END	BEGIN ....:=...
IF...THEN...ELSE	WHILE...DO...
CASE...OF...:...ELSE...END	REPEAT...UNTIL...
FOR...:=...TO/DOWNTO...DO...	WRITE(...)
READ(...)	CALL(...)
END.	

Integer arithmetic operations are available but with no precedence ordering - strictly left to right evaluation.

DATRON INTEGER PASCAL  
=====

a software review by Richard Beal

It was with considerable excitement that I loaded the cassette from Datron containing their new 12K integer Pascal package. Pascal has received much publicity as being superior to Basic, and while I feel that both languages have their advantages it would be marvellous to see Pascal running on a Nascom. The tape loaded with no difficulty, and I cold started the program.

Editor  
=====

The editor takes quite a lot of getting used to. When adding lines they don't usually have line numbers. A command is entered to renumber all the lines before lines are listed or changed. Changing lines would be very tedious if you did not use the NAS-SYS version, since the NAS-SYS line editing can be used as in Basic. You can add, change or delete individual lines by line number. At this point I found my system sometimes "crashed", forcing me to reload Pascal and start again. Eventually the problem was traced to the fact that entering a command when the file position is not at the end of file, instead of giving an error, crashes the system completely.

Compiler  
=====

The compiler is very fast indeed, and it operates in two passes. If there are any errors in the program these are detected in the first pass. The compiler stops after the first error, but it is so fast that this is not much of a disadvantage. The error messages just give an error number and display the line where the error was found. The actual error may be on the line before. If you don't have a correct "END.", at the end, some garbage is displayed but a correct message follows it.

Remember that only integers are supported, and this limits its usefulness for some applications. Character variables did not at first appear to be supported and the documentation does not mention them, but in fact they can easily be handled by treating them as arrays of integers. READ and WRITE statements support characters.

Statements supported are:

CONST	FUNCTION	VAR	PROCEDURE	BEGIN
END	IF-THEN-ELSE	CASE-OF	WHILE-DO	FOR-TO
READ	WRITE	CALL	MEM	

CALL is an extension allowing machine code routines to be called. MEM allows the equivalent of PEEK and POKE operations.

I then typed in the demonstration SORT program provided. This included the use of recursive calls and was impressively fast. When I came to save it on tape everything appeared all right, but when I later tried to read it in, disaster! The Pascal did not allow sufficient delay at the end of the line for my tape speed of 2400 BAUD, and I could not reload the program. A speed of 1200 BAUD is, I am told, the maximum it will work with at present.

It is not possible to send program listings to your own printer routine, only directly to the UART.

Conclusions  
=====

Using this package is considerably more difficult than using Basic. With a number of fairly minor improvements and with more comprehensive documentation this would become an impressive product.

Source and Price  
=====

Datron, 2 Abbeydale Road, Sheffield S7 1FD. Tel: 0742 - 585490  
Price 35.00 + VAT

MAPP 1-4Z  
=====

a software review by Richard Beal

MAPP 1-4Z is a 4K floating point arithmetic and functions package for the Z80 processor. It is completely independent of the specific computer or operating system under which it is run. It is also written in position independent code so that it can be moved to and then run in any convenient location in memory.

The package was provided on tape in Nascom 1 format, and the demonstration program provided at the front of MAPP 1-4Z had been written for the T4 monitor, but we read it into a Nascom 2 using NAS-SYS (and a special hardware interface) without difficulty. The demonstration programs worked at first attempt, and during testing of MAPP 1-4Z, no bugs or faults of any kind were detected. This is high quality software.

MAPP is used by coding an ordinary call to the start of it, which is followed by one or more MAPP instructions. The last of these is EDH which returns control to the ordinary Z80 instruction following. In effect MAPP provides an additional set of instructions for the processor. These instructions treat the Z80 registers in combination providing two 40 bit floating point registers. The effective accuracy is 8 significant digits. While this is better than Basic, it is far short of the conventional "double precision" which has 15 or 16 significant digits. Perhaps a MAPP 2 product could offer this feature.

The real point of MAPP is that you can add its 4K of code to any other program that you develop (normally in assembler), and give yourself good arithmetic capability without the need to write and test your own routines.

It would also be easy to write a demonstration program using MAPP to provide simple calculator capabilities.

MAPP provides:

- 12 arithmetic instructions (eg: square root, divide, etc)
- 9 transcendental functions (eg: sine, log, etc)
- 12 operational instructions (eg: push/pop, relative jump, etc)
- 10 data transfer and ASCII conversion instructions (eg: output ASCII in user selected format, etc)
- 1 delay instruction

MAPP also allows for an extended form of call which includes error handling, so that errors in the MAPP instructions can be handled by your program.

The documentation is excellent and is printed and bound. Updates to MAPP are supplied through distribution to registered users by the supplier.

MAPP was written by John Turton of the University of Sussex, and is supplied by:  
Enertech Ltd., 32, Gildredge Road, Eastbourne, East Sussex BN21 4SH.

PARKINSON BASIC TOOLKIT  
=====

a software review by Richard Beal

The author of NAS-DIS, David Parkinson has now produced a "Basic Toolkit". It is used with the Nascom 8K ROM Basic to provide a number of additional direct commands which are useful to the serious programmer. It is no use to people who just RUN Basic programs, but is very useful if you write new programs or alter old ones.

Eleven additional direct commands are provided. Each consists of a full stop '.', followed by a letter, and are as follows:

- .A Auto line number. (Initial line number and increment can be specified.)
- .C Cross reference listing. (See below.)
- .D Delete a block of lines.
- .F Find a string. (See below.)
- .H Convert a Hexadecimal number to decimal.
- .K Kill all unnecessary spaces in the program to save space, and optionally delete all REM statements.
- .N )
- .X ) Similar to the same NAS-SYS commands, but used inside Basic.
- .U )
- .R Renumber. (See below.)
- . (By itself) return to NAS-SYS.

The three most interesting commands are Cross reference, Find and Renumber.

Cross reference  
=====

Number cross reference (.C), lists every line number used and shows every line in the program where that line is referred to. Then it does the same for every function used, string array, string, numeric array, and finally every ordinary variable. The LINES command is used to control how many lines appear at a time, and the WIDTH also operates if you are using a printer. This command is most useful for checking the use of variables in a program, and often helps find errors, such as a variable which is only referred to once because it was spelt incorrectly.

Find  
=====

Find (.F), displays each line which contains a string which is to be searched for. The search can be made to start at a specified line number, and the string to be searched for can contain "wild" values.

Renumber  
=====

Renumber (.R), will normally renumber the whole program and you can specify the initial line number and increment if you wish (default is from 10 in 10s). However, an excellent feature is that you can renumber just part of a program. This enables you to have each subroutine start at a round number, such as a multiple of 1000. For example, to renumber a subroutine which starts at 7800 to 8000, you might enter:

```
.R 8000 10 7800 8999
= Renumber starting at 8000, in increments of 10, the part of the program between lines 7800 and 8999. Naturally, any GOSUBs, GOTOs, etc, outside this area which refer to locations within the area are renumbered accordingly. Also, if Renumber detects any errors, such as jumps to lines which do not exist, it reports this and leaves the program unchanged.
```

To use the toolkit, you load it from tape at 1200H. Then you type E1200 and the toolkit relocates itself and cold starts Basic for you. It is still possible to warm start Basic, and to reserve memory at the top if you require. The toolkit only works with NAS-SYS and the Nascom ROM Basic.

Reviewers always try to find some criticism to show that they are unbiased, but in this case, I give up.

The Parkinson Basic Toolkit is reasonably priced at: 14.95 inc. VAT, and is available from Henry's Radio, 404, Edgware Road, London W2 1ED.

## IMPERSONAL

SCURRILOUS MUSINGS

by Guy Klueless

With the recent setting up of the West London chapel of the Mafia (migration from North London perhaps?), it's certainly put the frighteners on one West London Nascom distributor, who is in the throes of revamping his (not so efficient) mailorder service to meet this new competition. If the Mafia are true to form, there should be an interesting undeclared price war breaking out in that area shortly.

One London dealer has recently had his knuckles rapped by Nascom for (unintentionally, he says) 'ripping off' Nascom software. Because of the difficulty of protecting software in law, I wonder exactly what threat was used?

Nascoms' design team, which was scattered to the winds around the time the Receiver appeared have got together and started a new company called Specialist Micro Design. Best of luck lads !! The interesting bit is that ex-Nascom Sales Director and past INMC President Kerr Borland is on the list of directors. Now Kerr has recently formed a promotions company (the name of which I forget, that's good promotion for you); all that is now required is to throw in ex-Nascom MD John Marshall, and we've got Nascom back with the old team. Hmmm !!!

Incidental intelligence recently revealed that the chairman of the Tangerine Computer Users Group is seriously thinking about buying a Nascom 2. Go on lad, then we'll show you how to write a User's Group magazine as well.

A recent issue of Datalink credits Tony Rundle with having designed the Nascom 2. All Nascom owners with Basic know Tony, he's the one who left those bugs in the Basic, even after he was told about them. Just as well he didn't have anything to do with the rest of it. If my initials were C.S. or D.R.W., I'd be round at my lawyers right now.

I hear that the INMC have had a bit of stick after the last issue. One disgruntled member sent a curt note to the effect that if the INMC is as hard up as they say they are, why waste space by publishing rubbish like that spoof readers' letter by the chairman, David Hunt (spelt backwards) and silly things like Lawrence. Well a spokesman for the INMC committee made this point:

They had hoped for 5 pages of ads which did not appear, so as the printers must have things in blocks of 4 pages, 3 pages were left unused.

Those items were included to fill the extra space, and a bit of light relief isn't a bad idea, most members appreciate it.

He also muttered under his breath that he'd looked in the writer's envelope about ten times and still didn't find any article for consideration for the magazine. On that score, if they decide to stop printing rubbish, they might decide to stop printing my bit, and that would never do.

# Z80 Books

BOOK REVIEW  
=====

By R.O'FARRELL

Practical Microcomputer Programming - The Z80  
-----

By W.J. Weller.  
Published Northern Technology Books, Evanston, USA.

Cost: Approx 20.00

I imagine that this book title has caught most microfans eyes as they scan advertisements of books on programming because of its high price. What do you get for your 20.00 ?

This is a thick hardcovered book of some 480 pages, well laid out and printed, on the Z80. It contains a complete listing for an interesting Assembler and monitor, and by sending off the back page to the publishers, one can receive a free paper tape of the assembler monitor object code. Great !, I hear hundreds (thousands ? (millions ?)) of readers say as they reach for the telephone to order this from their Friendly Local All Night Computer Store, or their F.L.A.N.C.S's answering machine for 24 hours ordering. Don't do it ! Wait ! There is a snag.

The snag is that this excellent book, for reasons that Mr. Weller explains in his introduction, does not use Zilog/Mostek mnemonics. Mr. Weller feels that in the first place most microists are already acquainted with the 8080, and he extends that language to cover the added Z80 instructions. Secondly, he feels that the Z80 language does not make a good practical programming language, and thirdly, that it requires about one third as many keystrokes again per line as conventional languages. These arguments may be true in the USA, where many people have graduated to the Z80 from 8080s, but I doubt if they are in general true in Europe, and in particular not of Nascomers, most of whom have assembled their own machine, with no previous experience of a microcomputer. This is the first and major snag with this book. The language used, extended 8080, is very incomprehensible after the clear syntax of the Z80 language. To change from one to the other can be done, and even done on the fly, but Murphy's law always prevails - there are always hidden bugs that take hours to find. One example will show this up - a valid Z80 statement would be JP BEGIN; meaning JUMP to the label BEGIN. This is also a valid statement in 8080 extended, meaning JUMP IF POSITIVE to BEGIN. This one has caught me on a number of occasions, with another experienced Nascomer looking over my shoulder as I translated. O.K. I hear you say, I can put up with that to get an assembler listing, and then I'll change the assembler to the type of mnemonic I like.

The assembler is very interesting (if you can see through the 8080 mnemonics), as it allows a number of pseudo ops which are not normally encountered - such as to define a store area of a certain size, and initialise it to a given character, and one called TITL which allows you to title a program and have its name on each page just by using the Pseudo-op. It also allows a listing to be started and stopped at defined points within a program - which is most useful and I have implemented that on ZEN for myself, and it leaves space for any extensions to the pseudo ops you may wish to make. The assembler is written entirely in 8080 code, and flags any instructions which are unique to the Z80 - so it is in fact a Z80 cross assmbler to run on an 8080 (if you are the unfortunate) and will also facilitate those who have occasion to convert programs back to 8080 code.

It is claimed that this assembler runs at 1000 lines per minute on a 2MHz system, which is quite fast (comparative figures being, for ZEAP on my N1 about 400 lines per minute, and ZEN about 1150 lines per minute). Unfortunately it takes up 9K of memory, compared with ZEAP's 2.8K and ZEN's 3.5K.

In the body of the book, Mr. Weller writes with great sense, and with some humour on his subject. Unfortunately, I find it difficult to read because of the necessity to translate from one mnemonic system to another, but he has some good ideas, and is definitely worth reading if you can manage to do so without the expense of buying the book. His monitor listing is interesting, as it (and the assembler) are fully commented, but it does not offer any sizeable advantages over the T2 bug (remember that one ?).

In my view, a book to be inspected before you purchase - and don't fall for the bait of the free assembler - for the 20.00, you could buy a copy of ZEN from Newbear, and have enough left over to buy at least one good book on the Z80.

Z80 and 8080 Assembly Language Programming  
-----

By Kathe Spracklen  
Published by Hayden Book Co  
\$7.95.

The name Spracklen is best known as that of the authors of Sargon the famous chess program for Z80 computers. This paperback book is written by Kathe Spracklen, and sets out to be an introduction to programming the Z80 in assembly language for absolute beginners. It is clearly written, showing the logical development of a program, with the emphasis on keeping proper documentation and notes. It would serve as an excellent introduction to machine language. Not only does it use the Z80 Zilog mnemonics, but it also illustrates every point in 8080 extended (TDL dialect). Those who wish to implement Sargon on their Nascom will find this most useful for assistance in translating the mnemonics from the TDL dialect in which Sargon is written to the Z80 mnemonics most of use are used to.

It has a number of exercises, and is fully provided with answers, so that you can check your work if you wish.

(Continued from page 59)

1EE0 D8 00 47 FF E5 08 1F 00	F8	1F40 20 20 20 20 20 20 20	5F	1FA0 18 20 18 20 18 20 18	9F
1EE8 45 FF 23 0A 00 00 FD FE	7A	1F48 20 20 20 20 20 20 20	67	1FAB 20 20 20 20 20 20 20	C7
1EF0 90 08 15 00 FF 00 FF 00	89	1F50 20 20 18 20 18 20 18	57	1FB0 20 20 20 20 20 20 20	CF
1EF8 FF 20 20 20 20 20 20	F5	1F58 18 20 18 20 18 20 18	57	1FB8 20 20 20 20 20 20 20	D7
1F00 20 20 20 20 20 20 20	1F	1F60 18 20 18 20 18 20 18	5F	1FC0 20 20 20 20 20 20 20	DF
1F08 20 20 20 20 20 20 20	27	1F68 20 20 20 20 20 20 20	87	1FC8 20 20 20 20 20 20 20	E7
1F10 20 20 07 20 07 20 07	E4	1F70 20 20 20 20 20 20 20	8F	1FD0 20 20 20 20 20 0F 20	DE
1F18 07 20 07 20 07 20 07	D3	1F78 20 20 20 20 20 20 20	97	1FD8 20 20 0F 20 0F 20 0F	C4
1F20 07 20 07 20 07 20 07	DB	1F80 20 20 20 20 20 20 20	9F	1FE0 0F 20 0F 20 0F 20 0F	EB
1F28 20 20 20 20 20 20 20	47	1F88 20 20 20 20 20 20 20	A7	1FEB 20 20 20 20 20 20 20	07
1F30 20 20 20 20 20 20 20	4F	1F90 20 20 18 20 20 20 18	9F	1FF0 20 20 20 20 20 20 20	0F
1F38 20 20 20 20 20 20 20	57	1F98 20 20 18 20 18 20 18	9F	1FF8 20 20 20 20 20 20 20	1F



# BASIC

A Comparative Review of Two Toolkits for Nascom. by James Weatherson-Roberts

Some time ago I saw an advert in one of the mags for a BASIC toolkit for my Nascom 2 from Bits and P.C.s, so after a lengthy 'phone conversation I ordered it by Barclaycard. The next day a friendly but apologetic gentleman rang from Yorkshire to tell me that Barclaycard said I couldn't have a Toolkit, but thank you for the thought anyway. The day after that a most unfriendly gentleman rang from Leeds to tell me that Barclaycard would be very pleased never to hear from me again.

A few days later, I happened to walk into my friendly local Nascom Distributor with (yet again) some problem or other.

"Oh, it's you, what is it this time Young Man ? Oh look, you probably need this (mutter,mutter)"  
"What is it ?"  
"A Bloo...Basic Toolkit, of course (mutter,mutter)"  
"Oh"

Under this enormous pressure, I found that I had bought a Basic toolkit with my Barclaycard (and still had my problem). When I got home, there was a neat little Jiffy bag from Yorkshire on the doormat, since Barclaycard had paid up after all.

So, I now have TWO Basic toolkits. Is this so bad ? Well, really it's quite an advantage, as the facilities offered by the two are sometimes quite different. So I here present a comparative review !!

## FACILITIES OFFERED (N/A : not available)

Henry's Toolkit Bits and P.C.'s

N/A APND <filename>  
Appends file <filename> to the end of existing file. This is a most useful facility since you can then keep a tape of subroutines and APND them onto the end of your existing BASIC file.

.A<start,increment> AUTO<start,increment>  
Auto line numbering. Both work; I prefer the Bits and P.C.s version as you can change the sequence by overtyping the existing number before entering the line.

.C N/A  
Brilliant! This produces a cross-reference of all line numbers, functions and variables (array or otherwise) Find out why your program crashes in an hour instead of a week! (...er, perhaps I should have used two variable names in that 65K Desertrrek...)

.D<start,end> DELE<start,end>  
Obviously "delete". Both work.

N/A DUMP  
Again a very useful debugging facility. This command dumps all simple variables (including strings but unfortunately not array variables) with their values at the end of the run. Must be used immediately after run ends (i.e. no editing) for accuracy.

.F<lineno.><string> FIND<string>  
Both very nice and a bit different. .F etc. starts at <lineno.> if specified and searches for <string> to the end of the text file. If you type for <string> "L?" for example, any character will match the "?"; useful if you use related variable names. However, it's a bit of a pain for Reserved words, as if you want to find say "REM" you have to use the graphics character equivalent. The Bits and P.C.s version has fixed this and is easier to use. But perhaps someone can explain why I can't have a "find and optionally change"? Both of these finds can cause confusion if you forget to turn them off.

N/A HELP  
Very nice idea but not as good as I'd like. ("Tough", I hear). After for example , a "? SN Error" , if you type "HELP" the line that caused the problems will be displayed at the bottom of your screen, with the cursor hopefully near the problem statement. Unfortunately, the Basic tends to cop out at different places depending on the type of error, so sometimes it's not a "HELP". However, if you use multi-statement lines (a very bad habit) this can indeed be a "HELP"

.H<hex value> HEX<up to 10 values>  
These commands accept hex and return decimal. Apparently they are useful for machine code, whatever that is - I really wouldn't know. Presumably the bigger the better, so the Bits and P.C.s gets the "\*" (scared of machine code ? Of course not..I'm petrified)

.K<S> N/A  
Again, I find this useful. .KS deletes all non-significant spaces; .K deletes the "REM"s as well, so making your programs even more incomprehensible ( and much faster).

.R<nu.start><inc><start><stop> RENU<inc>  
Renummer,of course: the one really essential facility offered by these add-ons. The Henry's version is far more versatile, allowing you to start subroutines on reasonable increments and with care, changing the order of your file!

N/A STEP  
This is an example of something that is lovely when it works. After typeing "STEP" you are asked for a string. You can enter "The value of"<list of variables> is"<list of variables>. When you type "RUN" the Basic program is single stepped with the values defined printed on the top line of the display. Great idea. Unless you have something else happening on the top line. In which case you can get some extraordinary Syntax errors. However, if you want to analyse a non- printing subroutine it can be very useful. But remember to turn it off.

## UTILITIES

N/A Inkey  
Not a command, but a routine that is rather trying to call. I prefer the routine published in INMC as this also gives the flashing cursor.

Keyboard Repeat  
Both toolkits have this facility. The Henry's one tells you how to vary rates.

N/A Printer handshake  
A useful extra, saves a wasted 2708 or lots of typing. Works. From Basic, one Poke on, one Poke off.

N/A RINK  
Supposed to scan the keyboard once and return with value if key pressed. I can't get this to work ( but see above re. machine code !).

.U,.X,.N N/A  
Activate U out, X out, Return to monitor respectively. Since I don't understand Nas-Sys the first two are an unknown quantity for me.

Irritating Features.

The Henry's format of "<Command>" is a nuisance compared to the meaningful names of the other. Soon get used to it though. The Bits and P.C.'s has some strange features (not really bugs but with no info....)

DOCUMENTATION of both is of the standard that we have come to expect, i.e. lousy. There seems small chance of gaining any comprehension of how either works, though the Henry's information is slightly better.

RELATIVE ADVANTAGES ...I generally use the Bits and P.C.s toolkit since it is 1) in ROM and instantly available 2) uses no RAM space at all except in the work area. The Henry's version eats 2K of expensive RAM. However, the price paid for this is that the Bits and P.C.s toolkit is very noticeably slower. Fair exchange ?

ARE THEY WORTH BUYING ? Well, I don't regret the money and wouldn't want to lose either. However I would trade both for the following facilities comprehensively implemented in ROM: 1) Renumber, like the Henry's 2) Find and optionally change 3) Append 4) Auto 5) Cross reference listing 6) Dump ...in that order, implemented using Keyboard repeat ( or use Nas-sys 3) The rest of the facilities are o.k. but Oh! for a find and optionally change.....

Please note: the author is a known Dodo; bear this in mind when buying either.

Below is a 'PRINT USING' routine, also from Mr. James Weatherson-Roberts.

```
780 REM ** DEMO ROUTINE **      790 PC=.1
800 GOSUB1150                  810 PRINT,PU$
820 PC=-1                      830 GOSUB1150
840 PRINT,PU$                  850 OP=1:PC=10
860 GOSUB1150                  870 PRINT,PU$
880 OP=1:PC=-100              890 GOSUB1150
900 PRINT,PU$                  910 PC=1234.0345
920 GOSUB1150                  930 PRINT,PU$
940 PC=1234.0354              950 GOSUB1150
960 PRINT,PU$                  970 PC=9999999
980 GOSUB1150                  990 END

1000 REM *****'PRINT USING'*****
1010 REM Version 3.2
1020 REM Variables used:
1030 REM PM$ (Print mask)
1040 REM SW (Switch)
1050 REM OP (Option)
```

```
1060 REM PC$ (Fudge variable)
1070 REM Call with value in PC (Print Call)
1080 REM Returns PU$ (Print using)
1090 REM -----
1100 REM !                               !
1110 REM !       Call Routine at 9013   !
1120 REM !                               !
1130 REM -----
1140 REM -----Test Limits-----
1150 IF PC<=-9999.99 THEN IF PC>=-9999.99 THEN12
10
1160 PRINT"Value "PC"is outside accuracy limit"
1170 PRINT"Please call programmer"
1180 PRINT
1190 PRINT"***** RUN ABORTED *****":END
1200 REM -----Reset Sign Switch-----
1210 SW=0
1220 REM -----Round and Extract Sign-----
1230 IF PC<0 THEN PC=ABS(PC):SW=1
1240 PC=INT(PC*100+.5)/100
1250 REM -----Make it a string-----
1260 PU$=STR$(PC)
1270 REM -----Strip off leading space-----
1280 PU$=RIGHT$(PU$,LEN(PU$)-1)
1290 REM -----Test for no decimal places-----
1300 IF PC=INT(PC) THEN PU$=PU$+".00":GOTO1360
1310 REM -----Test for one decimal place-----
1320 REM ----( Fudge for Microsoft Bug )-----
1330 PC=PC*10:PC$=STR$(PC):PC=VAL(PC$):PC$=""
1340 IF PC=INT(PC) THEN PU$=PU$+"0"
1350 REM ---Check there is leading zero-----
1360 IF LEN(PU$)<>3 THEN1390
1370 PU$="0"+RIGHT$(PU$,3)
1380 REM -----Change all '0'to'0'-----
1390 FOR I=1 TO LEN(PU$)
1400 PC$=MID$(PU$,I,1)
1410 IF PC$="0" THEN PC$="0"
1420 PU$=PU$+PC$
1430 NEXT I
1440 PU$=RIGHT$(PU$,I-1)
1450 PC$=""
1460 REM ---Message(Debit):Change to suit-----
1470 IF SW=1 THEN PU$=PU$+" p DB":GOTO1520
1480 PU$=PU$+" p   ":REM Five spaces
1490 REM ---Produce Output String-----
1500 REM ---OP Selects Floating#/Fixed#-----
1510 REM -----(1)Fixed #-----
1520 IF OP>0 THEN1570
1530 PM$="####"
1540 PU$=LEFT$(PM$, (14-LEN(PU$)))+PU$
1550 GOTO1600
1560 REM -----(2)Floating #-----
1570 PM$="    ":REM Four spaces
1580 PU$="#" +PU$
1590 PU$=LEFT$(PM$, (14-LEN(PU$)))+PU$
1600 OP=0:PM$="":PC=0
1610 RETURN
1620 REM*****END 'PRINT USING'*****
Ok
```

# A third toolkit

WATKINS TOOLKIT REVIEW

=====

On the closing date for copy for this issue a kind member said to me, "I've got the Watkins Basic Toolkit". I was tempted to reply that I hoped it wasn't contagious, or perhaps he was boasting about some obscure operation I hadn't heard of, but being my normal polite self and deciding to play it safe, I replied, "Never heard of it.". Anyway, he volunteered to give me a review of it, and as this issue contains reviews of the other two toolkits we know about, I asked him if I could have it yesterday, as this issue was already 'put to bed'. Well, he was a bit slow for yesterday, but the day after (or do I mean two days after yesterday), it was duly in my hands. I must admit when I read it, it was a little unclear, but as he had kindly provided me with (an almost unreadable) copy of the instructions, I managed to unscramble it. This article has had to be edited quite a bit, and as I don't know the animal at all, some of it is summize. My apologies to Mr. Watkins and Mr. Mathison if I've misinterpreted anything. - DRH.

The Watkins Basic Toolkit a software review by A. D. Mathison  
=====

## What you get

A cassette tape written in either N1 or N2 format containing on one side some demonstration and games programs, and on the other, a couple of copies of the toolkit. Also a couple of pages of badly copied, and at first reading, slightly misleading instructions.

## Operation

It appears that the tool kit is a series of machine code subroutines which may be called from Basic. The call being made to a small routine which takes the parameter passed to it from the USR call, and uses that parameter to determine which routine is to be called. There are eight routines in all. I will confine myself to the N2 version, but I believe that the differences between this and the N1 version are minimal. (The instructions make no mention of NAS-SYS/NASBUG compatibility, but it certainly works on NAS-SYS 1, no tests have been carried out on NASBUG or NAS-SYS 3.) After power up, initialize Basic as normal using the 'J' command and unless memory size is to be restricted for your own reasons, type 'ENTER' in reply to the 'Memory size' prompt. Get back into the monitor by using 'RESET' or 'MONITOR' and load the toolkit using the 'R' command. It occupies space from 0C80H to 0FE3H. At the end of the read, re-enter Basic using the 'Z' command. The functions are now initialized by a 'direct' DOKE 4100,3200 command or as statements within a Basic program.

Functions 0, 1, 2, 3 and 6 are best called with a ?USR(X) where X equals the function number. Functions 4 and 5 return a value into a variable X=USR(4) or X=USR(5) 7 and 8 are special and are entered as U=USR(7)A\$ or U=USR(8)A\$ where A\$ is the array to be saved or loaded. Functions 0 and 3 can be given maximum space by a CLEAR 0 but don't forget to reset the string space otherwise there will be no string space available when you come to run the program.

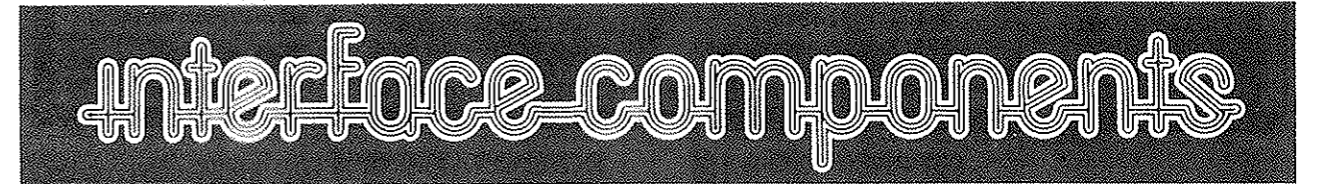
## Functions

- 0 EXPAND. Inserts spaces into a program to make it more readable. It does not touch spaces within REMs or inside quotes.
- 1 COMPRESS. The opposite of the above.
- 2 REMDEL. Same as 1, but also removes all REM lines.
- 3 RENUMBER. Renumber lines starting at 10 in increments of 10. The starting number can be changed by DOKE 3709,X and the increment changed by DOKE 3720,Y. Undefined line references are renumbered /999/ and may be located using the find command. Any renumbering errors are liable to corrupt the program.
- 4 INKEY2. The keyboard status is returned as the variable assigned to the USR call. This has a repeat action as long as the key is depressed.
- 5 INKEY1. As above but with no repeat action. Both 4 and 5 return 0 if no key is depressed.
- 6 FIND. On calling you are prompted with F? You may enter the functions you wish to find, either alphabetic or numeric or a combination of both, terminated by an 'ENTER'. The lines containing the data required is listed on the screen N lines at a time, where N has been set by the LINES command. Default is 5.
- 7 SAVE\$. Save a string array.
- 8 LOAD\$. Reloads a string array.

## Conclusion

The documentation is poor. Operation and use are good. Shame there is no LINK function instead of INKEY2. Overall very good value for money.

The tape is available from A. Watkins, 12, Merton Close, Maidenhead, Berks. Price 6.00.

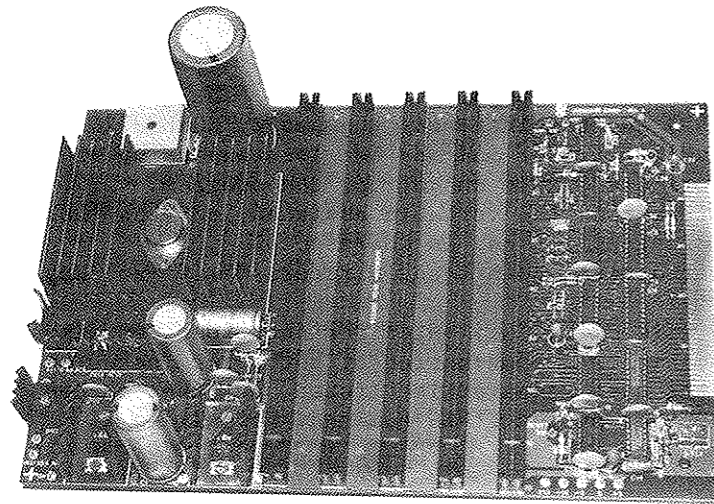


Interface Components stocks a full range of Nascom products and peripherals suitable for use with Nascom 1 or 2. In addition to this wide range of product we have the following items at highly competitive prices.

- NAS-SYS 3 in 2x2708 (N1) or 1x2716 (N2) - 25.00 + VAT (p&p 35p)
- MK36271 8K BASIC ROM - 20.00 + VAT (p&p 35p)
- IMP RIBBON CARTRIDGES - 6.60 + VAT (p&p 50p)
- FAN FOLD PAPER (2000 SHEETS) - 18.00 + VAT (p&p 2.50)
- NASCOM TESTER FOR NASCOM 1 and 2 - 55.00 + VAT (p&p 1.50)  
This board connects to the N1 pcb or the NASBUS and monitors all bus lines.
- MACHINE CODE PROGRAMMING FOR THE NASCOM 1 AND 2 - 4.50 (No VAT. 50p p&p)  
This book takes the reader step by step through most of the impressive Z80 instruction set, with particular reference to the Nascom.
- THE CP/M HANDBOOK by RODNEY ZAKS - 8.95 (No VAT. 75p p&p)  
An introduction to the use of CP/M equipped computers, and a reference text. Recommended for those adding CP/M to their Nascom.
- PROGRAMMING THE Z80 by RODNEY ZAKS - 9.75 (No VAT. 1.00 p&p)  
A thorough introduction to Z80 machine language programming, from basic concepts to advanced data structures and techniques.

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

'SUPERMUM' COMBINED  
MOTHER/BUFFER/PSU BOARD



**£85**  
+ £2.50 P&P + VAT



Gemini Microcomputers

Oakfield Corner Sycamore Road Amersham Bucks HP6 5EQ  
Telephone (02403) 22307 Telex 837788

The Gemini G806 'Supermum' makes it possible to exploit the full capabilities of the ubiquitous Nascom 1. This 12"x8" board sits parallel to the Nascom and provides a superb 5A PSU, a 5 slot motherboard, and a buffer board that includes 'Reset Jump' facilities and also overcomes problems experienced with the NM buffer board. At only 85.00 this product represents superb value for money. To existing and potential Nascom 1 owners we recommend most strongly that you consider the very easy expansion potential of this product.

All prices are correct at time of going to press.

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

<b>BITS &amp; PC'S</b> 4 Westgate, Wetherby, W. Yorks. Tel: (0937) 63774.	<b>ELECTROVALUE LTD.</b> 680 Burnage Lane, Burnage, Manchester M19 1NA. Tel: (061) 432 4945.	<b>TARGET ELECTRONICS</b> 16 Cherry Lane, Bristol BS1 3NG. Tel: (0272) 421196	<b>HENRY'S RADIO</b> 404 Edgware Road, London W2. Tel: (01) 402 6822.
<b>BUSINESS &amp; LEISURE MICROCOMPUTERS</b> 16 The Square, Kenilworth, Warks. Tel: (0926) 512127.	<b>28 St Judes, Englefield Green, Egham, Surrey TW20 0HB. Tel: (0784) 33603. Tlx: 264475.</b>	<b>INTERFACE COMPONENTS LTD.</b> Oakfield Corner, Sycamore Road, Amersham, Bucks. Tel: (02403) 22307. Tlx: 837788.	<b>Tlx: 262284 (quote ref: 1400)</b>



JOIN US  
HERE

INMC80 SUBSCRIPTIONS

I would like to subscribe to the INMC80 News for 12 months, starting from the INMC80-3 issue. I enclose my cheque / postal order / international money order (but NOT a French cheque) payable to "INMC80", value :

UK Subscriptions : 6.00  
Overseas subscriptions : 7.50

Name :

Address :

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

# A GREAT DEAL FROM 6 NASCOM DEALERS

and guaranteed after-sales service

## BUILT FLOPPY DISC SYSTEM FOR NASCOM 1/2 FROM £395 + VAT

It's here at last. A floppy disc system and CP/M. The disc unit comes fully assembled complete with one or two 5 1/4" drives (FD250 double sided, single density) giving 160K per drive, controller card, power supply, interconnects from Nascom 1 or 2 to the FDC card and a second interconnect from the FDC card to two drives, CP/M 1.4 on diskette plus manual, a BIOS EPROM and new N2MD PROM. All in a stylish enclosure.

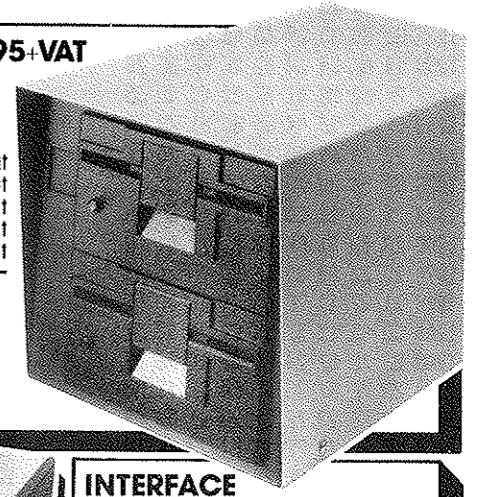
**CP/M SYSTEM.**  
Nascom 2 Single drive system . £450 + Vat  
Nascom 2 Double drive system £640 + Vat  
Nascom 1 Single drive system . £460 + Vat  
Nascom 1 Double drive system £650 + Vat  
Additional FD250 drives ..... £205 + Vat

### D-DOS SYSTEM

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

Single drive system ..... £395 + VAT  
(please state which Nascom the unit is for)

Certain parts of the CP/M and D-DOS disc systems are available in kit form. Details available on request.



## ENCLOSURE FOR N2 + 5

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

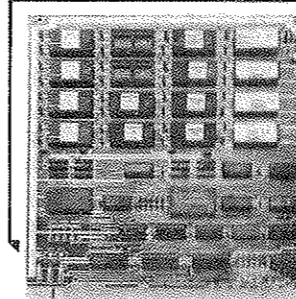
Kenilworth case £49.50 + Vat  
2-card support kit £7.50 + Vat 5-card support kit £19.50 + Vat



## INTERFACE ENHANCING UNIT

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

Castle Interface Unit .. **£17.50 + Vat**



## EPROM EXPANSION

The Nasbus compatible EPROM board accepts up to 32, 2716 or 16, 2708 EPROMs. It has a separate socket for the MK36271 BK BASIC ROM for the benefit of Nascom-1 users. And for Nascom-2 users, a wait state for slower EPROMs. The board also supports the Nascom Page Mode Scheme.

EPROM Board (kit) ..... **£55 + Vat**  
EPROM Board (built & tested) **£70 + Vat**

## A-D CONVERTER

For really interesting and useful interactions with the 'outside world' the Miliham analogue to digital converter is a must. This 8-bit converter is multiplexed between four channels - all software selectable. Sampling rate is 4KHz. Sensitivity is adjustable.

Typical applications include temperature measurement, voice analysis, joystick tracking and voltage measurement. It is supplied built and tested with extensive software and easy connection to the Nascom PIO.

Miliham A-D Converter (built and tested) **£49.50 + Vat**

## PROGRAMMER'S AID.

For Nascom ROM BASIC running under Nas-Sys. Supplied in 2x2708 EPROMs. Features include: auto line numbering; intelligent renumbering; program appending; line deletion; hexadecimal conversion; recompression of reserved words; auto repeat; and printer handshake routines. **Price £28 + Vat.**

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

## BASIC PROGRAMMER'S AID.

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

## PROM-PROG MKII.

2708 (multi-rol) and 2716 (single-rol) EPROM programmer kit controlled by N1/2 PIO. Supplied with comprehensive software for use with Nas-Sys. **Price: £25.95 + Vat.**

All prices are correct at time of going to press.

All the products are available while stocks last from the Nascom dealers below.

(Mail order enquirers should telephone for delivery dates and post and packing costs.) Access & Barclaycard welcome.

<b>BITS &amp; PC'S</b> 4 Westgate, Wetherby, W. Yorks. Tel: (0937) 63774.	<b>ELECTROVALUE LTD.</b> 680 Burnage Lane, Burnage, Manchester M19 1NA. Tel: (061) 432 4945.	<b>TARGET ELECTRONICS</b> 16 Cherry Lane, Bristol BS1 3NG. Tel: (0272) 421196	<b>HENRY'S RADIO</b> 404 Edgware Road, London W2. Tel: (01) 402 6822.
<b>BUSINESS &amp; LEISURE MICROCOMPUTERS</b> 16 The Square, Kenilworth, Warks. Tel: (0926) 512127.	<b>28 St Judes, Englefield Green, Egham, Surrey TW20 0HB. Tel: (0784) 33603. Tlx: 264475.</b>	<b>INTERFACE COMPONENTS LTD.</b> Oakfield Corner, Sycamore Road, Amersham, Bucks. Tel: (02403) 22307. Tlx: 837788.	<b>Tlx: 262284 (quote ref: 1400)</b>



SIMPLIFIED MODULAR PROSE  
NAS-SYS, EXECUTE AT 1000

1000 31 00 10 21 0F 0D 36 0D 01  
1008 EF 0C 00 21 55 12 11 CB 77  
1010 0B 01 2D 00 ED B0 31 00 27  
1018 10 CD 30 12 21 00 0D 3E B3  
1020 05 CD 45 12 C6 02 32 01 54  
1028 0D EF 0D 48 6F 77 20 6D FC  
1030 61 6E 79 20 63 68 61 72 46  
1038 61 63 74 65 72 73 20 70 5A  
1040 65 72 20 6C 69 6E 65 20 0F  
1048 3F 20 28 34 37 20 2D 0E B7  
1050 32 35 35 29 0D 00 CD FF FE  
1058 10 AF B4 28 02 2E FF 3E 70  
1060 2F BD 38 01 6F 22 02 0D 35  
1068 EF 0D 0D 44 65 6C 61 79 70  
1070 20 74 69 6D 65 20 3F 20 CE  
1078 28 30 20 2D 20 32 35 35 E9  
1080 29 0D 00 CD FF 10 AF B4 05  
1088 28 03 21 FF 00 B5 20 01 B9  
1090 2C 22 04 0D EF 0D 0D 00 08  
1098 21 82 12 7E E5 21 00 0D EE  
10A0 CD 45 12 57 15 E1 28 08 51  
10A8 23 7E B7 20 FB 15 18 F6 4E  
10B0 23 7E B7 28 05 CD 39 11 5C  
10B8 18 F6 23 7E B7 28 07 23 80  
10C0 7E B7 20 FB 18 F4 23 7E CD  
10C8 B7 20 D0 3E 20 CD 39 11 F4  
10D0 3A 01 0D 3D 32 01 0D 20 C5  
10D8 BF 2A 08 0D 36 0D CD F1 E7  
10E0 11 DF 6A CD 30 12 21 00 7A  
10E8 0D 3E 05 CD 45 12 C6 02 34  
10F0 32 01 0D 18 A3 2A 04 0D 36  
10F8 FF 2B 7C B5 20 FA C9 DF 25  
1100 7B FE 0D 28 0F FE 08 28 FC  
1108 08 FE 30 38 F2 FE 3A 30 E1  
1110 EE F7 18 EB DF 7C CD 32 63  
1118 11 38 E4 11 00 00 42 4F F8  
1120 E5 62 6E 29 29 19 29 09 80  
1128 EB E1 23 CD 32 11 30 EF 57  
1130 EB C9 7E FE 30 D8 E6 0F 6E  
1138 C9 E5 2A 08 0D 77 23 22 F2  
1140 08 0D ED 5B 06 0D 2A 02 ED  
1148 0D 13 B7 ED 52 19 28 06 B6  
1150 ED 53 06 0D E1 C9 E1 ED 2C  
1158 5B 08 0D 1B EB 06 00 7E 63  
1160 FE 20 28 05 2B 1B 04 18 1E  
1168 F6 D5 36 0D 22 0B 0D 78 39  
1170 B0 CA F0 11 18 01 C1 3A 10  
1178 0A 0D B7 28 08 2A 0B 0D C9  
1180 22 0D 0D 18 06 21 10 0D 29  
1188 22 0D 0D C5 3A 0A 0D B7 A2  
1190 28 1B 2A 0D 0D 2B 7E FE CF  
1198 20 28 06 FE 0D 28 D7 18 19  
11A0 F4 2B BE 28 FC 3E 0D BE BB  
11A8 28 CC 23 18 14 2A 0D 0D 40  
11B0 23 7E FE 20 28 06 FE 0D B9  
11B8 28 BC 18 F4 23 BE 28 FC BE  
11C0 2B EB 2A 0B 0D B7 ED 52 1F  
11C8 E5 C1 2A 0B 0D E5 D1 13 8A  
11D0 ED BB 23 36 20 3A 0A 0D 5A  
11D8 B7 28 01 2B 22 0B 0D 2A 50  
11E0 0B 0D 23 22 0B 0D C1 10 37  
11E8 A2 3A 0A 0D 2F 32 0A 0D 64

11F0 D1 21 10 0D E5 CD F5 10 C7  
11F8 E1 7E F7 F5 DF 62 30 18 DD  
1200 FE 1B 20 14 CF FE 0D 20 59  
1208 05 DF 6A C3 16 10 FE 1B 6A  
1210 20 06 DF 6A DF 6A DF 5B 14  
1218 F1 FE 0D 23 20 D6 CD 23 2F  
1220 12 EB C9 21 10 0D 22 08 60  
1228 0D 21 FF FF 22 06 0D C9 64  
1230 21 10 0D 06 08 3E 20 77 63  
1238 23 10 FC 22 08 0D 21 07 DB  
1240 00 22 06 0D C9 C5 47 ED 49  
1248 5F 86 38 01 3D 77 90 30 EC  
1250 FD 80 3C C1 C9 2A 2A 2A 23  
1258 20 53 49 4D 50 4C 49 46 9E  
1260 49 45 44 20 49 4E 54 45 94  
1268 47 52 41 54 45 44 20 4D 9E  
1270 4F 44 55 4C 41 52 20 50 B9  
1278 52 4F 53 45 20 32 20 2A 5F  
1280 2A 2A 0A 49 6E 20 78 61 98  
1288 72 74 69 63 75 6C 61 72 00  
1290 2C 20 00 4F 6E 20 74 68 A7  
1298 65 20 6F 74 68 65 72 20 71  
12A0 68 61 6E 64 2C 20 00 4B E1  
12A8 6F 77 65 76 65 72 2C 20 9E  
12B0 00 53 69 6D 69 6C 61 72 93  
12B8 6C 79 2C 20 00 41 73 2D CF  
12C0 61 20 72 65 73 75 6C 74 F2  
12C8 61 6E 74 20 69 6D 70 6C EF  
12D0 69 63 61 74 69 6F 6E 2C F5  
12D8 20 00 49 6E 20 74 68 69 26  
12E0 73 20 72 65 67 61 72 64 FA  
12E8 2C 20 00 42 61 73 65 64 25  
12F0 20 6F 6E 20 69 6E 74 65 CF  
12F8 67 72 61 6C 20 73 75 62 1A  
1300 73 79 73 74 65 6D 20 63 3B  
1308 6F 6E 73 69 64 65 72 61 70  
1310 74 69 6F 6E 73 2C 20 00 9C  
1318 46 6F 72 20 65 78 61 6D 1D  
1320 70 6C 65 2C 20 00 54 68 7C  
1328 75 73 2C 20 00 49 6E 20 46  
1330 72 65 73 70 65 63 74 20 59  
1338 74 6F 20 73 70 65 63 69 62  
1340 66 69 63 20 67 6F 61 6C 4B  
1348 73 2C 20 00 00 0A 61 20 A5  
1350 6C 61 72 67 65 20 70 72 70  
1358 6F 70 6F 72 74 69 6F 6E E5  
1360 20 6F 66 20 74 68 65 20 E9  
1368 63 6F 6F 72 64 69 6E 61 CA  
1370 74 69 6F 6E 20 63 6F 6D 9C  
1378 6D 75 6E 69 63 61 74 69 E5  
1380 6F 6E 20 00 61 20 63 6F E3  
1388 6E 73 74 61 6E 74 20 66 B9  
1390 6C 6F 77 20 6F 66 20 65 6F  
1398 66 66 65 63 74 69 76 65 F7  
13A0 20 69 6E 66 6F 72 6D 61 BF  
13A8 74 69 6F 6E 20 00 74 68 71  
13B0 65 20 63 68 61 72 61 63 AA  
13B8 74 65 72 69 7A 61 74 69 37  
13C0 6F 6E 20 6F 66 20 73 70 A8  
13C8 65 63 69 66 69 63 20 63 C1  
13D0 72 69 74 65 72 69 61 20 F3  
13D8 00 69 6E 69 74 69 61 74 DD  
13E0 69 6F 6E 20 6F 66 20 63 B1  
13E8 72 69 74 69 63 61 6C 20 03  
13F0 73 75 62 73 79 73 74 65 B5  
13F8 6D 20 64 65 76 65 6C 6F 17

1400 70 6D 65 6E 74 20 00 74 CC  
1408 68 65 20 66 75 6C 6C 79 35  
1410 20 69 6E 74 65 67 72 61 2E  
1418 74 65 64 20 74 65 73 74 49  
1420 20 70 72 6F 67 72 61 6D 4C  
1428 6D 65 20 00 74 68 65 20 8F  
1430 70 72 6F 64 75 63 74 20 65  
1438 63 6F 6E 66 69 67 75 72 A9  
1440 61 74 69 6F 6E 20 62 61 52  
1448 73 65 6C 69 6E 65 20 00 FC  
1450 61 6E 79 20 61 73 73 6F 82  
1458 63 69 61 74 65 64 20 73 69  
1460 75 70 70 6F 72 74 69 6E F5  
1468 67 20 65 6C 65 6D 65 6E 79  
1470 74 20 00 74 68 65 20 69 E2  
1478 6E 63 6F 72 70 6F 72 61 F0  
1480 74 69 6F 6E 20 6F 66 20 63  
1488 61 64 64 69 74 69 6F 6E E8  
1490 61 6C 20 6D 69 73 73 69 B6  
1498 6F 6E 20 63 6F 6E 73 74 D0  
14A0 72 61 69 6E 74 73 20 00 65  
14A8 74 68 65 20 69 6E 64 65 BD  
14B0 70 65 6E 64 65 6E 74 20 D2  
14B8 66 75 6E 63 74 69 6F 6E 32  
14C0 61 6C 20 70 72 69 6E 63 DD  
14C8 69 70 6C 65 20 00 61 20 27  
14D0 70 72 69 6D 61 72 79 20 08  
14D8 69 6E 74 65 72 72 65 6C 51  
14E0 61 74 69 6F 6E 73 68 69 53  
14E8 70 20 62 65 74 77 65 65 08  
14F0 6E 20 73 79 73 74 65 6D 37  
14F8 20 61 6E 64 2F 6F 72 20 8F  
1500 73 75 62 73 79 73 74 65 97  
1508 6D 20 74 65 63 68 6E 6F 2B  
1510 6C 6F 67 69 65 73 20 00 C8  
1518 00 0A 6D 75 73 74 20 75 95  
1520 74 69 6C 69 7A 65 20 61 47  
1528 6E 64 20 62 65 20 66 75 F1  
1530 6E 74 69 6F 6E 61 6C 6C A6  
1538 79 20 69 6E 74 65 72 77 7F  
1540 6F 76 65 6E 20 77 69 74 81  
1548 68 20 00 6D 61 78 69 6D 01  
1550 69 7A 65 73 20 74 68 65 81  
1558 20 70 72 6F 62 61 62 69 6C  
1560 6C 69 74 79 20 6F 66 20 4C  
1568 70 72 6F 6A 65 63 74 20 94  
1570 73 75 63 63 65 73 20 9E  
1578 61 6E 64 20 6D 69 6E 69 8D  
1580 6D 69 7A 65 73 20 74 68 B9  
1588 65 20 63 6F 73 74 20 61 5C  
1590 6E 64 20 74 69 6D 65 20 66  
1598 72 65 71 75 69 72 65 64 0E  
15A0 20 66 6F 72 20 00 61 64 01  
15A8 64 73 20 65 78 70 6C 69 D6  
15B0 63 69 74 20 70 65 72 66 D2  
15B8 6F 72 6D 61 6E 63 65 20 D2  
15C0 6C 69 6D 69 74 73 20 74 FB  
15C8 6F 20 00 6E 65 63 63 65 6A  
15D0 73 73 69 74 61 74 65 73 55  
15D8 20 74 68 61 74 20 75 72 C5  
15E0 67 65 6E 74 20 63 6F 6E 03  
15E8 73 69 64 65 72 61 74 69 52  
15F0 6F 6E 20 62 65 20 61 70 BA  
15F8 70 6C 69 65 64 20 74 6F 1E  
1600 20 00 72 65 71 75 69 72 CE  
1608 65 73 20 63 6F 6E 73 69 32

1610 64 65 72 61 62 6C 65 20 15  
1618 73 79 73 74 65 6D 73 20 66  
1620 61 6E 61 6C 79 73 69 73 9A  
1628 20 61 6E 64 20 74 72 61 F8  
1630 64 65 20 6F 66 66 20 73 FD  
1638 74 75 64 69 65 73 20 74 7B  
1640 6F 20 61 72 72 69 76 65 6E  
1648 20 61 74 20 00 69 73 20 6F  
1650 66 75 72 74 68 65 72 20 86  
1658 63 6F 6D 70 6F 75 6E 64 D3  
1660 65 64 20 74 61 6B 69 6E 76  
1668 67 20 69 6E 74 6F 20 61 40  
1670 63 63 6F 75 6E 74 20 00 32  
1678 70 72 65 73 65 6E 74 73 02  
1680 20 65 78 74 72 65 6D 65 80  
1688 6C 79 20 69 6E 74 65 72 C5  
1690 65 73 74 69 6E 67 20 63 B3  
1698 68 61 6C 6C 65 6E 67 65 EE  
16A0 73 20 74 6F 20 00 72 65 23  
16A8 63 6F 67 6E 69 73 65 73 19  
16B0 20 74 68 65 20 69 6D 70 8C  
16B8 6F 72 74 61 6E 63 65 20 DA  
16C0 6F 66 20 6F 74 68 65 72 ED  
16C8 20 73 79 73 74 65 6D 73 16  
16D0 20 61 6E 64 20 74 68 65 9A  
16D8 20 6E 65 63 65 73 73 69 F8  
16E0 74 79 20 66 6F 72 20 00 6A  
16E8 65 66 66 65 63 74 73 20 FE  
16F0 61 20 73 69 67 6E 69 66 07  
16F8 69 63 61 6E 74 20 69 6D 13

SPACE INVADERS  
NAS-SYS EXECUTE AT 1000

1000 06 0D 3A 01 00 FE 00 2B 84  
1008 02 06 1F 7B 32 0A 1E AF C0  
1010 32 05 1E 21 00 00 22 15 CD  
1018 1E 22 17 1E 22 19 1E 22 18  
1020 1B 1E CD 36 1B 21 CE 08 7E  
1028 CD 3A 1D EF 45 6E 74 65 D7  
1030 72 20 66 69 72 65 20 6B 03  
1038 65 79 20 2B 6E 6F 74 20 DF  
1040 41 2C 58 2C 43 2C 20 6F 3F  
1048 72 20 4E 45 57 4C 49 4E B7  
1050 45 29 20 00 CD 5C 1D FE 32  
1058 58 2B F9 FE 41 2B F5 FE 3B  
1060 43 2B F1 47 3A 0A 1E BB 2D  
1068 2B EA 7B 32 09 1E CD 69 91  
1070 1D 2A 15 1E 3A 04 1E FE 54  
1078 4E 20 05 22 17 1E 18 0C 76  
1080 FE 47 20 05 22 19 1E 18 6B  
1088 03 22 1E 1E 21 4E 09 CD 3E  
1090 3A 1D EF 50 72 61 63 74 E0  
1098 69 63 65 20 59 2F 4E 20 EF  
10A0 3F 20 00 CD 5C 1D CD 69 8E  
10A8 1D FE 59 20 05 3E 50 32 11  
10B0 05 1E 21 CE 09 CD 3A 1D FF  
10B8 EF 54 79 70 65 20 4E 2C F3  
10C0 47 20 6F 72 20 45 20 66 03  
10C8 6F 72 20 73 68 69 6C 6F 8B  
10D0 00 CD 49 1D 21 4E 8A CD 59  
10D8 3A 1D EF 20 20 20 20 CE

1700 70 6C 65 6D 65 6E 74 61 6D  
1708 74 69 6F 6E 20 6F 66 20 EE  
1710 00 61 64 64 73 20 6F 76 C8  
1718 65 72 69 64 69 6E 67 20 31  
1720 70 65 72 66 6F 72 6D 61 93  
1728 6E 63 65 20 63 6F 6E 73 4B  
1730 74 72 61 69 6E 74 73 20 6C  
1738 74 6F 20 00 00 0A 74 68 3B  
1740 65 20 73 6F 70 68 69 73 72  
1748 74 69 63 61 74 65 64 20 5D  
1750 68 61 72 64 77 61 72 65 B5  
1758 2E 00 74 68 65 20 61 6E CD  
1760 74 69 63 69 70 61 74 65 CA  
1768 64 20 66 6F 75 72 74 68 9B  
1770 2D 67 65 6E 65 72 61 74 9A  
1778 69 6F 6E 20 65 71 75 69 A9  
1780 70 6D 65 6E 74 2E 00 74 5D  
1788 68 65 20 73 79 73 74 65 C4  
1790 6D 20 63 6F 6D 70 61 74 BB  
1798 69 62 69 6C 69 74 79 20 C5  
17A0 74 65 73 74 69 6E 67 2E E3  
17A8 00 74 68 65 20 73 74 72 79  
17B0 75 63 74 75 72 61 6C 20 E7  
17B8 64 65 73 69 67 6E 2C 20 95  
17C0 62 61 73 65 64 20 6F 6E D3  
17C8 20 73 79 73 74 65 6D 20 C4  
17D0 65 6E 67 69 6E 65 65 72 34  
17D8 69 6E 67 20 63 6F 6E 63 F0  
17E0 65 70 74 73 2E 00 74 68 BD  
17E8 65 20 70 72 65 6C 69 6D 0D

17F0 69 6E 61 72 79 20 71 75 30  
17F8 61 6C 69 66 69 63 61 74 4C  
1800 69 6F 6E 20 6C 69 6D 69 29  
1808 74 2E 00 74 68 65 20 65 8B  
1810 76 6F 6C 75 74 69 6F 6E AB  
1818 20 6F 66 20 73 70 65 63 F0  
1820 69 66 69 63 61 74 69 6F 80  
1828 6E 73 20 6F 76 65 72 20 1D  
1830 61 20 67 69 76 65 6E 20 02  
1838 74 69 6D 65 20 70 65 72 66  
1840 69 6F 64 2E 00 74 68 65 03  
1848 20 70 68 69 6C 6F 73 6F 7E  
1850 70 68 79 20 6F 66 20 63 31  
1858 6F 6D 6F 6E 61 6C 69 CC  
1860 74 79 20 61 6E 64 20 73 4E  
1868 74 61 6E 64 61 72 64 69 C7  
1870 7A 61 74 69 6F 6E 2E 00 4E  
1878 74 68 65 20 67 72 65 61 90  
1880 74 65 72 20 66 69 67 68 A1  
1888 74 2D 77 6F 72 74 68 69 DE  
1890 6E 65 73 73 20 63 6F 6E C1  
1898 63 65 70 74 2E 00 61 6E 59  
18A0 79 20 64 69 73 63 72 65 CE  
18A8 74 65 20 63 6F 6E 66 69 C8  
18B0 67 75 72 61 74 69 6F 6E 31  
18B8 20 6D 6F 64 65 2E 00 74 37  
18C0 68 65 20 74 6F 74 61 6C E9  
18C8 20 73 79 73 74 65 6D 20 C5  
18D0 72 61 74 69 6F 6E 61 6C 42  
18D8 2E 00 00 00 CD DA 1B DA BA

10E0 20 20 20 4E 20 3D 20 4E 69  
10E8 6F 76 69 63 65 00 CD 49 24  
10F0 1D 21 8E 0A CD 3A 1D EF E9  
10F8 20 20 20 20 20 20 20 08  
1100 47 20 3D 20 47 6F 6F 64 5E  
1108 00 CD 49 1D 21 CE 0A CD 12  
1110 3A 1D EF 20 20 20 20 07  
1118 20 20 20 45 20 3D 20 45 90  
1120 78 70 65 72 74 00 CD 49 7A  
1128 1D CD 5C 1D 32 04 1E FE EE  
1130 4E 20 1E 2A 17 1E 3E 08 72  
1138 32 6E 1D 3E 0E 32 AF 1D 90  
1140 3E 2B 32 B0 1D 3E 50 32 76  
1148 B1 1D 3E 06 32 AD 1D 18 7F  
1150 43 FE 47 20 1E 2A 19 1E 88  
1158 3E 05 32 AE 1D 3E 05 32 1E  
1160 AF 1D 3E 30 32 B0 1D 3E E8  
1168 38 32 B1 1D 3E 06 32 AD D4  
1170 1D 18 21 FE 45 C2 29 11 16  
1178 2A 1B 1E 3E 03 32 AE 1D 2A  
1180 3E 02 32 AF 1D 3E 46 32 85  
1188 B0 1D 3E 14 32 B1 1D 3E F6  
1190 06 32 AD 1D 22 15 1E CD C5  
1198 2C 1D 21 D4 09 CD 3A 1D 14  
11A0 EF 4E 55 4D 42 45 52 20 89  
11A8 4F 46 20 20 50 4C 41 59 C4  
11B0 45 52 53 20 31 2F 32 20 7D  
11B8 3F 00 CD 49 1D 3E 31 32 DC  
11C0 02 1E 32 03 1E CD 5C 1D 8A  
11C8 FE 31 2B 05 3E 32 32 02 D9  
11D0 1E CD 2C 1D 11 CD 0B 21 1F  
11D8 B2 1D 01 27 00 ED B0 3A B7

11E0 02 1E FE 32 2B 0B 11 EA 6F  
11E8 0E 21 09 1D 01 0A 00 ED 13  
11F0 B0 CD 49 1D 21 1C 08 CD F6  
11F8 3A 1D EF 50 4C 41 59 45 CA  
1200 52 20 31 00 CD 49 1D 2A 12  
1208 15 1E DD 21 E5 0B CD 82 8A  
1210 1A 21 58 0B 22 0B 1E 22 2A  
1218 0D 1E CD 42 18 11 F9 22 AB  
1220 21 4A 08 01 31 03 ED B0 77  
1228 3E 03 32 FB 1D 32 FA 1D 0B  
1230 3A 02 1E FE 32 3E 03 2B 35  
1238 01 AF 32 F9 1D 2E 37 3A E1  
1240 05 1E FE 50 20 02 2E 04 17  
1248 7D 32 EA 1D 32 EF 1D 32 80  
1250 E5 1D 07 32 FB 1D 3E 01 F4  
1258 32 E9 1D 32 EE 1D 3E E4 F5  
1260 1D 21 40 00 22 E7 1D 22 3B  
1268 EC 1D 22 F1 1D 21 00 00 D4  
1270 22 0F 1E 22 11 1E 22 13 57  
1278 1E 2B 22 1D 1E 3E 03 32 A3  
1280 FC 1D 3E 01 32 E6 1D 32 51  
1288 EB 1D 32 F0 1D CD B9 1C 83  
1290 21 00 00 22 1F 1E 22 21 65  
1298 1E 22 23 1E 3A AE 1D 32 62  
12A0 F6 1D 3A AD 1D 32 FD 1D 15  
12A8 3E 02 32 F7 1D 3E 06 32 86  
12B0 FE 1D 3E FE 32 7A 0E 32 02  
12B8 7B 0B 32 48 0B 32 49 0B 55  
12C0 3E AD 32 09 0B 32 3A 0B 67  
12C8 3A F6 1D 3D CC 52 15 32 C9  
12D0 F6 1D CD 02 16 3A FD 1D 2E  
12D8 3D CC 65 16 32 FD 1D 3A F4

12E0 FB 1D 3D F5 CC B8 16 F1 C7  
 12E8 28 14 32 FE 1D 3A E5 1D EC  
 12F0 4F 06 0A FE 0B 30 01 47 E2  
 12F8 79 CD 7D 1D 10 FA CD 76 37  
 1300 17 3A FC 1D 3D CC DE 17 7B  
 1308 32 FC 1D 3A AF 1D 47 3A ED  
 1310 E5 1D FE 01 28 08 FE 05 57  
 1318 78 38 03 3A B1 1D CD 12 C5  
 1320 19 3D CC 24 18 CD 92 1E 0B  
 1328 C3 C8 12 31 00 10 3A FA 4D  
 1330 1D 3D 32 FA 1D 47 3A 03 6A  
 1338 1E DD 21 D3 08 21 F8 1D 7B  
 1340 11 F9 1D FE 31 28 05 DD B3  
 1348 21 F3 0B EB 70 78 E6 07 3A  
 1350 47 C6 3D 0D 77 00 1A E6 F4  
 1358 07 B0 C2 33 14 CD AA 1E B0  
 1360 21 1C 08 CD 3A 1D EF 20 EB  
 1368 47 41 4D 43 20 4F 56 45 9F  
 1370 52 20 00 CD 49 1D 2A 13 65  
 1378 1E 3A 03 1E FE 31 20 05 58  
 1380 22 0F 1E 18 03 22 11 1E 4E  
 1388 21 4A 08 11 4B 08 01 80 F3  
 1390 01 36 2D ED B0 21 96 08 56  
 1398 CD 3A 1D EF 54 79 70 65 60  
 13A0 20 52 20 66 6F 72 20 52 FE  
 13A8 65 70 6C 61 79 00 CD 49 EC  
 13B0 1D 21 D6 08 CD 3A 1D EF F2  
 13B8 54 79 70 65 20 43 20 74 64  
 13C0 6F 20 43 68 61 6E 67 65 A8  
 13C8 20 73 68 69 6C 6C 00 CD E7  
 13D0 49 1D 21 16 09 CD 3A 1D AD  
 13D8 EF 54 79 70 65 20 4E 45 2F  
 13E0 57 4C 49 4E 45 20 66 6F 67  
 13E8 72 20 65 78 69 74 00 CD 14  
 13F0 49 1D 21 56 09 CD 3A 1D DD  
 13F8 EF 54 79 70 65 20 50 20 2C  
 1400 66 6F 72 20 50 72 61 63 01  
 1408 74 69 63 65 00 CD 49 1D F4  
 1410 CD 5C 1D 32 05 1E 47 3A 40  
 1418 0A 1E B8 CA 00 00 78 FE 4C  
 1420 52 CA 97 11 FE 50 CA 97 A7  
 1428 11 FE 43 20 E3 CD 2C 1D A7  
 1430 C3 71 10 1A F5 3E 19 CD BB  
 1438 E6 18 3E 2A CD E6 18 2A A7  
 1440 21 1E 7C B7 28 CD ED 4B 32  
 1448 1D 1E 1E 03 36 20 09 1D 34  
 1450 20 FA 2A 25 1E 36 20 2A 6B  
 1458 1F 1E 36 20 F1 47 E6 07 24  
 1460 78 20 27 21 F9 22 11 F9 79  
 1468 1E 3A 03 1E FE 31 28 01 4D  
 1470 EB 21 4A 08 01 31 03 E5 FC  
 1478 D5 C5 ED B0 CD AA 1B C1 16  
 1480 E1 D1 ED B0 CD FD 18 C3 88  
 1488 8D 12 32 FA 1D 11 F9 1E AC  
 1490 3A 03 1E 06 32 FE 31 28 8E  
 1498 05 11 F9 22 06 31 78 32 BE  
 14A0 03 1E 21 4A 08 01 31 03 7D  
 14A8 F5 ED B0 CD AA 1B 21 F9 FA  
 14B0 1E F1 20 03 21 F9 22 11 43  
 14B8 4A 08 01 31 03 ED B0 CD BD  
 14C0 FD 18 2A 13 1E 3A 03 1E 9F  
 14C8 FE 31 20 08 22 11 1E 2A AE  
 14D0 0F 1E 18 06 22 0F 1E 2A AB  
 14D8 11 1E 22 13 1E 3A 03 1E C9

14E0 FE 31 20 37 3A E4 1D 32 E7  
 14E8 EE 1D 3A E9 1D 32 E4 1D 7A  
 14F0 3A E5 1D 32 EF 1D 3A EA A2  
 14F8 1D 32 E5 1D 07 32 FB 1D AE  
 1500 3A E6 1D 32 F0 1D 3A EB B6  
 1508 1D 32 E6 1D 2A E7 1D 22 BF  
 1510 F1 1D 2A EC 1D 22 E7 1D 8C  
 1518 C3 8D 12 3A E4 1D 32 E9 E5  
 1520 1D 3A EE 1D 32 E4 1D 3A 04  
 1528 E5 1D 32 EA 1D 3A EF 1D BE  
 1530 32 E5 1D 07 32 FB 1D 3A 04  
 1538 E6 1D 32 EB 1D 3A F0 1D D1  
 1540 32 E6 1D 2A E7 1D 22 EC C6  
 1548 1D 2A F1 2D 22 E7 1D C3 9B  
 1550 8D 12 2A 21 1E 7C B7 20 C0  
 1558 39 3A B0 1D CD 12 19 3D E2  
 1560 C2 FE 15 3E 10 CD 12 19 90  
 1568 3D C2 FE 15 ED 4B 1D 1E 02  
 1570 21 0A 08 78 01 01 00 B7 E9  
 1578 20 06 21 39 08 01 FF FF 14  
 1580 ED 43 1D 1E 22 21 1E 36 97  
 1588 0D 09 36 0A 09 36 09 C3 FE  
 1590 FE 15 7E FE 0D 28 16 FE 7D  
 1598 09 28 12 3A FE 1D 3D 32 B4  
 15A0 FE 1D C2 FE 15 3E 06 32 1E  
 15A8 FE 1D C3 E5 15 ED 4B 1D EA  
 15B0 1E 36 20 09 22 21 1E 7E 21  
 15B8 FE A0 CA FE 15 3E 09 08 97  
 15C0 3E 0D C8 40 28 01 08 77 03  
 15C8 09 7E FE A0 CA FE 15 36 15  
 15D0 0A 09 7E FE A0 CA FE 15 F1  
 15D8 FE 21 20 05 CD 5C 19 18 BB  
 15E0 1D 08 77 18 19 ED 4B 1D 17  
 15E8 1E 1E 03 36 20 09 1D 20 D8  
 15F0 FA 3E A0 32 09 08 32 3A BC  
 15F8 08 26 00 22 21 1E 3A AE B4  
 1600 1D C9 11 0A 08 01 C0 FF DF  
 1608 2A 1F 1E 7C B7 C8 3A F7 B1  
 1610 1D 3D 32 F7 1D C0 3E 02 C6  
 1618 32 F7 1D 7E FE 21 20 3F 70  
 1620 36 20 09 B7 ED 52 19 38 DC  
 1628 36 7E FE 20 28 04 FE 2A 64  
 1630 20 04 36 21 18 2B 3E 21 63  
 1638 CD B9 19 28 22 FE 19 28 76  
 1640 0B CD E6 19 20 0A CD EF 13  
 1648 19 CD A7 1A 36 2A 18 0F 8C  
 1650 FE 0D CA 5C 19 FE 0A CA B2  
 1658 5C 19 FE 09 CA 5C 19 26 4F  
 1660 0D 22 1F 1E C9 21 B9 0B B3  
 1668 11 BA 0B 01 40 00 7E FE 11  
 1670 2A 20 02 36 20 B7 FE 19 20 5F  
 1678 30 36 20 09 B7 ED 52 19 2C  
 1680 30 21 7E 06 19 FE 20 28 CA  
 1688 19 FE 2A 28 15 06 2A FE 4A  
 1690 21 28 0F 3E 19 CD B9 19 F4  
 1698 28 09 CD E6 19 C2 2E 13 AB  
 16A0 18 01 70 01 40 00 B7 ED 24  
 16A8 42 2B 01 0A 08 B7 ED 42 24  
 16B0 09 D2 6B 16 3A AD 1D C9 EF  
 16B8 3A E5 1D 07 32 FB 1D 21 7C  
 16C0 79 08 11 7A 0B 01 40 00 2E  
 16C8 3A E6 1D B7 20 06 21 4A 63  
 16D0 08 11 4B 0B 7E CD E6 19 9F  
 16D8 CA 25 17 09 B7 ED 52 19 0C

16E0 38 F2 3E 01 32 E4 1D 3A CC  
 16E8 E6 1D 21 79 0B 01 FF FF A5  
 16F0 B7 20 06 21 4A 08 01 01 58  
 16F8 00 09 7E FE 20 28 FA CD A2  
 1700 E6 19 28 05 FE FE C8 18 1F  
 1708 F0 36 20 B7 ED 42 08 7E D1  
 1710 FE 21 20 0A 08 CD EF 19 4D  
 1718 CD A7 1A 3E 2A 08 08 77 AC  
 1720 09 09 C3 F9 16 3A E4 1D 56  
 1728 B7 CA E2 16 21 7A 0B 11 6F  
 1730 7A 0B 01 40 00 2B 7E FE B4  
 1738 FE 28 2E CD E6 19 20 F5 84  
 1740 36 20 09 B7 ED 52 19 38 FD  
 1748 08 3E 61 32 FA 1D C3 2B 3D  
 1750 13 47 7E FE 21 20 09 7B FF  
 1758 CD EF 19 CD A7 1A 06 2A 02  
 1760 70 01 40 00 B7 ED 42 18 26  
 1768 CC 3A E6 1D EE 01 32 E6 8F  
 1770 1D AF 32 E4 1D C9 3A 01 8A  
 1778 00 FE 00 28 05 CD 69 00 F0  
 1780 18 02 DF 61 F5 3A 05 C0 31  
 1788 01 FF FF FE 10 28 09 03 ED  
 1790 3A 02 0C FE 10 28 01 03 21  
 1798 ED 43 23 1E F1 FE 50 20 7F  
 17A0 04 CD 5C 1D C9 21 09 1E 12  
 17A8 BE C0 2A 1F 1E 7C B7 C0 97  
 17B0 2A 25 1E 01 C0 FF 09 3E 3B  
 17B8 21 CD B9 19 28 1A 06 21 FB  
 17C0 FE 20 28 0E FE 2A 28 0A 85  
 17C8 06 2A FE 19 C4 EF 19 C4 B6  
 17D0 A7 1A 70 78 FE 21 28 02 D9  
 17D8 26 00 22 1F 1E C9 ED 4B 75  
 17E0 23 1E 79 B7 28 3B 2A 25 1A  
 17E8 1E 09 2B 7D FE 89 28 31 AE  
 17F0 23 23 7D FE BA 28 2A 2E FF  
 17F8 B7 ED 42 B7 ED 42 36 20 31  
 1800 09 3A F3 1D 57 3A F5 1D 0E  
 1808 5F 78 B7 28 03 7A 53 5F 05  
 1810 72 09 3A F4 1D 77 22 25 AC  
 1818 1E 09 7E FE 20 C2 2B 13 F3  
 1820 73 3E 03 C9 3E 08 4F C8 15  
 1828 39 CD 12 19 91 4F 06 00 57  
 1830 F2 34 18 05 2A 25 1E 09 01  
 1838 7D FE BA C8 FE 89 C4 27 8F  
 1840 19 C9 E5 21 CA 0A 1E 20 52  
 1848 3A 00 1E 57 0E 04 06 04 2B  
 1850 73 23 10 FC 77 23 72 23 39  
 1858 72 23 77 23 06 04 73 23 3F  
 1860 10 FC 0D 20 E9 21 0A 0B D0  
 1868 0E 04 06 03 73 23 10 FC 3D  
 1870 06 06 72 23 10 FC 06 03 3E  
 1878 73 23 10 FC 0D 20 EB 21 6B  
 1880 4A 08 0E 04 06 03 73 23 9E  
 1888 10 FC 72 23 72 23 77 23 70  
 1890 77 23 72 23 72 23 06 03 75  
 1898 73 23 10 FC 0D 20 E5 CD 31  
 18A0 FD 18 3A 05 1E FE 50 28 A0  
 18A8 1A E1 01 40 00 3E 07 CD 0E  
 18B0 D9 18 3E 18 CD 09 18 CD 9A  
 18B8 D9 18 3E 0F CD 09 18 CD 69  
 18C0 D9 18 C9 DD E1 11 07 01 69  
 18C8 DD 19 3E 07 DD 77 00 DD 4C  
 18D0 77 02 DD 77 04 DD 77 06 13  
 18D8 C9 E5 06 08 77 23 36 20 9F

18E0 23 10 F9 E1 09 C9 21 BA B2  
 18E8 08 01 31 03 ED B1 ED 2B E6  
 18F0 36 20 23 18 F7 AF 47 CD 53  
 18F8 7D 1D 10 FB C9 21 BA 0B 34  
 1900 3A F3 1D 77 23 3A F4 1D 48  
 1908 77 22 25 1E 23 3A F5 1D 6C  
 1910 77 C9 C5 E5 21 FF 1D 47 97  
 1918 ED 5F 86 38 01 3D 77 90 80  
 1920 30 FD 80 3C E1 C1 C9 01 8E  
 1928 40 00 11 4A 08 B7 ED 42 CA  
 1930 B7 ED 52 19 D8 7E CD E6 61  
 1938 19 20 F2 09 7E 06 19 FE 20  
 1940 20 28 17 FE 19 28 13 FE 08  
 1948 2A 28 0F 06 2A FE 21 28 39  
 1950 09 3E 19 CD B9 19 C8 C3 F3  
 1958 28 13 70 C9 26 00 22 1F 4F  
 1960 1E 2A 21 1E ED 4B 1D 1E 73  
 1968 78 B7 28 02 2B 2B 36 20 86  
 1970 3E 04 CD 12 19 3D 28 03 2E  
 1978 C6 30 77 23 01 00 00 B7 09  
 1980 28 16 01 64 00 FE 31 28 93  
 1988 0F 01 C8 00 FE 32 28 08 09  
 1990 01 2C 01 36 30 AF 18 15 19  
 1998 B7 3E 32 36 35 28 0E 3E B7  
 19A0 02 CD 12 19 3D 36 30 28 7E  
 19A8 04 3E 32 36 35 23 36 30 29  
 19B0 6F 26 00 09 E5 C1 C3 04 04  
 19B8 1A E5 C5 47 4E 21 B5 1D ED  
 19C0 3A F4 1D FE 0E 28 0E 21 84  
 19C8 9D 1D 78 FE 21 20 03 21 76  
 19D0 A5 1D 79 01 03 00 ED A1 B6  
 19D8 28 07 23 EA D6 19 C1 E1 BE  
 19E0 C9 7E C1 E1 77 C9 FE 07 27  
 19E8 C8 FE 18 C8 FE 0F C9 E5 62  
 19F0 C5 F5 01 06 00 21 B8 1D 93  
 19F8 ED B1 4E 06 00 CD 04 1A EE  
 1A00 F1 C1 E1 C9 D5 E5 F5 2A 4F  
 1A08 13 1E E5 3A 03 1E DD 21 91  
 1A10 CD 08 FE 31 28 04 DD 21 5B  
 1A18 ED 0B B7 09 30 03 21 FF 3D  
 1A20 FF 22 13 1E CD 82 1A 2A 1F  
 1A28 13 1E 11 DD 05 B7 ED 52 5B  
 1A30 19 38 33 E1 B7 ED 52 19 BE  
 1A38 38 14 E5 11 94 11 2A 13 76  
 1A40 1E B7 ED 52 19 38 1F E1 BF  
 1A48 B7 ED 52 19 30 19 3A FA EE  
 1A50 1D 3C 32 FA 1D 21 D3 0B 0B  
 1A58 11 F3 0B 3A 03 1E FE 31 0E  
 1A60 2B 01 EB 34 18 01 E1 2A E6  
 1A68 13 1E ED 5B 15 1E B7 ED D2  
 1A70 52 19 38 0A 22 15 1E DD 69  
 1A78 21 E5 0B CD 82 1A F1 E1 DE  
 1A80 D1 C9 FD 21 93 1D AF FD AE  
 1A88 5E 00 FD 56 01 B7 ED 52 4A  
 1A90 38 03 3C 18 F8 19 C6 30 40  
 1A98 DD 77 00 DD 23 FD 23 FD 23  
 1AA0 23 3E 01 B8 20 E0 C9 36 D6  
 1AA8 2A 3A E5 1D 3D 32 E5 1D 99  
 1AB0 C0 2E 37 3A 05 1E FE 50 9A  
 1AB8 20 02 2E 04 7D 32 E5 1D 07  
 1AC0 3E 03 32 FC 1D 3E 01 32 07  
 1AC8 E4 1D 3A 03 1E FE 31 2A 97  
 1AD0 0B 1E 28 03 2A 0D 1E 11 A4  
 1AD8 58 0B B7 ED 52 19 01 40 AZ

1AE0 00 28 23 F5 3A 04 1E 11 A7  
 1AE8 18 09 FE 4E 28 0A 11 58 0A  
 1AF0 09 FE 47 28 03 11 98 09 35  
 1AF8 F1 B7 ED 52 19 01 CD FF D2  
 1B00 28 04 ED 4B E7 1D ED 43 B3  
 1B08 E7 1D 09 FE 31 28 05 22 AE  
 1B10 0D 1E 18 03 22 0B 1E CD 89  
 1B18 F5 18 E5 21 0A 08 11 0A 73  
 1B20 08 13 36 20 01 B0 03 ED 4D  
 1B28 B0 E1 CD 42 18 E1 3E 01 1E  
 1B30 32 E6 1D C3 90 12 CD 2C DE  
 1B38 1D 21 4E 08 CD 3A 1D EF FA  
 1B40 47 72 61 70 68 69 63 73 8C  
 1B48 20 59 2F 4E 3F 20 00 CD B5  
 1B50 5C 1D CD 69 1D FE 59 28 B6  
 1B58 18 3E 3D 32 F3 1D 32 F5 6F  
 1B60 1D 3E 0E 32 F4 1D 3E 7F E4  
 1B68 32 00 1E 3E 25 32 01 1E B7  
 1B70 C9 3E A7 32 F3 1D 3E AF 68  
 1B78 32 F4 1D 3E AE 32 F5 1D 03  
 1B80 3E DF 32 00 1E 3E 9F 32 17  
 1B88 01 1E 3E 0F D3 06 AF D3 6A  
 1B90 04 C9 3E 31 CD 12 19 3D 04  
 1B98 CD 3E 31 CD 12 19 3D 4E 66  
 1BA0 06 00 21 89 08 09 CD 27 73  
 1BA8 19 C9 21 3A 08 11 30 00 49  
 1BB0 06 0F 3E FF 0E 10 77 23 D5  
 1BB8 0D 20 FB 19 10 F6 3E 02 5A  
 1BC0 CD 12 19 3C 32 08 1E ED 54  
 1BC8 5B 25 1E DD 2A 25 1E DD A8  
 1BD0 3C FF 20 DD 36 01 20 DD 51  
 1BD8 21 00 08 21 27 1E 3E 1E DE  
 1BE0 CD 12 19 C6 04 32 07 1E 14  
 1BE8 47 3E 14 CD 12 19 77 23 2E  
 1BF0 C5 D5 E5 3E 06 CD 12 19 C6  
 1BF8 47 21 00 00 11 40 00 B7 83  
 1C00 ED 52 10 FB 3E 0F CD 12 92  
 1C08 19 D6 08 4F 17 9F 47 09 70  
 1C10 E3 C1 71 23 70 D1 C1 23 89  
 1C18 73 23 72 23 3A 00 1E FE B5  
 1C20 7F 20 07 3E 1F CD 12 19 37  
 1C28 18 07 3E 7F CD 12 19 C8 E3  
 1C30 FF 77 23 10 B4 11 0A 08 CC  
 1C38 DD 21 27 1E AF 32 06 1E 9C  
 1C40 3A 07 1E 47 C5 DD 7E 08 22  
 1C48 B7 28 3C 32 0E 1E DD 35 E7  
 1C50 00 DD 4E 01 DD 46 02 DD 9A  
 1C58 6E 03 DD 66 04 36 20 3D BF  
 1C60 28 25 09 AF ED 52 19 38 11  
 1C68 1E 7E FE 20 28 0C 3C 28 D6  
 1C70 16 3D CD E6 19 20 03 AF 7D  
 1C78 18 DD DD 75 03 DD 74 04 63  
 1C80 DD 7E 05 77 DD 7E 00 DD AE  
 1C88 77 00 01 06 00 DD 09 C1 C9  
 1C90 10 E2 3A 06 1E B7 28 08 B3  
 1C98 3E 0A CD B0 1C C3 38 1C AC  
 1CA0 3A 08 1E 3D C8 32 08 1E 79  
 1CA8 C8 27 CD B0 1C C3 C7 1B F4  
 1CB0 C5 47 CD 7D 1D 10 FB C1 0E  
 1CB8 C9 3A 03 1E 57 1E 20 21 AE  
 1CC0 CD 0E FE 31 28 03 21 ED 1C  
 1CC8 0B 0E 18 E5 21 00 00 08 33  
 1CD0 CD 04 1A FD 21 1C 08 FD 16  
 1CD8 36 00 50 FD 36 01 4C FD F7

1CE0 36 02 41 FD 36 03 59 FD 01  
 1CE8 36 04 45 FD 36 05 52 FD 0A  
 1CF0 36 06 20 FD 72 07 06 10 F4  
 1CF8 AF CD 7D 1D 10 FA 06 05 3F  
 1D00 DD E1 DD E5 DD 73 00 DD CA  
 1D08 23 10 F9 FD 21 1C 08 06 99  
 1D10 08 FD 73 00 FD 23 10 F9 CE  
 1D18 06 10 AF CD 7D 1D 10 FA 6B  
 1D20 08 3D 20 A8 0E 00 CD 04 29  
 1D28 1A DD E1 C9 21 00 08 36 45  
 1D30 20 11 01 08 01 FF 03 ED 77  
 1D38 B0 C9 3A 01 00 FE 00 28 2F  
 1D40 04 22 18 0C C9 22 29 0C C7  
 1D48 C9 3A 01 00 FE 00 28 06 95  
 1D50 2A 18 0C 36 20 C9 2A 29 2D  
 1D58 0C 36 20 C9 3A 01 00 FE D9  
 1D60 00 2B 04 CD 3E 00 C9 CF 4C  
 1D68 C9 F5 3A 01 00 FE 00 28 A4  
 1D70 06 2A 18 0C F1 77 C9 2A 3C  
 1D78 29 0C F1 77 C9 F5 F1 F5 D6  
 1D80 F1 3D 20 F9 C9 7F 25 25 76  
 1D88 20 00 00 07 1E 18 14 03 25  
 1D90 0A 00 00 10 27 E8 03 64 3D  
 1D98 00 0A 00 01 00 DF D3 D3 45  
 1DA0 20 DC 20 00 00 DF DC D3 67  
 1DA8 20 DC 20 00 00 06 08 0E FD  
 1DB0 28 50 30 30 30 30 2F 64  
 1DB8 33 20 20 20 4B 49 47 48 8B  
 1DC0 45 53 54 20 53 43 4F 52 20  
 1DC8 45 20 20 20 20 30 20 0A  
 1DD0 20 20 30 30 30 30 2F 4C  
 1DD8 33 20 20 20 20 20 20 08  
 1DE0 20 20 20 00 01 27 00 40 C5  
 1DE8 00 01 37 01 40 00 01 37 B6  
 1DF0 01 40 00 3D 0E 3D 06 02 DE  
 1DF8 00 00 00 4D 03 01 06 63 CF  
 1E00 7F 25 31 31 00 00 00 0A 2E  
 1E08 01 00 00 58 08 58 08 E6 DA  
 1E10 00 00 00 E6 00 00 00 00 14  
 1E18 00 00 00 00 00 FF FF ED 21  
 1E20 00 00 00 FF FF AC 08 00 F3  
 1E28 FA FE AC 0B D1 00 46 FF 0B  
 1E30 38 0A 9A 00 C5 FF 36 0B 2F  
 1E38 A4 00 82 FE B8 08 9D 00 CF  
 1E40 BC FE 24 09 92 00 00 FF D6  
 1E48 AC 0B CD 00 0A FE 9A 08 74  
 1E50 A7 00 47 FF F3 0A E7 00 3F  
 1E58 06 FF B8 09 EE 00 7D FE A5  
 1E60 A6 08 E4 00 BC FF 24 0B FA  
 1E68 A4 00 44 FF 34 0A BC 00 37  
 1E70 7C FE 28 0A B8 00 86 FE 79  
 1E78 B8 0B A3 00 B8 FE AC 0B 69  
 1E80 D1 00 7E FF 1E 08 CC 00 DE  
 1E88 BE FE 28 09 84 00 87 FE 9C  
 1E90 33 0A C1 00 C0 FE 2C 09 9F  
 1E98 8E 00 86 FE B8 08 81 00 09  
 1EA0 84 FF B4 0A 93 00 45 FF D6  
 1EA8 36 0A D8 00 BA FF 20 0B C2  
 1EB0 E0 00 C1 FE 2E 09 BC 00 60  
 1EB8 B8 FE 22 09 94 00 04 FF 51  
 1EC0 B8 08 C1 00 41 FF 0B 08