# inmc news issue 6

February/March 1980

CONTENTS
========



INMC Services.
==============

INMC News. This is the main function of the INMC. The newsletter appears every 6-8 weeks, depending on whether you send us articles or not !

Back Issues. Back Issues of the INMC News are available for 70p each + 15p p & p per order.(40p Overseas)

Program Library. Programs sent in by members are vetted by the committee and added to the library list. We admit to being a lot behind on this one !!

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

PLEASE NOTE: WE ARE NOT NASCOM !! This address is purely a postbox and we cannot arrange to send you leaflets or answer sales queries.

Nascom Microcomputers

# news release

Nascom Microcomputers Ltd. which has become Britain's leading Micro board Company announce a re-organisation within the Company and also, sadly, the resignation of their Marketing Director, Kerr Borland.

The John Marshall - Kerr Borland team started Nascom in January 1978. They have steered it through many birth - pains to reach the dominant market position they have now achieved. This very success has resulted in the split.

Kerr Borland has always said that his interests lay in promoting and creating a sound market base for new products. Once the product was successfully launched, others were better suited to continue. For this reason he is leaving Nascom to set up a new promotions company called Product Launch. This company will cope with all the specialised marketing that is needed in the short term when presenting new products to the market place.

Nascom immediatly become his first customer and he will continue to launch Nascoms new product range due throughout 1980.

In the wake of Kerr Borlands departure, Nascom are re-organsing their Sales and Marketing function. Sales Management will be taken over by another Board member, Bill Bulman, and the company will create a new Marketing section separate from the Sales function.

Nascom has a huge user base throughout Europe and Scand-inavia and an extensive distributor network. In two years it has reached a market position that has encouraged many small companies to produce peripherals for use with Nascom equipment.

This firm base which spreads across industrial,educational and home user areas makes an excellent springboard for attacking the 1980's market place.

# NASPEN

Word processors !!!! Two words which are becoming as rapidly devalued amongst computing circles as the words 'High Fidelity' are already amongst the audiophiles. Now its not our purpose to bandy definitions, but when definitions become meaningless, then their use ought to be actively discouraged. All this comes to mind, as, at a recent exhibition the words 'Word Processor' seemed to be thrown about with such abandon that any potential customer could only have been confused by the multiplicity (or lack of) specifications which were on display. We saw one beastie which for a modest one hundred and fifty pounds had just about as much in its 4k as half the new Nascom 2K Naspen at about one fifth the price; and they had the cheek to call it a 'Word Processor'!

Which brings us to the subject of Naspen. We have made passing mention of this before, and the results may be seen in parts of this, and the last two newsletters. We have been priviledged to have been loaned preproduction samples to play with, and to say the least we are impressed. And it is not generalized into a 'Word Processor', but is much more specifically a 'Text Editing and Formatting' package and a pretty superior one at that. Now that Naspen is becoming available, it is perhaps time to talk about it a bit, and tell you what it is, and what it does.

Naspen is a program supplied in 2 2708 EPROMs for use with expanded Nascom 1 or 2. There are two versions, VS.1 for Nas-sys monitors and VT.1 for Nasbug T4, it is not suitable for Nasbug T2 and B-Bug. It is really intended for use with 16K of RAM, but at a push 8K can be used, provided the pointer to top of RAM is altered accordingly. Naspen is aimed at people who would want to enter text into their Nascom, and then print it (or just save it on tape for reference). Its main uses would be in the preparation of documents, drafts (including INMC newsletters), repetetive letters, price lists etc. In other words small business applications where editing and correction of printed text would show the benefit of automation.

Because Naspen was designed to be used by inexperienced operators, all the commands are direct acting, and single character only, for instance, in writing a tape, there is no need to tell Naspen where to write from and to, and then enter the line. The W command acts directly, and automatically starts writing to tape two seconds after 'W' has been typed. A major feature is the repeat keyboard which repeats characters at a rate determined by the operator. In the Insert and Append modes the text being written is always displayed, whilst in the edit mode the text may be moved around at will. It is best to imagine the screen as being a window onto the text buffer, and the window is allowed to move over the text buffer. There are no problems with lines longer than 48 characters as the lines simply wrap round the screen (a little disconcerting at first, but after an hour or so, no longer a problem).

The documentation supplied with Naspen is thorough and detailed and again written with the inexperienced operator in mind. We understand that the documentation was tried out on a typist who had never touched a Nascom in her life, and then re-written in the light of her mistakes. We found it accurate if a little clinical in style. The demonstration examples stopped rather abruptly. All the commands were

explained in detail, but it is pity that no abrieviated list of commands was given.

Essentially there are two modes, the command/edit mode, and the append/insert mode. In all there are 44 commands (43 in the VT version) which are may be considered in six groups.

1) A and I are the append and insert modes. Append places text at the end of the text, Insert places text within the text. Lower case commands c, d, and i, change, delete and insert single characters in the edit mode. D deletes a line in the edit mode. M is used to move chunks of text around within the text.

2) C, G, L, S, s, X, 1, 2, 3 and 4 are all formatting commands. 1, 2, 3 and 4 set up the line and page lengths. L left justifies the text to width, whilst S and s left/right justify the text to width, inserting extra spaces as required. G generates pages to length, whilst C and X remove the affects of formatting.

3) Finding bits of text is important, and the F, f and a commands will find strings of up to 40 characters within the text.

4) Commands for entering and saving text to tape are covered by the R, W and V (VS version only) commands. There is also a special tape command, the J command, which will insert an incoming tape in front of text already the Naspen.

5) Moving the cursor about within the text is perhaps the most important facility of any text editor, and Naspen has no less than 14 cursor move commands. The text and cursor may be set back to the start, the cursor may skip backwards and forwards in pages, lines, tens of characters or singly. Making it very easy to position the cursor anywhere in the text.

6) The last group are all unrelated. The K command will kill text and because of its irreversible action is protected against accidental use by the Y command. N returns Naspen to the monitor, whilst P prints the text. Commands 5 and 6 set the repetition rate of the repeat keyboard.

In use we found Naspen very easy to get on with, and in preparing INMC pages it is wonderful, as none of us qualify as typists, the ability to skip through what we have done, then correct it with out having to retype the lot has saved an enormous ammount of time. As mentioned elsewhere, we shunt 'Pen' tapes about, edit them and finally print them on an old IBM printer. Incidentally, you may have noticed we have avoided the 'pound sign' in this text. Not because 'Pen' can't use it, but because the IBM golfball doesn't have one.

In all Naspen seems a very versatile and worthwhile addition to the Nascom stable for those who want a reliable text editor. We have found that it covers all normal (sic) facilities in a very easy to use package. It takes about three hours to get conversant with it. And as a final recommendation, takes a look at these two pages. It's a *real* pity it doesn't find the spelling mistakes as well!! That would be really something to shout about.

# N1-2

## THE STORY OF THE NASCOM ONE-TWO

Or Why Two Computers are Better than One by Richard Beal

Is the Nascom One-Two a marvellous product of the distant future ? No, it is already here, and consists of a Nascom 1 connected to a Nascom 2 to form a new and versatile computer system with the following features:

(a) Ability to read data from Nascom 1 tapes directly into a Nascom 2. This works even with an unexpanded system.

(b) Intelligent giant print buffer for serial printers such as the Nascom Imp or Teletype. This enables output from ZEAP, BASIC or NASPEN to be routed to the Nascom 1 memory virtually without delay, and with automatic compression of blanks, to save valuable RAM space. Printing is completely independent of the Nascom 2, and can proceed at the same time as more data is being sent across at high speed. If the print buffer becomes full, the Nascom 2 will automatically wait until printing starts, or if required the print buffer can be cleared if the output is not, after all, needed. Printing can be paused at any time, and the Nascom 1 display shows the number of characters waiting to be printed.

(c) Almost instantaneous transfer of the whole of the contents of memory from one machine to the other, allowing very fast recovery of programs and data when testing machine code programs. For example ZEAP could be used on the Nascom 2, and the source code and generated machine code both held in memory. Then before testing the program, which could well crash and change the contents of any part of the memory, the whole memory is copied in a few seconds to the Nascom 1. When the Nascom 2 crashes, it is simply reset, and the data brought back. Within seconds the ZEAP source can be edited and re-assembled and the process repeated.

These facilities are very useful, and mean that if you have bought a Nascom 2, your Nascom 1 can continue to be used instead of putting it in the attic, or trying to sell it, which is rather sad after all the effort needed to build it. It also gives you another reason for buying a Nascom 2 if you already have a Nascom 1 !

At this point you may be wondering what sort of equipment you need for all this. If you have a Nascom 1 and Nascom 2, all you need is ..... NOTHING!!!

Obviously if the Nascom 1 is not expanded, only feature (a) is possible. The only hardware work required is to connect various lines from the Nascom 2 26 way ribbon cable to the Nascom 1 PIO sockets. Only one port on each machine is used, with all data transfer being interrupt driven using the handshake lines. To understand this, read the PIO technical manual until you are sure you know it all. (I have read it about 30 times and am still not completely confident). Alternatively, read on.

Connect the 8 data lines of port B on the Nascom 2 to the coresponding 8 data lines of port B on the Nascom 1. Connect the ground line of the N2 to pin 9 (ground) of the N1. Connect the B READY line from the N2 to the B STRB on the N1 and the B READY line on the N1 to the B STRB line on the N2. Do not connect the power lines together! You may choose to wire a switch so you can see the N1 or N2 display on the monitor screen, but you don't really need the N1 display. If you wire such a switch, wire the earth lines together.

One further point on hardware is that for some strange reason the memory board links togther the IEI and IEO. This doesn't matter on the N1, but the N2 brings these lines out correctly to NASBUS, and the PIO won't interrupt unless IEI is high. Simply look at the edge of the memory board, next to line 19, and you will see that it is connected to the next line. Break this join by scraping away with a screwdriver. (Ed. Don't do that, the memory board is correct, it is the mini Vero bus on the N2 which is wrong. Break track 19 on the Vero board. See note under the article about PIOs.)

As you will have realised, the key to the Nascom One-Two is in the software. We will just cover feature (a) the cassette input via Nascom 1. This is extremely useful for converting your tapes from N1 to N2 format.

Since giving you the program listings would make it much too easy, here is a description of the little programs which you need to put into each computer. (The listings will be put in the library in due course.)

## Cassete input transfer program for Nascom 1

This program initializes the PIO and then goes into a loop which does nothing at all. However, each time there is an interrupt the interrupt routine waits until there is a character from the cassette input, and then outputs it to the PIO and returns. The full list of actions is:

Disable the CPU interrupts with a DI instruction.
Disable PIO (03H to control port).
Ensure PIO is happy by pushing address of next instruction on to the stack, and then using a RETI.
Load I register.
Set interrupt mode 2, with IM2.
Output the interrupt vector (low order half address of the interrupt address table).
Output 0FH to PIO to set it to output mode.
Set the interrupt control word by outputting 87H to the PIO.
Enable the CPU interrupts with an EI instruction.
Go into a tight loop (actual code 18 FEH).

That is the end of the initialization program, which is executed.

Then you need an interrupt address table. The high order part of the address of this table has been stored in the I register, and the low order part has been sent to the PIO as the interrupt vector (this must be an 'even' number, ie. last bit = 0). The interrupt address table actually contains just one address, which is the address of the

interrupt routine itself. This address is placed at the start of the table, low order byte first, as usual.

The interrupt routine itself calls its own copy of the SRLIN routine and loops until the routine returns with the Carry set. It then outputs the character to the PIO, enables the CPU interrupts and immediately returns. This last bit of code reads:

```
    OUT (5), A
    EI
    RETI
```

## Cassette input program for Nascom 2

The other half of the software lives in the Nascom 2. The program initializes the PIO just like the other program, except that the PIO mode is 4FH, for input mode. After setting up the PIO, the CPU interrupts are not enabled. Instead a dummy READ to the PIO is made to ensure that handshaking starts. Then a version of the READ routine, copied out of NAS-SYS, is executed. If the routine ends, control returns to NAS-SYS. Instead of the need for changing tables and RST RIN to get an input, normal calls are made to a new routine called CIN. This routine reads as follows:

```
    CIN   OR A
          EI
    SLP   JR NC SLP
          RET
```

This loops until an interrupt occurs, and the interrupt routine sets the Carry flag, and puts the input character into A. The interrupt routine reads simply:

```
    PROC  IN A, (5)
          SCF
          RETI
```

Note that it does not re-enable CPU interrupts.

As you can see, the use of the PIO requires a bit of thought, but if you read the notes above, and have a look at the PIO manual, you should be able to write these routines for yourself, exactly as you wish, which is much more interesting than typing in some HEX listings from the library.

Please write in and let others know if you have any other uses for two (or more) machine systems. How about a controlling machine loading other slave machines with data, programs, perhaps even interpreters as well, then doing other work while the slave machine performs the subordinate tasks. This would be one way to tackle a multi-user operating system with resources such as printer or disc attached to only some machines.

Richard Beal

PS You could use two Nascom 2 computers, if you don't already have a Nascom 1.

PPS Don't blame me if you get your wires mixed up and blow up both computers.

SOFT SPOTS
==========

By Richard Beal (and others)

This tip presupposes that you are a person who assembles machine code in HEX, in memory, and then wants to convert it to DATA statements for use as machine code subroutines with the 8K BASIC. Assuming Nas-sys is in use, try this:

Assemble the machine code routines, in HEX, at the correct location in memory (and perhaps try them if possible). Go into BASIC (making sure that the free memory space does not overwrite the machine code routines). Then use the following command:

```
WIDTH 40:FOR A = B TO C STEP 2:PRINT DEEK(A);:NEXT
```

Where B is the start address in decimal, and C is the end address. The BASIC will print a series of numbers, stopping well short of the right hand edge of the screen. Using the Nas-sys edit commands, edit in line numbers and commas, and the job is done. No mistakes through trying to calculate the decimal equivalents of the HEX, or through copying the numbers wrong. What's more the data is in double byte form, which may be loaded down using DOKE commands, taking half the time that it would take to POKE the data down.

---

Suppose you want to test to see if one of the 16 bit registers HL, DE or BC is zero. For example, this could be at the end of a loop that was counting down. A version of this in a magazine had the following:

```
    DEC DE
    LD A, D
    OR A
    JR NZ LOOP
    LD A, E
    OR A
    JR NZ LOOP
```

It would be a lot easier to put:

```
    DEC DE
    LD A, D
    OR E
    JR NZ LOOP
```

---

Suppose you want to jump to a certain address in your program, and that this address is currently stored in the HL register. You can just code:

```
    JP (HL)
```

This works very well.

Now suppose that you need to have set HL to a certain value when the jump is made. The method above is useless because HL can't be at two different values at once. So code:

```
    LD HL, value of HL
    PUSH HL
    .
    .
    LD HL, address to jump to
    EX (SP), HL
    RET
```

or:

```
    LD HL, address to jump to
    PUSH HL
    LD HL, value of HL
    RET
```

The second method is the simplest, but often the value of HL is already on the stack, making the first method the most commonly used.

Now suppose that you in fact wanted to call the routine, not to jump to it as above. In this case the return address can be pushed onto the stack in advance. For example:

```
    LD HL, value of HL
    PUSH HL
    .
    .
    LD HL, return address
    EX (SP), HL
    PUSH HL
    LD HL, address of routine
    EX (SP), HL
    RET
```

This will call the routine and then return to the specified address. The original value of HL has been preserved and passed to the routine.

If the subroutine decides to, it can change its own return address, again using:

```
    EX (SP), HL
```

However, this is bad programming practice because it destroys the structure of CALLs.

---

It is bad practice to set the stack pointer except at the start of the program where it can be useful for re-initialisation. It is very bad practice to ever use the instruction DEC SP because this implies that you have data stored on the stack at an address less than the current stack pointer (I've used DEC SP to 'throw away' unnecessary data on the stack, I don't see anything wrong with that; Ed.). NAS-SYS 1 used this instruction, and is, I'm afraid, noninterruptable. By noninterruptable I mean that an interrupt at the wrong moment will crash the program. In fact close examination of the NASBUG monitors shows that the storing and restoration of registers is noninterruptable so they are no better. Will something be done to correct this small defect? Maybe!!

---

Do you know the correct name for a 'crash', meaning the loss of control by a program resulting in unpredictable changes to memory locations and usually requiring the reset button to be pressed? The correct name for a 'crash' caused by bad code is a 'program fault'. It sounds more impressive but it is just as annoying!

## ZEAP in EPROM

A number of people have sent in details of how to put ZEAP into EPROMs and copy it dowm to its correct location. Others have worked out how to put extra features into ZEAP, such as paginated output.

(continued on page 24)

# N2 *Documentation Errata*

Nascom Engineering have forwarded a list of errata for the Nascom 2 manual which they have asked us to publish: so here goes.

"Nascom 2 construction manual

page 5 para 7 The diagram 'component representation' shows the + at the wrong end of the diode symbol.

page 6 para 8 TR1 and TR2 are PNP and not NPN; TR4 is shown inverted in the fourth column of diagrams.

page 10 para B4 IC9 should be N2DB PROM; IC17 should be IC16

para C1 IC25 should be '81LS97 or 95/97'

para B5 delete 'causes'; insert 'may cause'.

page 11 para H1 TP4 and TP5 are inactive at this stage, though the stated signals are available from them during normal operation."

"Nascom 2 circuit diagram

1) The drawing number, 4-022-103, is to be deleted and replaced with 022-401 (sheet 1 of 5), 022-402 (sheet 2 of 5), 022-403 (sheet 4 of 5), 022-404 (sheet 4 of 5), and 022-405 (sheet 5 of 5)."

(Editors note: we think they mean 'sheet 3 of 5' for drawing 022-403; they didn't tell us where you get the drawings from.)

"2) (022-401) TR1 should be shown as a PNP common collector stage with R14 connected to the emitter and collector grounded.

3) (022-402) TR2 should be shown as a PNP common emitter stage, with R36 connected to the collector and the emitter connected to +12 volts.

4) (022-404) The capacitor C25 on pin 5 of the UHF modulator should be marked DC53.

Pin 15 of IC65 is connected to pin 2 ) of IC65 and not
Pin 10 of IC65 is connected to pin 11 ) as shown. Ed.

5) (022-401) IC9/12 is connected to IC5/11 and IC9/11 is connected to IC5/9."

"Nascom 2 Hardware manual

page 15 1st item in the test point list should read 'TP1: 300Hz from IC31/3'.

page 20, note 5 Switch positions are reversed.

page 20 C9 Is as stated and not a tantelum capacitor as mentioned in the construction notes.

C14 Is as listed and not a ceramic as stated elsewhere.

C19 Is a tantelum capacitor though this is not mentioned.

C29 ditto

DC45 & DC52 are as stated and not tantelum capacitors as stated elsewhere.

page 31 IC2 should be marked 74LS257 and not 74LS32.

page 35 IC58 Is shown as a 74LS163, should be 74LS123.

Note: Difficulty has been reported in identification of certain capacitors. Though it is not possible to guarantee that all manufacturers follow this convention, it is suggested that capacitors marked 102 be regarded as 1n and those marked 100 as 100p unless there are obvious signs to the contrary (size, or the possession of a positive lead).

Nascom 2 RAM board construction notes
page 5 It is reported that certain boards harbour an etching fault whereby the decoupling capacitor for IC8 fails to connect with the ground rail. It is reasonable to check for this fault and to correct it if possible, although its consequences are unlikely to be noticeable and are nothing to do with 'memory plague'.
page 12 et seq The memory test programs are written for the T2 monitor and will not operate under Nas-sys 1. It is recommended that the advice given on page 15 is followed and that a small test program is written to test for 'plague' only if this condition is suspected. The printed test programs are not specifically designed to reveal plague." (And probably won't. Ed.)

(Editors note: We do not like these unspecified references to 'memory plague' as unsuspecting purchasers of Nascom 2s would have no idea what is meant; only hardened Nascom 1 owners who read these newsletters will be in the know. For new Nascom 2 owners, 'memory plague' is a condition which arises in some memories due to noise and speed problems, and leads to unreliable memory performance (not to be confused with suspect chips). Several notes have been published about 'memory plague', and we hope to collect them all together and publish a compendium of memory mods in the next issue. Memory plague may be detected in a Nascom 2 by the failure of the Basic to initialize properly, or the 'free memory' message giving silly or inconsistent answers after initialization. This is not the same as the total failure of the Basic to initialize if the memory construction notes are followed to the letter (having missed the small errata); where the constructor is instructed to connect decode pad 12 to P5, thus enabling the RAM board EPROM block to page F, thereby totally disabling half of the Basic.)

"8K Basic programming manual
page 16 The list of reserved words should include DEEK and MONITOR.
page 24 The second sentence should read: 'It uses locations 1000H to 113EH (4414) ..... .' The second sentence should read: 'Locations 113FH to DFFFH are therefore ..... .'"

(Editors note: Well thats what it said, and we couldn't understand it either, so we looked it up in a manual that was a couple of months old. That didn't make sense either, suggesting that the Basic used 8K+ of workspace, so after a little investigation we think page 24 should read as follows:
"Nascom Basic leaves locations between 0C80H (3200 decimal) and 0FFFH (4095 decimal) inclusive for use by user machine code routines. It uses locations 1000H (4096) to 10F9H (4345) inclusive for workspace and resides in E000H (-8192) to FFFFH (-1) inclusive. Locations 10FAH (4346) to DFFFH (-8193) inclusive are therfore available for the users' Basic program and data.")

"page 26 program 1 line 40: the word 'to' should be in upper case."

For a manual the size of the Nascom 2 manual you must expect a few mistakes but we bet they haven't all been found (we haven't tried looking yet), but if you come across any please let us know so that they can be ammended in later manuals. That way new users of Nascom 2 might not run into some of the problems that have been experienced.

# S-100

## S-100 EXPANSION for NASCOM 1

This is a shortened version of an article sent in by Mr. K. P. Curtis of New Barnet. The INMC would very much like to hear from anyone else who has expanded their Nascom via the S100 bus.

I decided to expand my Nascom 1 with a S100 expansion kit currently being offered. Despite being more expensive in the short term, I felt that it widened the choice available to further enhance my system.

The kit consisted of :-

1) A double sided PCB without plated through holes that will accomodate three cards.
2) Two edge connectors and ribbon cable to connect the Nascom to the expansion board.
3) All required IC's and sockets except one, a tri-state buffer for control signals which is optional. (Would recommend this is used ED.)
4) Lots of vero pins to make the through board connections.
5) Decoupling capacitors.
6) A 2K Basic interpreter (B-Basic).

### Assembly

The instruction for assembling the board are adequate but the following points should be noted:

Insert all vero pins before soldering and very carefully check their positions as it is very easy to get them in the wrong positions.

Some pins need to be cut short to allow sockets to sit flush on the board.

The longest part of assembly was making up the ribbon cable assembly - very tedious.

### Documentation

A circuit diagram of the logic is provided but does not match the board layout and the logic that generates the MEMEXT signal, although functionally the same, is misleading. Also only the pin numbers of the buffers are given and some of the IC legends appear to have been incorrectly printed on the board.

Beside these errors I found two connections not made in the external I/O logic : IC5/15 to IC10/1 and IC9/11 to IC9/10.

### Power Supply

The ready built 8K RAM card that is being used requires an 8V unregulated supply from the S100 bus. This may be obtained from the existing power supply - after the rectifier but before the dropper resistors and regulator. Before attempting to use the kit and memory though, the rating of the rectifier should be checked - mine lasted for

all of five minutes. I now have an 8 AMP bridge which, although bigger, can be easily fitted with the aid of a piece of aluminium which also serves as a heat sink.

Since the RAM board draws about one and a half amps a healthy earth is required - a thick piece of wire from the power supply to the expansion board. However, this creates an earth loop, which perhaps is not desirable. (Suggest that all earths then taken from expansion board Ed.)

## Getting It Working

With the kit now assembled and 8K of RAM plugged in that should have been the end of the story, but there were the inevitable problems. The memory could be read from and written to , but very soon the machine would lock up. On power-up everything looked fine but the system would not initialise properly on pressing the reset key. Many hours were spent looking for non-existant faults. Eventually it became apparent that the MEMEXT signal was at fault but not why. With the internal memory select used there was no problem. With external memory select used the problem reappeared, but on monitoring MEMEXT with an AVO it disappeared again. In an attempt to cure the fault the chips at each end of the MEMEXT signal were changed but to no avail. Since the AVO cured the problem a resistor was soldered between the signal line and earth, this made a minor improvement. Next a capacitor was tried between the signal and earth at the NASCOM 1 end, this fixed it. Decoupling the expansion board end had no effect.

## Conclusion

This board is not easy to build and may require a small decoupling capacitor to be added to the main N1 board on the MEMEXT line. Having this signal so close to the earth and the +5v lines would appear not to be a good idea. A criticism of the design is that tri-state buffers are used unnecessarily on the control and address lines, surely ordinary buffers would have done the job. Since getting this expansion working I have experienced no other problems and am very happy with it.

Mr. Curtiss neglected to mention the manufacturers of this product, but we believe it is the one supplied by Comp Components of Barnet.

There were a couple of other points in the letter attached to Mr. Curtiss' article which ought to be mentioned. One of the reasons he gave for choosing the S-100 mother board was that it allowed greater choice of 'goodies' that may be fitted. Whilst generally true, this particular board is not truly S-100 compatible, and may give rise to problems under special circumstances. Another reason was that in his opinion the Nas-bus was responsible for some of the trouble experienced with Nascom expansion. In our opinion the trouble is not concerned with the Nas-bus but with the series 1 memory card. Happily these troubles can all be cured. Finally Mr. Curtiss asks why a non ASCII keyboard was chosen for Nascom and says, "Surely it would only cost a few pounds

(continued on page 18)

# DOCTOR DARK'S DIARY-4

(An every day story of simple programming folk)

Issue five of the INMC News arrived at Zilog Villas just in time for Christmas - this was clearly the result of a prodigious effort by both the committee and the post office. Having suggested to our illustrious editor that he should print both the episodes I had sent in at once if there was a shortage of material, I was caught with my pencil down. The pressure is on now, no more playing with the piranhas until I have typed this episode out.

## MAKING M5 RUN BIGGER PROGRAMS

If you have run Microdigital's M5 interpreter, you will know that it can only run programs of about 230 bytes or less, depending on how much use is made of the stack. On an expanded Nascom, only a few changes need to be made to the M5 interpreter to move the program store to the additional memory. The changes are as follows:

```
0DE1  :  00  10
0E06  :  F0  0E
0E2F  :  FF  0F
0E5A  :  FF  0F
0EB3  :  00  10
0EDB  :  FF  0F
```

As well as changing the program store location, I have also changed the variable stores from the top line of the screen, which was rather unsightly, and put them where the program used to be. There is still plenty of space for the stack to work in, it would take about a hundred consecutive pushes to cause problems - if your calculation is that complex, you should be using Fortran! Should you happen to write a program in M5 that uses 32K of RAM, you may find jumps from the low end of memory to the high end are fairly slow. Serve you right for writing such long programs.

## TINY DISASSEMBLER

When I first used M5, I wanted to find out how it worked, so I decided I would disassemble it. Hand disassembly is incredibly tedious, and Personal Computer World still have not printed the end of Parkinson's Revas (a Nascom disassembler, which it just happens you can buy if you have the money). Half way between the two alternatives is a program written by R. M. Lucas, and published in Computing Today, October 1979. This program is called Intab (short for Intelligent Tabulate) and breaks a program up into its separate instructions - much easier to understand than a rectangular block of code. As Intab is probably protected by all sorts of copyright laws, I have taken the diabolical liberty of writing an improved version, which is fully relocatable, and (sorry!) only works with T4.

```
ED 5B 0E 0C CD 3E 00 FE 2E 20 01 CF FE 20 20 F4
01 05 00 EF 1F 00 EB CD 32 02 EB EF 3E 20 00 1A
FE EF 20 0A D7 68 A7 28 2B F7 13 1A 18 F8 FE DD
28 28 FE FD 28 24 FE ED 28 42 0E 1A 2A 3B 0C D5
11 9B 00 19 D1 ED B1 06 03 28 0B 01 19 00 ED B1
06 02 28 02 06 01 D7 38 18 AA D7 32 2A 3B 0C D5
11 B2 00 19 D1 ED B1 28 DE 2A 3B 0C D5 11 D4 00
19 D1 0E 0B ED B1 28 DC 06 02 18 DA D7 10 0C 2A
3B 0C D5 11 CE 00 19 D1 ED B1 28 BB 18 C6 06 01
CD 44 02 CD 3C 02 13 1A 10 F6 C9 01 11 31 32 3A
C2 C3 C4 CA CC CD D2 D4 DA DC E2 E4 EA EC F2 F4
FA FC 21 22 2A 36 CB 06 0E 10 16 18 1E 20 26 28
2E 30 38 3E C6 CE D3 D6 DB DE E6 EE F6 FE 43 4B
53 5B 73 7B 09 19 23 29 2B 39 E1 E3 E5 E9 F9    (Phew!)
```

How to use it

1. Load the above into a section of RAM not used by the program you
2. Load the program you want to dissect. wish to examine.
3. Use the command E xxxx yyyy NL , where xxxx is the address of the first byte of the above, and yyyy is the address at which disassembly is to start.
4. Press the space bar to produce each line of code in turn, and full stop to return to the monitor.

The program does not understand about data tables, and the fact that RST 16 (D7) is followed by a displacement when using T4.

WHAT HAVE I DONE WITH IT SO FAR?

Having found out how M5 functions, I decided to write a Pilot interpreter, as threatened in episode 3. In a trice (about three weeks) it was finished. It didn't work. Not only that, it was so complex, and my documentation so poor, that I could easily have spent another three weeks trying to find the errors.

MORALS a) Test each section as it is written
b) Make detailed notes, you will forget how it works, and nobody else knows!

In a flash (another ten days) I took the easy way out, and re-wrote the whole thing, testing each section as it was completed. When it was finished, it worked first time, some of the time. The dreaded memory plague had struck - fortunately, it was in its mildest form, and was easily dispelled with a National chip in the recommended place. This is interesting, as the memory test programs supplied with the board did not reveal any problems, obviously there is an application for the interpreter as a test program here.....

Anyway, I now have a 2K Pilot interpreter running, and have even written a few programs in Pilot, the easiest language I have ever seen. I'm now half way through version 3, and it looks even better. I will keep you informed of its progress.

Doctor Dark and Marvin wish a happy and prosperous 1980 to all intelligent life-forms in the Galaxy (and to the rest of you, the secret is to bang the rocks together....)

SHOCK HEADLINE

AUTHOR FINDS OWN BOOBS IN INMC ARTICLE by Ms D. R. Hunt

No, not a refugee from page 3 (we got told off by readers last time we let a naughty word onto these hallowed pages), but to draw your attention to the fact that we all make mistakes. Last issue, we published a bit about adapting the series 1 memory card to carry the 8K Basic ROM, and went on to describe how to put the remaining three EPROM sockets back into use. Well the Basic ROM bit works, but the EPROM bit doesn't!!!

Having fitted the Basic ROM to the card, and written it up for the INMC, in a most un-natural fit of enthusiasm, I modified the board for the EPROMS, 'zapped' in a couple of EPROMS, gave it a quick go and wrote it up, but because of the closeness of the copy date, I didn't get round to actually testing it thoroughly. Later, (over Christmas), horror of horrors, fit some EPROMS and the Basic does funny things. Close examination of the circuit reveals that enable of IC24 (pin 1) comes on when the Basic ROM comes on, so with EPROMS fitted they are enabled along with the Basic ROM resulting in rubbish on the bus. The strange thing is though, that in the conflict between the EPROMs and the ROM, the ROM usually won, so Basic didn't just crash as you'd expect, but became unreliable.

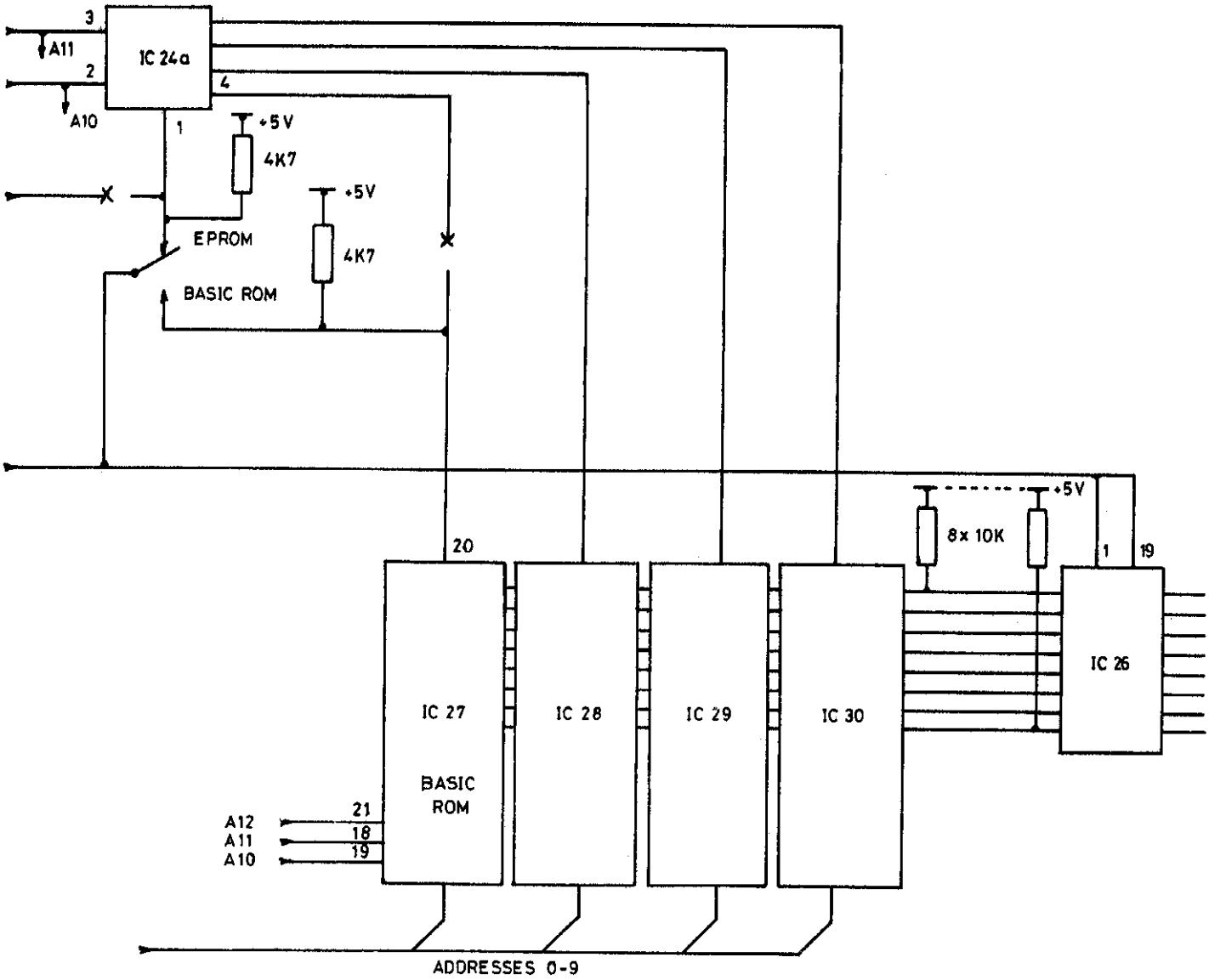
At first sight this problem looks very difficult, requiring a three pole switch (very hard to find) to switch the IC24 enable along with the Basic enable, and the decodes. However, look at the circuit again, it won't take long to realize that the enable line to IC26 (which we already use for the Basic ROM) could just as well be used to enable IC24 instead of the signal currently used.

So, now to the correct version, refer back to the previous article, and notice that a 4K7 pull up resistor (on the right hand connector) is used to disable the Basic ROM when not in use, leave this where it is, but reverse the connections to the other side of the switch. Viz: IC26 pin 19 is connected to the pole of the switch, and IC27 pin 20 is connected to the left hand connector. Further, another 4K7 pull up resistor has to be fitted to IC27 pin 20 to pull the Basic ROM 'off' when the switch is over the other way. Carefully cut the track to IC24 pin 1 (adjacent to the pin), and connect pin 1 to the right hand connector of the switch, leaving the existing 4K7 pull up resistor in place to turn IC24 'off' when the switch is over to the Basic.

All this seems to work satifactorily, but we have heard of a few cases of difficulty (even without EPROMs on board), a 'Stop Press' was added to the previous article about fitting a 100n capacitor across the back of the ROM to decouple the supply lines, I have found that 2u2 bead tant. is better. You wouldn't believe the 'crud' the Basic ROM generates on the supply line as it's enabled, if you've got a 50MHz scope to hand, have a look at it across pins 12 and 24. A further aid to quieting down and speeding up the system is to decouple each EPROM socket with 10n, and to pull the data outputs of the ROM and EPROM block high with 10K resistors.

We've mentioned memory plague before, and a recent note in the INMC newsletter gives a more consistent cure. However, whilst looking for the troubles with the Basic ROM with EPROMs in situ, a remarkably naughty bit of tracking on the earth line of the EPROM block was discovered. The earth track is connected to the ground plane close to pin 12 of IC27, but nowhere else!!! So I now suggest when gridding the back of the pcb that pin 12 of ICs 28, 29 and 30 be incorporated in the grid thereby incorporating the EPROM sockets firmly into the ground plane. It can't do any harm, and seems to do some good (but difficult to prove).

By this time the back of the memory board is beginning to look like the classic 'rats nest' with bits of wire and resistors hooked all over it. Don't worry, finding the cures, and incorporating the mods was fun wasn't it? And after all, the thing works reliably (or should anyway).



# Memory maps

## Software of the future

What is the perfect memory layout for the future? What software should be resident in ROM and what should be loaded from tape? You might like to think about these questions in regard to some of the software that already exists or is rumoured to be around.

Nas-sys 1 — This must be in ROM, and naturally lives at 0000H, although it is not totally impossible to put it elsewhere with Nascom 2.

Basic — Both Rom and tape versions exist. Nascom 2 has it at E000 - FFFFH.

ZEAP — This assembler is a candidate for ROM as it is so useful. ZEAP 2.0 can be in ROM at D000H or RAM at 1000H.

REVAS — This disassembler can be located anywhere. Is it used often enough to make it worth having in ROM? It could occupy C400 - CFFFH

All this brings us to memory mapping. So much is going on by independent people, hoping to sell products under the Nascom banner, that a meeting was held recently between Nascom Engineering, and other interested parties. The aim; to ensure that there is room for everything, and that everything can be kept under control. The meeting primarily discussed a standard Nascom 2 expanded to a maximum of 48K and fitted with Nas-sys. Non standard Nascoms and disc based Nascoms were not discussed as their memory requirements will be entirely different. But everything decided is equally relevant to an expanded Nascom 1 fitted with Nas-sys.

One of the mains aims was agreed, and that is that no external product can advertise that it is 'approved by Nascom' unless it has been submitted to Nascom for approval, and that it conforms to the memory mapping laid down. Another aim was to produce a reccommended memory map for use with Nascom 2, but is was soon discovered that so many diverse ideas were around that it was only possible to set out guidelines, and that firmware producers would have to look at their product to see where it could be sensibly fitted. This is in the producer's interest, because if something is located in a daft place then it inhibits its own market.

```
FFFF    8K Basic
E000
====
DFFF    ZEAP 2.0          or general assembler type software
D000                      or extensions to Basic.
====
CFFF    REVAS             or general disassembler software
C000                      or Colour graphics control software.
====
BFFF    Naspen            or related word processing software.
B800
```

```
B7FF    Extensions to the operating system
B000                    or extensions to Naspen.
====
AFFF    Colour graphics RAM or general RAM space.
9800
====
97FF    Programmable grahics RAM or general RAM space.
9000
====
8FFF    General RAM space.
1000    Start of general program space.
====
0FFF    Usual stack space.
0F80
====
0F7F    Workspace for firmware
0D00            or workspace for programs.
====
0CFF    Extended operating system workspace.
0C80
====
0C7F    Existing operating system workspace.
0C00
====
0BFF    Existing video RAM.
0800
====
07FF    Existing operating system.
0000
```

As you will notice, firmware which is unlikely to be used together has been assigned coresidence. For example, no-one is likely to want to use the colour graphics firmware (which runs under Basic) with REVAS (which is a machine code disassembler). Similarly, any extensions to Naspen may be made from B000 - B7FF, as with a more powerful word processor in residence, extensions to the operating system would hardly be required, and so on. However, extensions to the Basic could imply the use of a more powerful operating system, or a software number cruncher such as MAPP1-3Z could be put in place of ZEAP, and used to extend the precision of the Basic math to 14 digits (if any one has done this please let us know).

So please bear these suggestions in mind in writing your software, that way maximum portability, and least inconvenience is caused.

---

S-100 (continued from page 12)

more for a real one". Well we don't know, but in general the Nascom keyboard costs rather more (not less) than the average ASCII keyboard fitted to competing computers, and is a very rugged and superior product. We don't know why all the control is done in software, as it would have cost little more to make the keyboard truly ASCII. Mind you, a software controlled keyboard lets you do lots of clever things which a hardware keyboard could never do (look at the N Ray Moonlander program for instance). Finally, does it really matter, as ASCII keyboards can easily be run into PORT 0 with very little modification to the software if required.

---

# PIOs & RRRepeat KBDs

A few newsletters back we did an article about the PIO (MK3881), and tried to get readers out of the 'Catch 22' situation, where the PIO technical manual could only be understood if the reader knew something about PIO's. The article dealt with the general aspects of the PIO, and how to use it. Well Richard has been busy, and has sent us this:

## Understanding the PIO

The PIO manual provides a good description of how this powerful device works, but there are some aspects of it which are not easy to understand. Perhaps the most confusing of all is the use of the 'handshake' signals, known as STROBE and READY lines. This article will briefly explain how input and output modes work, and then will give an example showing how two PIO's can communicate, when one is the output and the other is the input.

### Input mode

The PIO port must be set to mode 1 and interrupts enabled. As usual the I register, interrupt vector and interrupt mode must be set up correctly, and the PIO must not think that it is already processing an interrupt, so a dummy RETI instruction should be executed. Also, when in input mode, a dummy read should be issued once at the start to initialize handshaking. At last you are ready to enable the CPU interrupts with an EI instruction.

The sequence of events is as follows:

The input STROBE line goes from low to high, indicating that data is ready in the PIO input register. This change causes an interrupt, and the interrupt routine gets the data with an IN instruction. Issuing this instruction gets the data and also causes the PIO READY output line to go from low to high, indicating that the data has been received. The interrupt routine ends by enabling the CPU interrupts again and then immediately returns with a RETI, which re-enables the PIO interrupts.

### Output mode

The PIO port must be prepared as for input mode, except that it is set to mode 0.

The sequence of events is as follows:

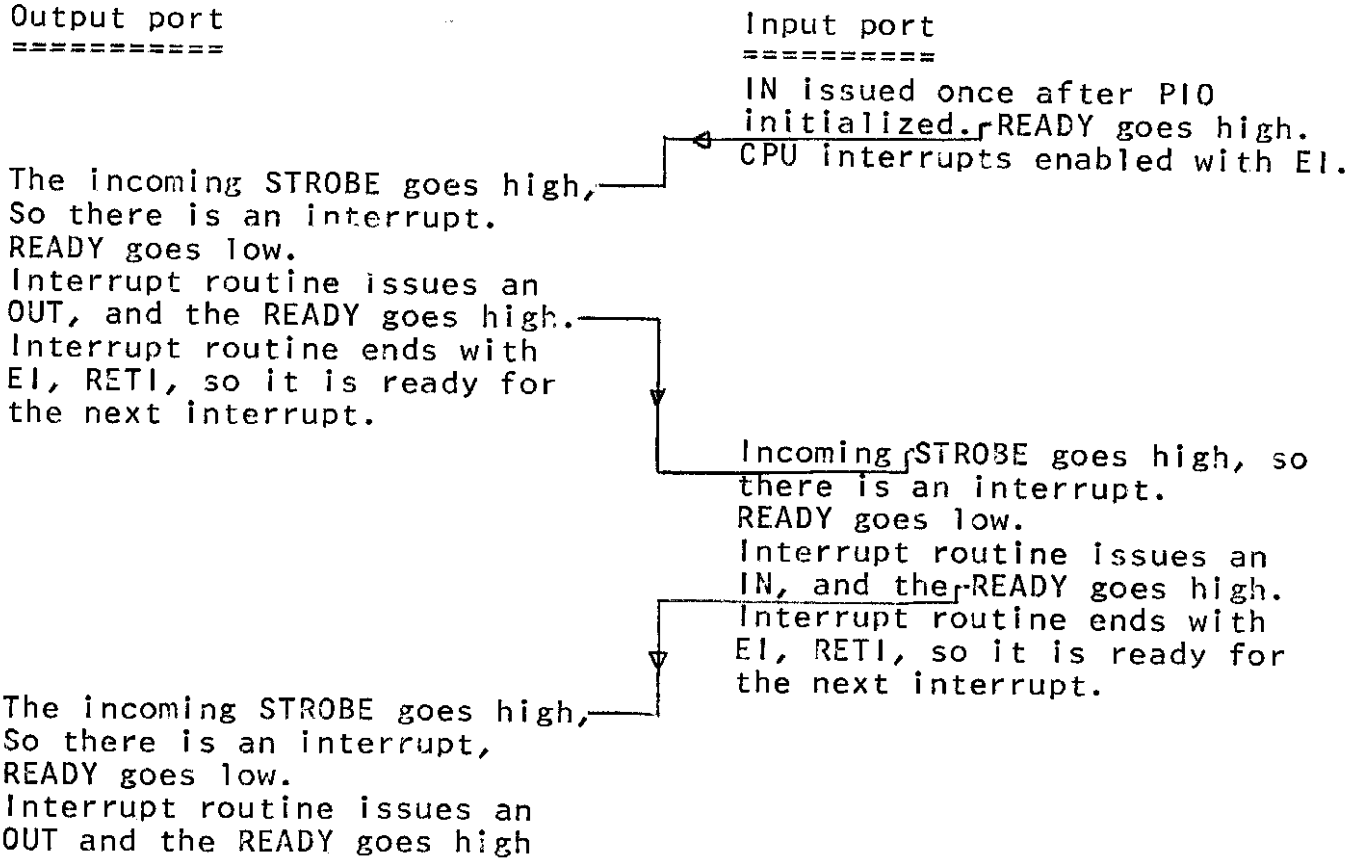
The input STROBE line goes from low to high, indicating that the previous output has been received. This change causes an interrupt, and the interrupt routine outputs the data with an OUT instruction. Issuing this instruction not only outputs the data to the PIO, but also causes the PIO READY line to go high, indicating that data is ready in the PIO. The interrupt

routine ends by enabling the CPU interrupts again and then immediately returns with a RETI, which re-enables the PIO interrupts.

You may have wondered when the READY line goes low. In both modes the READY line goes low when an interrupt is generated. This may be used by the external device to acknowledge that an interrupt has been received. It could be used to inhibit further interrupts from the external device, particularly where a number of PIO's are 'daisy chained' together, and higher priority interrupts are already being processed, hence there will be a delay before the external device is processed. In other words, it says to the external device "OK, seen you, you are in the queue, don't bother me again until I go high.". It has no internal affect on the PIO, save to get it ready to go high again when processing of the interrupt is complete.

Example of handshaking sequence
==========

For two PIO's to handshake, the STROBE (input) line of each is connected to the READY (output) line of the other. This diagram shows a typical sequence of events, starting with a dummy read from the input port to start the process.




You could try this with just one PIO, by making the two ports of your PIO talk to each other.

A note for Nascom 2 owners. The memory card joins the 'daisy chain' (IEI and IEO) lines together, and under these conditions the interrupts won't work. Cut one of the tracks on the vero buss connector. Do not cut the link on the memory board, as this would be required if that card were used on a bigger buss with 'daisy chain' priority interrupts.

Its beginning to look as if this newsletter was written by Richard, because this screed also came from Beal mansions.

## Repeat keyboard for Nas-sys

by Richard Beal

In the last issue we printed a repeat keyboard routine for Nasbug T4. This can be adapted to Nas-sys, but we have now developed a routine which is simpler. This routine has given us an opportunity to show you the output of ZEAP 2.0, notice the sorted symbol table; there is a listing of the routine and a program to put the routine into use. Execute the program at 0C90H.

The routine itself is from 0C80 to 0CF5H. After the program is executed it returns to Nas-sys. You can then use Nas-sys or execute ZEAP 2.0 or Basic and enjoy the benefits of a repeating keyboard. Just hold down any key, and after a delay it will repeat quite quickly, which is ideal for cursor movements. Try holding down several keys at once!! Note that the @ key does not repeat because of its dual function on Nascom 1. We also attach a tabulation of the code to make it easier to type in.

## Memory usage and changing the speed

We hereby declare the following memory locations will for all time have the specified meanings:-

```
0C2C - 0C2DH    KCNT counter for repeat keyboard
0C2E - 0C2FH    KLONG delay before repeating starts
0C30 - 0C31H    KSHORT speed of repeating keys
```

These were unused locations in the Nas-sys work area. To change the initial delay before repeating starts, and the speed of repeat, change the values at KLONG and KSHORT.

| Suggested values: | 4MHz | 2MHz |
|---|---|---|
| KLONG | 0280H | 0140H |
| KSHORT | 0050H | 0028H |

The version overleaf is for 4MHz (Nascom 2).

## Repeating keyboard from Basic

You can also set up a repeating keyboard within Basic, by typing in the Basic subroutine at lines 24000 to 24090. To execute it, type in: RUN24000
You will find that you then have a repeating keyboard. Note that line 24030 sets the speeds, we suggest the following values:

| | 4MHz | 2MHz |
|---|---|---|
| Initial delay | DOKE 3118,640 | DOKE 3118,320 |
| Repeat rate | DOKE 3120,80 | DOKE 3120,40 |

```
ZEAP Z80 Assembler - Source Listing

                0010 ; REPEAT KEYBOARD ROUTINE
                0020 ; WRITTEN BY RICHARD BEAL

0000            0040        ORG  0
0000 0061       0050 ZKBD   EQU  #61
0000 005B       0060 ZMRET  EQU  #5B
0000 0072       0070 ZNIM   EQU  #72
0000 0070       0080 ZSRLIN EQU  #70
0000 0076       0090 ZUIN   EQU  #76
0000 0C01       0100 KMAP   EQU  #0C01
0000 0C2C       0110 KCNT   EQU  #0C2C
0000 0C2E       0120 KLONG  EQU  #0C2E
0000 0C30       0130 KSHORT EQU  #0C30
0000 0C7A       0140 $UIN   EQU  #0C7A
0000 0280       0150 LONG   EQU  #0280
0000 0050       0160 SHORT  EQU  #0050

                0180 ; SET UP ROUTINE
0C90            0190 RKS    ORG  #0C90
0C90 21F30C     0200        LD   HL,RKT
0C93 DF72       0210        SCAL ZNIM
0C95 21B00C     0220        LD   HL,RK
0C98 227B0C     0230        LD   ($UIN+1),HL
0C9B 218002     0240        LD   HL,LONG
0C9E 222E0C     0250        LD   (KLONG),HL
0CA1 215000     0260        LD   HL,SHORT
0CA4 22300C     0270        LD   (KSHORT),HL
0CA7 DF5B       0280        SCAL ZMRET

                0300 ; REPEAT KEYBOARD ROUTINE
0CB0            0310 RK     ORG  #0CB0
0CB0 DF61       0320        SCAL ZKBD
0CB2 3007       0330        JR   NC,RK2
                0340 ; KEY PRESSED
0CB4 2A2E0C     0350        LD   HL,(KLONG)
0CB7 222C0C     0360        LD   (KCNT),HL
0CBA C9         0370        RET
                0380 ; KEY NOT PRESSED
0CBB 2A2C0C     0390 RK2    LD   HL,(KCNT)
0CBE 2B         0400        DEC  HL
0CBF 222C0C     0410        LD   (KCNT),HL
0CC2 7C         0420        LD   A,H
0CC3 B5         0430        OR   L
0CC4 C0         0440        RET  NZ
                0450 ; TEST AND CLEAR TABLE
0CC5 21020C     0460        LD   HL,KMAP+1
0CC8 010008     0470        LD   BC,#0800
                0480 ; SET UP MASK IN D
0CCB 16FF       0490 RK3    LD   D,#FF
0CCD 7D         0500        LD   A,L
0CCE FE06       0510        CP   6
0CD0 2002       0520        JR   NZ,RK5
0CD2 16BF       0530        LD   D,#BF
0CD4 FE09       0540 RK5    CP   9
0CD6 2002       0550        JR   NZ,RK6
0CD8 16C7       0560        LD   D,#C7
                0570 ; TEST FOR KEY DOWN
0CDA 7E         0580 RK6    LD   A,(HL)
0CDB A2         0590        AND  D
0CDC 2806       0600        JR   Z,RK7
0CDE 0E01       0610        LD   C,1
                0620 ; CLEAR KEYS DOWN
0CE0 7A         0630        LD   A,D
0CE1 2F         0640        CPL
0CE2 A6         0650        AND  (HL)
0CE3 77         0660        LD   (HL),A
                0670 ; NEXT ROW
0CE4 23         0680 RK7    INC  HL
0CE5 10E4       0690        DJNZ RK3
0CE7 79         0700        LD   A,C
0CE8 B7         0710        OR   A
0CE9 C8         0720        RET  Z
                0730 ; REPEAT SPEED
0CEA 2A300C     0740        LD   HL,(KSHORT)
0CED 222C0C     0750        LD   (KCNT),HL
                0760 ; SCAN KEYBOARD
0CF0 DF61       0770        SCAL ZKBD
0CF2 C9         0780        RET

                0800 ; INPUT TABLE
0CF3 76         0810 RKT    DEFB ZUIN
0CF4 7000       0820        DEFB ZSRLIN,0

                0840 ; END OF LISTING

ZEAP Z80 Assembler - Symbol Table

0C7AH 0140 $UIN          0C2CH 0110 KCNT
0C2EH 0120 KLONG         0C01H 0100 KMAP
0C30H 0130 KSHORT        0280H 0150 LONG
0CB0H 0310 RK            0CBBH 0390 RK2
0CCBH 0490 RK3           0CD4H 0540 RK5
0CDAH 0580 RK6           0CE4H 0680 RK7
0C90H 0190 RKS           0CF3H 0810 RKT
0050H 0160 SHORT         0061H 0050 ZKBD
005BH 0060 ZMRET         0072H 0070 ZNIM
0070H 0080 ZSRLIN        0076H 0090 ZUIN

     TC90 D00 0 0
       0C90 21 F3 0C DF 72 21 B0 0C
       0C98 22 7B 0C 21 80 02 22 2E
       0CA0 0C 21 50 00 22 30 0C DF
       0CA8 5B 00 00 00 00 00 00 00
       0CB0 DF 61 30 07 2A 2E 0C 22
       0CB8 2C 0C C9 2A 2C 0C 2B 22
       0CC0 2C 0C 7C B5 C0 21 02 0C
       0CC8 01 00 08 16 FF 7D FE 06
       0CD0 20 02 16 BF FE 09 20 02
       0CD8 16 C7 7E A2 28 06 0E 01
       0CE0 7A 2F A6 77 23 10 E4 79
       0CE8 B7 C8 2A 30 0C 22 2C 0C
       0CF0 DF 61 C9 76 70 00 00 00
       0CF8 00 00 00 00 00 00 00 00
```

```
24000 REM ** SET UP REPEAT KEYBOARD
24010 FORI=3248TO3316STEP2:READJ:DOKEI,J:NEXT
24020 DOKE3195,3248:DOKE3189,3315
24030 DOKE3118,640:DOKE3120,00:END
24040 DATA25055,1840,11818,8716,3116,10953
24050 DATA3116,8747,3116,-19076,8640,3074,1
24060 DATA5640,32255,1790,544,-16618,2558,544
24070 DATA-14570,-23938,1576,270,12154,30630
24080 DATA4131,31204,-14153,12330,8716,3116
24090 DATA25055,30409,112
```

# Program hints

By N.Ray

## ONE OR TWO NOTES ABOUT Z80 MACHINE-CODE PROGRAMMING

Reading other people's Z80 code programs convinces me that many people are unaware of some very useful tricks.

For instance, although it is well-known that XOR A may be used to clear A (and the carry bit) the other operations on A have their uses.

E.g. AND A and OR A set Z and S flags from contents of A and may frequently replace CP Ø.

AND A (or OR A) can also be used to clear the carry flag if you don't mind corrupting the other flags. CP A will clear the Z flag.

A warning of general application is in order here.

AND A is not the same as CP Ø
INC A is not the same as ADD A,1 (Sim. DEC)
CPL,INCA is not the same as NEG

In each case the code on the left differs from that on the right in the way it affects the carry flag (and certain other flags).

While on the subject of programming dodges, here is an interesting example:

```
E6ØF    HEXASC:   AND ØFH
C69Ø              ADD A,9ØH
27                DAA
CE4Ø              ADC A,4ØH
27                DAA
C9                RET
```

The lower 4 bits in register A are interpreted as a hex digit and the corresponding hex character in ASCll is found in A on exit. You should try to convince yourself that it works correctly (HINT : watch the half-carry flag).

## RECURSION

Recursion can often be used to good advantage in machine-code programming, provided there is ample stack space. A recursive routine is one that calls itself. As a demonstration, consider the 'Tower of Hanoi' problem.

N rings are stacked on one of three poles (as indicated)

The problem is to move the rings from X to Z (using Y) by transferring one ring at a time from one pole to another, such that a larger ring is never placed on top of a smaller ring.

A recursive solution is:-

```
LET   Move N from A to C using B
BE    $(  IF  N=0  RETURN
          Move N-1 from A to B using C
          Take top ring from A to C
          Move N-1 from B to C using A   $)
```

The solution is then a call to 'Move N from X to Z using Y'. The solution is easily seen to be intuitively correct - 'Take' represents the action of moving a single ring. We can easily see that for N rings, the routine nests to a depth N, and Take is called $2^N$ times.

I have sent a program for inclusion in the INMC library that runs on a minimal NASCOM and neatly demonstrates the correctness of the routine. It moves one ring for each press of any key.

In the program BC, DE, HL points at the 'poles' and N is stored in A.

It is arranged that BC, DE, HL always point at the top ring of a pole.

This program is also an excellent demonstration of the power of the Z80. I am able to keep all the variables in registers, thus saving on both execution time and storage space.

---

SOFT SPOTS (continued from page 8)

However, we hear that ZEAP 2.0 has been produced by the producers of ZEAP 1, and that it is a great improvement over the original. It has many useful features such as sorted symbol table. Also it will be available in either RAM (at 1000H) or ROM (at D000H) versions. It obviously makes better use of memory to run ZEAP in EPROM. It runs in 4K.

Zeap 2.0 runs only with Nas-sys, since it was designed for Nascom 2 owners, but it will of course run on a Nascom 1 fitted with Nas-sys. A version called ZEAP 1.2 has similar features and runs under Nas-bug.

Details such as price, and trade in offers for owners of ZEAP 1 will be available soon, so please don't write to us.

---

Just a simple little test (no prizes). What command can you enter to make Nas-sys un-resettable?

---

# "us"

We've had a couple of letters recently saying that the newsletters are very nice and all that, but how can we be objective when we all work for Nascom. Well this has shown us that we have never bothered to tell you more than the sketchiest details of who we are, or for that matter, how we 'run' the INMC.

Can we make it absolutely clear that none of us, except Kerr Borland, work for Nascom. For the sake of those who have joined the INMC over the last year, we will tell you how this came about, and a bit more about ourselves. The Committee is a group of individuals who were, for various reasons, known to Nascom from before, or shortly after, the Nascom 1 launch. One night early last summer we met at the invitation of Kerr, and the idea of an 'ad hoc' INMC committee, to organize a newsletter, was tossed around. As the evening progressed, over certain liquid refreshments, a degree of amicable agreement was reached; to the extent that we ended up having to phone each other the following morning to find out what we had agreed to. At that time we were Richard Beal, Howard Birkett, Paul Greenhalgh, and David Hunt. Paul worked for Nascom, and we thought he would be good liason between us, and the product we were to support (may be we thought he would be a good source of inside information as well, but this did not prove to be the case). So over a period of time the first couple of newsletters saw the light of day, and our appeals for contributions started to be answered. Paul left Nascom, and strangely, Howard joined them. As Howard considered that he had too much of a vested interest in Nascom, he resigned from the INMC committee, to be replaced by two volunteers who had come to light amongst the membership; Derek Brough, and Eddie Pounce.

As far as the committee goes, our main resposibilty is to produce a newsletter once every couple of months (sometime perhaps we will reach our schedule), to keep an eye on the INMC library, and to act as a filter for, and if possible, answer any corespondence which can't be dealt with by dealers or Nascom engineering. In return for this, and our general support of the product, Nascom provide us with a letter box at Chesham, and secretarial services to handle distribution of programs from the library, and the newsletter mailing list.

We are independant of Nascom in most repects, although we appreciate the help they give us. The only stipulation (or better, restraint) imposed by Nascom is that we must not print derogatory comment on any other product which directly competes with Nascom, which makes sound commercial sense. We hope that we are objective in our aims, if Nascom ever 'lay an egg' over some product or other, we hope we will be the first to say so.

Just so you know a bit more about the INMC commitee, we've each produced a 'potted' history. For professional reasons we will not name the Committees members' employers. So here goes:

Richard Beal has been with the commitee from the start. He is 29, married and lives near Richmond Surrey. He has worked with computers for about 8 years, and until recently was a systems analyst. He is is now working for a large international management consultantcy. He owns a 16K Nascom 1 (which must qualify as being one of the first to come from Chesham), a 32K Nascom 2 running ZEAP, Naspen, and 8K Basic.

He also has an EPROM programmer, a Teletype 43 Terminal and an acoustic coupler. He is the designer of B-Bug, Nasbug and Nas-sys, which were all designed for his own purposes, and later made available to Nascom. His resposibility on the INMC is software problems, and software related corespondence.

Derek Brough joined the committee in October, is 36, is married with one child and lives in Barnet. He has a B.Sc, and is a lecturer in computing sciences at a London college. He has played with computers since school. He has an unexpanded Nascom 1 fitted with a number of hardware goodies, including joystick controls, and an interrupt driven IBM printer. He also has access to expanded Nascoms through his work (but says he usually can't get near them). He has recently undertaken the incredible task of re-assembling the complete INMC library so that the bulk of it may be used with Nas-sys. His main responsibility with the INMC is the library, and he is the nearest to Librarian we have got. He is also re-organizing the library to make sense of the various breeds of programs now coming in; ie: minimum system Nasbug, expanded system Nasbug, expanded system Nas-sys, Tiny Basic and 8K Basic.

Kerr Borland is 35 and he says, "I was born to be a rich man's son, unfortunately, my father didn't work hard enough". He also says, "I am qualified for absolutely nothing, and spends most of my time doing what I am best qualified for!". He has also been known to say that he is a most likeable fellow, and we have noticed that he often tells us what a modest person he is. Kerr is Sales Director of Nascom, is married with one or more children (we are not sure whether it's us who don't know, or Kerr; Ed.), and lives in Totteridge. As president of the INMC he is our 'inside contact' with Nascom.

Paul Greenhalgh hails from the the north Country, is single, 24, and lives in Chesham. He has worked for an Electricity Board, and an electronic security firm and until recently he worked for Nascom. He is now employing his talents with a components supply company and Nascom distributor. He doesn't have a Nascom, but since leaving Nascom is saving his pennies to get one. He has tinkered with computers since leaving university, where he graduated with a B.Sc. On the INMC he is responsible for getting all the bits for the newsletter together, final editorial, pasteup, getting the newsletter printed, and distibution.

David Hunt is 34, lives in Harrow, has three children, and is still married despite his Nascom. He was trained as an electronics development engineer, leaving college with an HNC. After leaving the electronics field he went into retailing, starting in Hi-fi and later with a retail electronics company (not the same one as Paul), where he has been General Manager for the past five years. He has only been playing with computers for about 3 years, and was in the process of designing a kit for his company to market when Nascom 1 came along. He promptly dropped his own design, and is now a Nascom distributor. He owns a 32K Nascom 1 running Nas-sys and Nasbug, ZEAP, Naspen and 8K Basic. All firmware is kept on an 8K EPROM board, and his Nascom is fitted with programable graphics and high speed CUTS interfaces, he also has a TI 745 terminal. As INMC chairman, he keeps an eye on the hardware side of things, and acts as editor before passing copy on to Paul.

Eddie Pounce joined the committee in October, is 27, is married with two children and lives in Watford. He has been playing with computers since leaving school, and throughout his university training, where he gained a B.Sc Hons. in computing. He is a data administrator for a large multi-national company working in London. He owns a Nascom 2 with Naspen, and fitted with an automatic double cassette I/O with auto. stop/start. He is equally at home with machine level code, Basic, and the hardware side of things, and has a general background in both mainframe, mini and micro computing. On the INMC he acts as sub-editor, and deals with technical corespondence.

So now you know who we are; as to the INMC itself, we meet about once every two months to chew over recent developments, and get feedback from each other. Corespondence is usually farmed out between members by post, and the letters are dealt with by the individuals, poor Paul seems to 'cop' most of them. Newsletter bits are usually rewritten (we call it editing) by Richard, Paul, Eddie and Dave, in most cases on Naspen tapes. Drafts are printed usually on the TI printer, and Paul and Dave have a late night session putting it all together to find out how much we've got. We usually aim for about 30 pages. If we are short of copy, some programs are rooted out and printed to make up the space.

Final editing is carried out on the Naspen tapes and printed on an IBM printer on A4 sheets. Paul then takes the lot, pastes it all up, filling spare bits as he goes. At this late stage Chas. is chivvied up to come up with another 'Lawrence' and any other line drawings required, and the whole lot sent to the printers. When it is all printed, the girls at Nascom get landed with seeing that they all get posted.

So thats who we are and what it is all about. The whole process usually takes about two months, and a lot of copy for the next newsletter is on file even before the previous one is 'put to bed'. So when you send us letters or copy, don't despair if it doesn't turn up in the next newletter, we're horrors for holding things over to the next issue. But the final message is we still need copy, keep it coming. We want reviews on MAPP1-3Z and the cheap colour graphics which is knocking around. We also want a review on any EPROM programmers. Please.

---

## NASCOM 1 VDU PROBLEM - IC 18 MODIFICATION

Mr. L. E. Gilham of Bedfont asked us to clarify the position with regard to the modification to IC 18 to cure the problem with the VDU when a space is displayed where a letter should be.

The problem is around the timing of the character decoding in IC 16 and is corrected by:

1) Bending Pin 5 of IC 18 straight so that it is not connected when IC 18 is in the socket.

2) Connecting Pin 5 to Pin 12 of IC 18 on the underside of the board.

---

## SOFTWARE LIBRARY

Below is the current list of Programs in the software library. No NAS-SYS machine code yet, but we should have some very soon. Prices are indicated by each program and postage charges are indicated below.

FRENCH MEMBERS please note that cheques in Francs are not accepted by British banks.

### NASBUG/B-BUG MINIMUM SYSTEM MACHINE CODE

### ASSEMBLY LISTINGS

| Prog. No. | Description | Price |
|---|---|---|
| T1 | CRASH 0C50-0FC9 | £0.75 |

Land the space-craft. Real Time. Can you make NASA proud of you? Commented teletype assembly listing. Machine code published in INMC 4.

By H. Birkett

| | | |
|---|---|---|
| T2 | REACTION TEST 0C50-0FCC | £0.60 |

When the random digit appears, press the same digit on the keyboard. Better than 0.6 seconds is good - how good are you? Commented teletype assembly listing Machine code published in INMC 3.

By D. Hunt

| | | |
|---|---|---|
| T3 | BOUNCING BEASTIE 0D00-0DAB | £0.20 |

A character bounces around the screen leaving a trail behind it. Commented teletype assembly listing.

Adapted by D. R. Brough

| | | |
|---|---|---|
| T4 | REVERSE 0C60-0F88 | £0.70 |

Arrange the randomly generated list of numbers in order in the fewest moves by reversing any number of them at a time. Commented teletype assembly listing Machine code published in INMC 3.

By H. Birkett

| | | |
|---|---|---|
| T5 | DOUBLE MASTERMIND 0C50-0FD5 | £0.75 |

Try and crack the NASCOM's four digit code while it tries to crack yours. (By the way, it won't let you cheat on the scoring!) Commented teletype assembly listing. Machine code published in INMC 2

By D. Ritchie

| | | |
|---|---|---|
| T6 | POSSUM 0D00-0D84 | £0.20 |

The hex digits 1 to F are displayed and then flash one at a time. By pressing a button connected to the PIO socket, or any key on the keyboard, the ASCII

character corresponding to the code flashing will then be displayed. For use by the severely disabled. Commented teletype assembly listing.

By F. Butler, modified by D. Brough

T7 JJ 0C50-0DE5 £0.40

Enter the machine code listing of this (in INMC 3) before seeing how it works. Written as an experiment in machine code string handling, but entertaining nevertheless. Commented teletype assembly listing.

By H. Birkett

T8 UNIZAP 0D00-0F74 £0.75

Recreate the life of the universe. Take it from the "big bang" to its "ultimate destiny" by shooting stars. Based on "Shooting Stars" in Byte magazine, May 1976. Commented teletype assembly listing.

Adapted by D. R. Brough

T9 RANDOM BUZZ-WORD LINKER 0C50-0F35 £0.50

Short of jargon? Then use this "Functional organisational flexibility" to give you "Limited transitional communication" in your "Balanced digital engineering". Commented teletype assembly lisiting. Machine code published in INMC 4.

By D. Hunt and R. Beal

T10 QUIZ PROGRAM 0C50-0DAA £0.65
plus ITALIAN FILE 0DB0-0FD9

What is Italian for Please, or English for Fratello? Find out how good your Italian is with this quiz. Also additional files (e.g. Capitals, Cockney, Rhyming slang) available soon. Commented teletype assembly listing.

By M. Bell

T20 ROBOTS 0C70-0FE6 £0.60

Six homicidal robots chasing you across a field of electrified pylons. The robots head directly towards you and so by careful movement you can persuade them to electrocute themselves on the pylons. Speed and difficulty selectable from cinch to impossible! Good instructions, teletype assembly listing, no comments.

By H. Birkett

T21 T.V. TEST PATTERN 0D00-0D19 £0.10

Displays test pattern of "+" symbols on screen to set up colour T.V. Well commented.

By R. Cogliatti

T22 NIM 0D00-0FC4 £1.20

The classic game of Nim. Nicely commented with good instructions.

By N. A. Purver

T23 ATTACK 0D00-0E7F £0.50

Shoot the descending aliens before they get you! Commenting fair. By M. R. Perry

T24 OCTAL TO HEXADECIMAL CONVERTOR 0D00-0DBC £0.25

Rejects all illegal input characters. Well commented. By J. Hill

T25 SPACE-INVASION 0C50-0FD2 £1.20

Very similar to the game seen in many amusement arcades. As you get better, it gets harder. Addictive. Well commented. By G. Clarke

T26 DIGITAL CLOCK 0D00-0D7B £0.30

Simple 24 hour digital clock (hours, mins., secs.) Nice demo of arithmetic and counter manipulation. Well commented but ties up processor 100% By B. C. Winch

T27 ROADRACE 0C50-0FE0 £0.85

Steer your car home with minimum damage. Choice of speed. Addictive. Well commented. By M. Parker-Rhodes

T28 DECIMAL TO HEXADECIMAL CONVERTOR 0C90-0D07 £0.15

Nicely commented. Poor input validation. Positive numbers only. By G. Harriman

T29 LIFE 0C50-0FA0 £0.60

The ubiquituous simulation of life, using special characters to give an expanded Universe. Works well, very fast, good instructions, well commented. By J. Haigh

T30 MOON-LANDER 0C50-0FEC £1.50

Ingenious moon-landing program using keyboard as "crash-pad joystick". Below 2500ft display expands to terrain map, and when you land, look out for those boulders. Well commented, good instructions, teletype assembly listing. INMC games competition winner - object code listing in INMC issue 3.

Submitted by N. Ray

T31 COMPACT ASCII EDITOR 0F70-0FDF £0.15

Crude editor allowing ASCII characters to be located and changed. Well commented, good instructions. (Nice demo, not very practical). By A. Fountain

T32 CARRE CHINOIS 0C50-0F3F £1.50

We have no idea what this game is or what it does (we ain't French is we?) Totally written (and presumably (?) beautifully commented) in French. By G. Bochent

T33 LOLLYPOP LADY TRAINER 0C60-0F95 £0.70

See if you can get the chickens across the road without getting them run over. Well commented.

By M. Parker-Rhodes

T34 RANDOM I-CHING CHARACTER GENERATOR 0C50-0CF3 £0.30

Great mystical significance (for those who understand these things). Nicely commented.

By C. Blackmore

T35 WALLED CHASE 0C50-0E27 £0.90

For two players, one chases the other but here are invisible walls in the way, which appear when you hit them. Fascinating game, well commented.

By S. Montgomery-Smith

T36 SUB-SEARCH 0E0B-0F31 £0.50

Ship traverses screen dropping random depth charges on randomly moving submarines.

By T. Bailie

T37 RANDOM DISPLAY 0C60-0C84 £0.10

Displays random pattern of asterisks. Nicely commented.

By G. Harriman

T38 SUBMARINES 0C60-0E50 £0.50

Hit the randomly displayed submarines with your steerable depthcharges. Good instructions, no comments.

By N. D. Wallbridge

T39 BURST THE BALLOON 0C50-0E02 £0.70

Shoot the balloons as they make their way up the screen, at the mercy of random breezes. Well commented, good instructions.

By J. Waddell

Postage and Packing
UK customers please add 15p per order, overseas customers 40p.

---

## A NOVEL WAY OF EXPANDING A NASCOM 1

Suggested by Mr. L. E. Gilham of Bedfont.

If you have a 1K monitor on a NASCOM 1 a convenient way of expanding by 1K is to put a MK4118 (1K x 8) in the spare EPROM socket. This is done by bending Pins 18,19,20,21 straight and connecting them to CE (old Pin 20), +5v, RD, WR respectively either by putting the chip in a socket and then bending this socket and putting it in the socket on the board or by bending the chip pins and using 'Soldercon' pins. This will give another 1K of RAM from 0400H to 07FFH.

## 8K Basic programs

B1 Hello 0.20
Is your problem health, sex (yes please), money or your job? Let this fun program make some suggestions.

B2 Russian Roulette 0.20
Will you or your Nascom survive as you take it in turns to fire a revolver at yourselves.

B3 Startrek 0.80
A real-time version of this popular program where your mission, as Captain of the Enterprise, is to wipe out the Klingons before your time is up and the ship's energy runs out.

B4 Cubist Art 0.05
Impress the neighbours with the artistic capability of your Nascom. Sort of animated wallpaper. (Graphics required).

B5 3D Noughts and Crosses 0.30
Play against the Nascom on a 4 x 4 x 4 board. Can you beat it or, more difficult, can you take it to a draw. A prize for the first person to convince us that he took it to a draw.

B6 Calender 0.10
Enter the year required (between 1601 and 2399) and let the Nascom calculate the calender for that year.

B7 Magic Labyrinth 0.40
You are in a 5 level labyrinth, and your quest is to find the 5 Objects of State, one on each level. To make matters more interesting there are good, neutral and evil creatures that lurk in the labyrinth, which may help or hinder you as they feel fit.

B8 Eliza 0.40
With this program loaded, your Nascom is specially trained in psychoanalysis. What are your particular hangups? (A full listing appeared in INMC 5.)

B9 Camel 0.30
You have stolen the priceless idol belonging to a tribe of knock-kneed pigmies. They want it back and are in hot pursuit. You are in the desert, on a camel!! Can you reach safety before the pigmies (or wild Neringi Berbers) catch you?

B10 Comrade X 0.60
You are premier of the communist island of Niatirb. You must decide your country's budget, harvest, and economic strategy. You have 8 years in office, can you survive before the peasants revolt, or you are exiled (or something worse)?

The above are all available from the library, singly, at the above prices (+ 15p UK postage, 40p overseas per order), or as Basic Programs Book 1 which also includes other goodies at 2.50 (+ 40p UK postage, 95p overseas).

```
 Piranha by J & E Long
Re-assembled for Nas-sys by
Richard Beal.
Execute at 1000H

T 1000 18E2
  1000 31 00 10 CD D6 14 CD 49 1E
  1008 15 CD 8E 10 CD DA 14 21 74
  1010 5F 18 7E FE 4F 28 0C E5 7B
  1018 CD 9E 10 3A B7 15 B7 F4 54
  1020 90 12 E1 CD 00 12 CD 1F 7E
  1028 11 CD 2D 11 21 5E 18 06 F1
  1030 16 C5 E5 CD 56 13 E1 CD E4
  1038 66 15 C1 10 F4 3A B7 15 8E
  1040 FE FF 20 04 3E 20 18 02 E9
  1043 3E 41 32 F6 0B 18 C0 21 03
  1050 CC 0B 11 9F 15 06 17 1A 33
  1058 77 13 23 10 FA 3A B9 15 27
  1060 C6 30 32 E3 0B D6 30 07 93
  1068 07 07 16 00 5F 2A C2 15 FC
  1070 19 11 BA 15 06 08 7E 12 17
  1078 23 13 10 FA AF 32 90 15 4E
  1080 C9 3A B9 15 FE 09 C8 3C 6C
  1088 32 B9 15 18 C2 E1 CD 29 49
  1090 12 FE 30 D8 FE 3A D0 D6 96
  1098 30 32 B9 15 18 B1 3A B9 94
  10A0 15 3C 47 21 F4 0B CD DB 10
  10A8 10 10 FB 21 62 18 56 23 E7
  10B0 5E CD AE 13 3A B8 15 BA 6D
  10B8 20 19 FE 01 20 04 3E 2E 90
  10C0 18 02 3E 01 32 B8 15 3A 62
  10C8 C1 15 47 21 F4 0B CD E9 CB
  10D0 10 10 F8 3A 90 15 FE 14 E9
  10D8 C0 18 A6 3A B6 15 3C 32 D9
  10E0 B6 15 FE 20 C0 AF 32 B6 30
  10E8 15 7E FE 20 20 03 36 31 33
  10F0 C9 FE 39 C2 FB 10 36 30 33
  10F8 23 18 EE 3C 77 C9 7E FE 31
  1100 20 28 16 FE 30 20 05 36 F8
  1108 39 2B 18 F2 3D 77 FE 30 69
  1110 C0 2B 3E 20 BE C0 23 77 82
  1118 C9 3E FF 32 B7 15 C9 21 17
  1120 8F 15 34 C0 23 34 C0 23 03
  1128 34 C0 23 34 C9 3A BA 15 56
  1130 CD CA 11 FE 01 C0 3E 08 EE
  1138 CD CA 11 B7 C0 CD A7 14 F0
  1140 7C B5 C8 36 01 23 3E 64 46
  1148 CD CA 11 47 3A BF 15 B8 0E
  1150 30 04 36 41 18 02 36 49 A5
  1158 23 3A BD 15 CD CA 11 47 87
  1160 3A BC 15 80 77 2B 2B 77 40
  1168 CD 69 15 E5 3A BB 15 CD 80
  1170 CA 11 FE 03 28 2D FE 02 B2
  1178 28 1D FE 01 28 0C 06 C0 C7
  1180 3E 2E 21 4A 0B 11 01 00 85
  1188 18 23 06 40 3E 2E 21 4A F1
  1190 08 11 01 00 C3 AD 11 06 42
  1198 FF 3E 0D 21 38 08 11 40 A5
  11A0 00 18 0A 06 01 3E 0D 21 46
  11A8 0B 08 11 40 00 CD CA 11 C5
  11B0 3C 19 3D 20 FC EB E1 1A 55
  11B8 FE 20 28 05 2B 2B 36 4F EF
  11C0 C9 70 23 72 23 73 3E 2A 9D
  11C8 12 C9 C5 E5 47 2A 8D 15 71
  11D0 0E 08 7C 0F AC 0F 0F AC F8
  11D8 0F AD 0F 0F 0F EE 01 E6 A7
  11E0 01 29 85 6F 0D 20 EB 22 49
  11E8 8D 15 7C 85 67 78 6F 80 6A
  11F0 B8 30 FB 7C BD 30 D6 B8 DB
  11F8 38 03 90 18 FA E1 C1 C9 51
  1200 DF 62 00 D0 2B 36 01 CD 52
  1208 69 15 E5 21 EA 17 06 1C C1
  1210 BE 28 05 23 10 FA E1 C9 E4
  1218 3E 1C 90 E6 FE 21 CE 17 FE
  1220 16 00 5F 19 5E 23 56 EB 82
  1228 E9 CF C9 00 3E C0 18 1A EB
  1230 3E 01 18 16 3E 40 18 12 57
  1238 3E FF 18 0E 3E BF 18 0A CC
  1240 3E C1 18 06 3E 41 18 02 08
  1248 3E 3F E1 BE 28 05 77 2B 45
  1250 36 2C C9 2B 36 0C C9 E1 A4
  1258 36 00 C9 E1 3A B7 15 2F 7F
  1260 32 B7 15 C9 01 EE 17 11 50
  1268 06 18 21 8C 08 E5 1A 13 5F
  1270 FE FF CA 00 10 77 23 FE F1
  1278 2D 20 F3 CD 29 12 FE 21 F1
  1280 CA 00 10 02 03 03 77 E1 CC
  1288 D5 11 40 00 19 D1 18 DD 9F
  1290 21 5E 18 7E FE 01 C0 3A B0
  1298 F1 0B FE 20 C8 21 F4 0B AC
  12A0 CD FE 10 21 62 18 46 23 91
  12A8 4E CD 07 13 57 2B 2B 7E 1A
  12B0 B7 20 0C 3A B8 15 FE 01 AB
  12B8 3E FF 28 02 3E 01 77 CD B4
  12C0 46 13 5F 07 83 5F 0F B3 35
  12C8 5F A2 B7 C8 06 07 3A D7 78
  12D0 12 EE 08 32 D7 12 7B 07 87
  12D8 5F A2 B7 28 15 10 F7 7B 61
  12E0 07 07 A3 5F 06 08 7B 0F 9A
  12E8 5F A2 B7 28 0B 10 F7 36 22
  12F0 00 C9 7B 07 5F 0F 0F A3 6D
  12F8 CD 40 13 77 06 10 21 F3 CB
  1300 0B CD FE 10 10 F8 C9 E5 AF
  1308 21 74 15 AF 32 8C 15 3E 85
  1310 08 32 8B 15 7E B7 5F 16 A7
  1318 00 F2 1E 13 16 FF EB 09 57
  1320 7E EB 23 FE 20 28 0C FE 0F
  1328 25 28 08 56 3A 8C 15 82 43
  1330 32 8C 15 23 23 3A 8B 15 36
  1338 3D 20 D6 3A 8C 15 E1 C9 03
  1340 E5 21 73 15 18 04 E5 21 03
  1348 74 15 BE 28 05 CD 69 15 1A
```

```
1350 18 F8 23 7E E1 C9 35 C0 B3
1358 3E 01 CD CA 11 B7 C0 23 EC
1360 7E FE 4F C8 FE 58 CA 9B C1
1368 14 E5 FE 49 CC C6 13 E1 41
1370 23 7E 2B 2B 77 CD 69 15 3C
1378 16 00 5E 7B B7 C8 B7 F2 A2
1380 84 13 16 FF 23 46 23 4E 19
1388 0A FE 20 20 06 CD 6E 15 39
1390 36 4F C9 E5 60 69 19 EB A3
1398 E1 1A FE 25 CA 2E 14 FE D3
13A0 20 C2 40 14 2B 72 23 73 1C
13A8 0A 12 3E 20 02 C9 C5 21 E6
13B0 F6 F7 19 11 C0 FF 06 10 AF
13B8 19 05 7C B7 F2 B8 13 58 31
13C0 7D C6 40 57 C1 C9 3E 64 D9
13C8 CD CA 11 47 3A C0 15 B8 91
13D0 D8 CD 69 15 E5 56 23 5E C2
13D8 CD AE 13 42 4B 3A 62 18 BA
13E0 57 3A 63 18 5F CD AE 13 EC
13E8 78 32 57 FE 7F 38 01 2F 41
13F0 47 79 93 5F B7 F2 F9 13 6A
13F8 2F B8 38 0D 7B B7 FA 05 68
1400 14 3E 40 18 0F 3E C0 18 E3
1408 0B 7A B7 F2 12 14 3E 01 AF
1410 18 02 3E FF E1 2B 47 7E 4C
1418 B8 C8 2F B8 28 02 70 C9 F6
1420 FE C0 28 07 FE 40 28 03 8A
1428 36 40 C9 36 01 C9 CD 6E B6
1430 15 7E FE 50 28 05 2B CD 4A
1438 9A 14 C9 AF 23 23 77 C9 F8
1440 CD 6D 15 CD 9A 14 CD BB A6
1448 14 CD 9A 14 21 01 00 09 16
1450 CD 69 14 21 FF FF 09 CD A3
1458 69 14 21 40 00 09 CD 69 89
1460 14 21 C0 FF 09 CD 69 14 BB
1468 C9 7E FE 20 CA 80 14 FE 3D
1470 25 C8 FE 2B C8 E5 C5 EB F7
1478 CD BB 14 CD 9A 14 C1 E1 45
1480 EB C5 CD A7 14 C1 7C B5 BE
1488 C8 3A BE 15 77 23 36 58 99
1490 CD 69 15 72 23 73 3E 2B 60
1498 12 C9 23 36 4F CD 69 15 7A
14A0 46 23 4E 3E 20 02 C9 21 B5
14A8 65 18 06 15 7E FE 4F 28 47
14B0 08 CD 66 15 10 F6 21 01 3C
14B8 00 2B C9 21 5E 18 06 16 73
14C0 CD 68 15 7E 23 BA 20 07 A0
14C8 7E BB 20 03 C3 6D 15 23 A0
14D0 10 EE 21 00 00 C9 AF 32 AD
14D8 B9 15 21 00 08 06 10 0E 07
14E0 40 36 20 23 0D 20 FA 10 E4
14E8 F6 21 0A 08 11 8A 0B 06 D1
14F0 30 3E 25 77 12 23 13 10 66
14F8 FA 06 0E 21 4A 08 77 0E 12
1500 2F 23 0D 20 FC 77 0E 11 26
1508 23 0D 20 FC 05 20 EF 21 9E
1510 5E 18 36 2C 23 36 50 23 C9
1518 36 2C 23 36 00 23 01 E2 EE
1520 09 70 23 71 3E 07 02 16 9F
1528 15 23 23 36 4F CD 68 15 67
1530 15 20 F6 21 E9 0B 06 0C 97
1538 11 93 15 1A 77 13 23 10 DD
1540 FA 3E FF 32 B7 15 C3 4F 9C
1548 10 21 4D 08 06 0D 11 14 1B
1550 16 E5 1A 13 FE 40 28 04 F7
1558 77 23 18 F6 E1 D5 11 40 1C
1560 00 19 D1 10 EC C9 23 23 6A
1568 23 23 23 23 C9 2B 2B 2B 53
1570 2B 2B C9 01 C0 01 02 C1 29
1578 02 04 01 04 08 41 08 10 F9
1580 40 10 20 3F 20 40 FF 40 E3
1588 80 BF 80 01 00 BB 70 D1 59
1590 37 00 00 53 63 6F 72 65 D8
1598 20 20 20 20 20 20 30 2A C7
15A0 2A 20 50 49 52 41 4E 48 C1
15A8 41 20 2A 2A 20 20 20 20 F2
15B0 50 68 61 73 65 20 07 00 DD
15B8 2E 09 28 04 28 3C 20 64 18
15C0 1C FF C4 15 48 01 05 60 77
15C8 30 0A 01 32 48 01 50 5C 3F
15D0 30 14 04 4B 44 01 48 58 5D
15D8 2C 1E 07 64 44 01 44 54 7F
15E0 2C 28 0A 7D 40 02 40 50 A2
15E8 28 32 0D 96 40 02 3C 4C C4
15F0 28 3C 10 AF 38 02 38 48 E2
15F8 24 46 13 C8 38 02 34 44 04
1600 24 50 16 E1 30 04 30 40 25
1608 20 5A 19 FA 28 04 28 3C 3B
1610 20 64 1C FF 59 6F 75 20 22
1618 68 61 76 65 20 66 61 6C 25
1620 6C 65 6E 20 69 6E 74 6F 4F
1628 20 74 68 65 20 41 6D 61 CE
1630 7A 6F 6E 2C 20 51 6E 64 1C
1638 40 20 6D 75 73 74 20 61 F8
1640 76 6F 69 64 20 74 68 65 69
1648 20 6D 61 6E 2D 65 61 74 21
1650 69 6E 67 20 50 69 72 61 50
1658 6E 68 61 21 40 50 72 65 2D
1660 73 73 20 61 6E 79 20 6B 4F
1668 65 79 20 74 6F 20 73 74 66
1670 61 72 74 2C 20 6F 72 20 1A
1678 30 2D 39 20 74 6F 20 73 BA
1680 65 74 20 70 68 61 73 65 A0
1688 2E 40 50 72 65 73 73 20 39
1690 48 20 74 6F 20 73 74 6F 67
1698 70 2C 20 6F 72 20 74 68 47
16A0 65 20 61 64 6A 61 63 65 93
16A8 6E 74 20 6B 65 79 73 20 9C
16B0 74 6F 40 20 6D 6F 76 65 C0
16B8 20 69 6E 20 61 6E 79 20 4D
16C0 64 69 72 65 63 74 69 6F 29
16C8 6E 2E 20 20 50 72 65 73 54
```

```
16D0 73 20 6B 65 79 20 74 77 CD
16D8 69 63 65 40 20 74 6F 20 82
16E0 6D 6F 76 65 20 6D 6F 72 1B
16E8 65 20 71 75 69 63 6B 6C 0C
16F0 79 2E 40 53 77 69 6D 20 AD
16F8 66 72 6F 6D 20 73 69 64 22
1700 65 20 74 6F 20 73 69 64 DF
1708 65 20 66 6F 72 20 62 6F DC
1710 6E 75 73 20 70 6F 69 6E 53
1718 74 73 2E 40 57 68 65 6E 16
1720 20 74 68 65 20 73 63 6F FD
1728 72 65 20 69 73 20 6F 76 17
1730 65 72 20 31 30 30 30 2C 2B
1738 20 74 75 72 6E 20 6F 6E 35
1740 20 6F 72 40 20 6F 66 66 F3
1748 20 61 75 74 6F 70 69 6C 7D
1750 6F 74 20 62 79 20 70 72 47
1758 65 73 73 69 6E 67 20 41 59
1760 2E 40 50 72 65 73 73 20 12
1768 50 20 74 68 65 6E 20 30 EE
1770 2D 39 20 74 6F 20 73 65 E8
1778 74 20 70 68 61 73 65 2E 62
1780 40 50 72 65 73 73 20 53 57
1788 20 66 6F 72 20 6E 65 77 70
1790 20 67 61 6D 65 2E 40 50 1F
1798 72 65 73 73 20 21 20 74 41
17A0 6F 20 63 68 61 6E 67 65 AC
17A8 20 6B 65 79 73 20 75 73 A3
17B0 65 64 2E 40 50 72 65 73 98
17B8 73 20 22 20 74 6F 20 6C 13
17C0 65 61 76 65 20 70 72 6F E9
17C8 67 72 61 6D 2E 40 00 00 F4
17D0 64 12 00 10 2C 12 34 12 F1
17D8 38 12 30 12 3C 12 40 12 1B
17E0 48 12 44 12 57 12 8D 10 AD
17E8 5B 12 22 22 21 21 53 0D 52
17F0 59 33 4E 32 47 34 4A 36 13
17F8 54 37 55 39 42 31 4D 33 1B
1800 48 35 50 2E 41 30 4E 65 37
1808 77 20 67 61 6D 65 2D 55 D3
1810 70 2D 44 6F 77 6E 2D 4C D6
1818 65 66 74 2D 52 69 67 68 26
1820 74 2D 55 70 2C 6C 65 66 01
1828 74 2D 55 70 2C 72 69 67 14
1830 68 74 2D 44 6F 77 6E 2C 15
1838 6C 65 66 74 2D 44 6F 77 52
1840 6E 2C 72 69 67 68 74 2D 3D
1848 48 61 6C 74 2D 50 68 61 2F
1850 73 65 2D 41 75 74 6F 70 76
1858 69 6C 6F 74 2D FF BF 4F 62
1860 2C 01 0A 16 01 49 60 FF 6E
1868 0A AF 28 49 2F FF 08 72 52
1870 02 4F 46 FF 0A D4 02 4F 4D
1878 5C C0 0B 15 00 49 2F 40 84
1880 0A 58 03 4F 2D C0 0A 95 D8
1888 13 4F 3D 40 0A 93 22 4F 8D
1890 61 C0 0A D4 5F 4F 44 40 D9
1898 0B 53 12 4F 32 C0 0A D4 3F
18A0 12 4F 2D 40 0A 54 1A 4F 4D
18A8 3A FF 09 D6 1A 4F 25 FF 65
18B0 09 56 83 4F 3E C0 09 AC AC
18B8 83 4F 2A FF 09 2C E5 4F 34
18C0 1B 40 09 2A D6 4F 4F 00 DA
18C8 08 DF 99 4F 4F 00 09 B4 BB
18D0 B5 4F 4F 00 FF 4F B5 4F 3D
18D8 FF 00 FF 00 B5 4F FF 00 F1
18E0 FF 00 FF 00 FF 00 FF 00 F4
```

---

PLUG.PLUG.PLUG
==========

The Merseyside Nascom Users Group, some 150 strong, has put together a 64 page, A4 book entitled "Nascom Programs and Information". The main part of the book is dedicated to the programs, some fifteen in all including:

3D noughts and crosses
Othello
Income Tax
Screenwriter
Pico Pilot
Crash Landing
2K Tiny Basic

The INMC is able to offer these to its members for 2.75 including postage and packing. Please note that the programs in this book are for use with Nasbug T1, T2, T4 and B-Bug but not Nas-Sys. For use with Nas-Sys the various monitor calls require changing, other patching may be required.

---

```
10 REM           *** HANGMAN ***
20 REM
30 REM By Howard Birkett, with minor assistance
40 REM from Paul Greenhalgh, and mods by Dave Hunt.
50 REM
60 REM For use with Nascom 1/2 using Nas-sys 1
70 REM monitors only.
80 REM
90 REM The vocabulary may be changed by
100 REM altering the DATA statements from line
110 REM 1010 onwards. The DATA statement at
120 REM line 1000 is the number of words in the
130 REM vocabulary.
140 REM
150 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
160 H$="HANGMAN"
170 DIM A(26),W(20):C=0:P=0
180 CLS:GOSUB 950:PRINT
190 INPUT "What is your name ";N$
200 RESTORE:READ I:DIM R(I)
210 FOR J=1 TO I:R(J)=0:NEXT
220 REM
230 REM ** Re-enter here
240 CLS:GOSUB 950 :W=0
250 SCREEN 1,2:PRINT N$" ="P;TAB(24) "Computer ="C
260 FOR I=1 TO 26:A(I)=0:NEXT
270 RESTORE 1000:READ R:R1=10*R
280 FOR I=1 TO R1:R9=INT(RND(1)*R+1)
290 IF R(R9)=0 THEN R(R9)=1:GOTO 310
300 NEXT:FOR I=1 TO R:R(I)=0:NEXT:GOTO 280
310 FOR I=1 TO R9:READ W$:NEXT
320 FOR I=1 TO LEN(W$):W(I)=0:NEXT
330 T=0
340 GOSUB 680:GOSUB 740
350 GOSUB 800
360 IF T=7 THEN 540
370 IF W=1 THEN 650
380 SCREEN 5,13:INPUT "What letter ";L$
390 REM
400 REM ** Check input validity
410 IF LEN(L$)=1ANDL$>="A"ANDL$<="Z" THEN 450
420 GOSUB 870:SCREEN 5,13
430 PRINT"One LETTER only will do !!"
440 GOSUB 900:GOSUB 870:GOTO 380
450 L=ASC(L$)-64:IF A(L)=0 THEN 480
460 GOSUB 870:SCREEN 5,13
470 PRINT "You've already tried "L$:GOTO 440
480 T1=0:A(L)=1:FOR I=1 TO LEN(W$)
490 IF L$=MID$(W$,I,1) THEN W(I)=1:T1=1
500 NEXT:GOSUB 870
510 IF T1=0 THEN T=T+1
520 FOR I=1 TO LEN(W$):IF W(I)=0 THEN 340
530 NEXT:W=1:GOTO 340
540 REM
550 REM ** Lose message
```

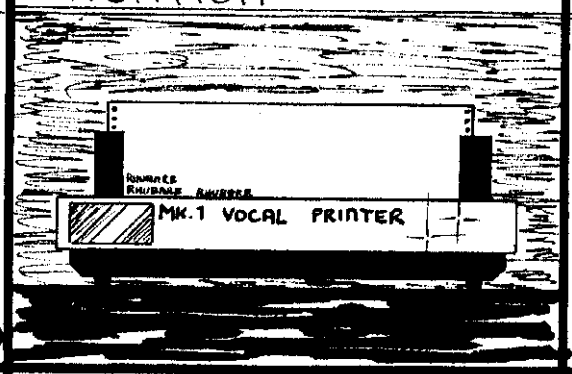
```
550 C=C+1
570 SCREEN 1,13
580 PRINT"Bad luck "N$" you lost.":PRINT
590 PRINT"The word was "W$".";
600 INPUT " Play again ";Q$
610 IF LEFT$(Q$,1)="Y" THEN 220
620 CLS:GOSUB 950 :PRINT:PRINT "Goodbye "N$:END
630 REM
640 REM ** Win message
650 SCREEN 1,13
660 PRINT "Well done "N$", you won (this time !!)"
670 P=P+1:GOTO 600
680 REM
690 REM ** Print "H A N G M A N" message
700 SCREEN 18,4:FOR I=1 TO 7
710 IF T>=I THEN PRINT MID$(H$,I,1);" ";:GOTO 730
720 PRINT "* ";
730 NEXT:PRINT:RETURN
740 REM
750 REM ** Print the letters used
760 SCREEN 5,7:PRINT "Letters used : ";
770 FOR I=0 TO 25:IF A(I)=0 THEN 790
780 PRINT MID$(A$,I,1);
790 NEXT:PRINT:RETURN
800 REM
810 REM ** Print the word so far
820 SCREEN 5,10:PRINT "Word is : ";
830 FOR I=1 TO LEN(W$)
840 IF W(I)=0 THEN PRINT "- ";:GOTO 860
850 PRINT MID$(W$,I,1);" ";
860 NEXT:PRINT:RETURN
870 REM
880 REM ** Clear input line
890 SCREEN 1,13:PRINT SPC(46):PRINT:RETURN
900 REM
910 REM ** Delay; change to 2000 for 4MHz
920 FOR Z=1 TO 1000:NEXT:RETURN
930 REM
940 REM ** Put up title
950 FOR I=1 TO LEN(H$)
960 POKE 3037+I,ASC(MID$(H$,I,1)):NEXT:RETURN
970 REM
980 REM ** Vocabulary
990 REM    Change line 1000 to number of words
1000 DATA 49
1010 DATA KING,GARDEN,FRIEND,CASTLE,GARAGE
1020 DATA LAMPSHADE,FINGER,DRAWER,FIREPLACE
1030 DATA MACHINE,COMPUTER,RADIATOR,LEATHER
1040 DATA KETTLE,PICTURE,YACHT,HARBOUR
1050 DATA KITCHEN,ENGINE,TELEPHONE,SHOP
1060 DATA BICYCLE,ALBATROSS,GIRAFFE,TREE
1070 DATA NEWSPAPER,COOKER,SAUCEPAN,STREET
1080 DATA FLOWER,PENCIL,HANDBAG,DIARY,DAIRY
1090 DATA PARAPHERNALIA,DOG,MOUSE,ELEPHANT
1100 DATA FRUITCAKE,CASSETTE,CROQUET
1110 DATA TELEVISION,GUITAR,PIANO,ZEBRA
1120 DATA UMBRELLA,BIBLE,BANANA,HEDGEHOG
Ok
```

# THE PERSECUTION OF THE INTELLECTUAL

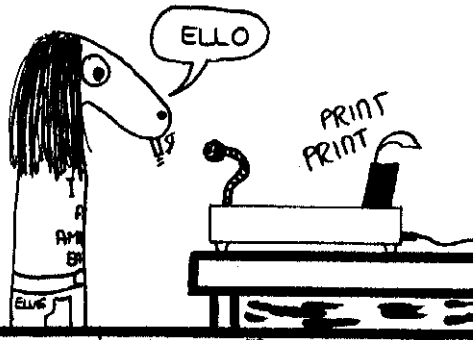
SAGA FOUR : noblesse obligé

starring lawrence the long-haired (weirdo)

LAWRENCE'S LATEST INVENTION



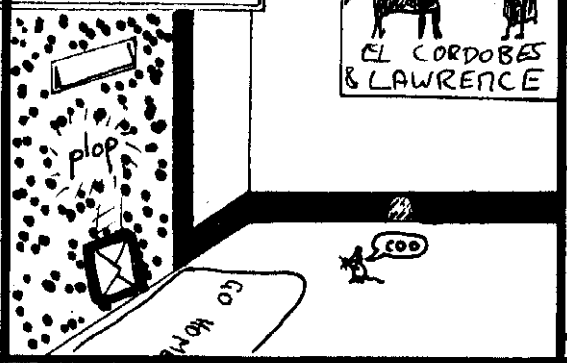

UNDERGOING TESTS



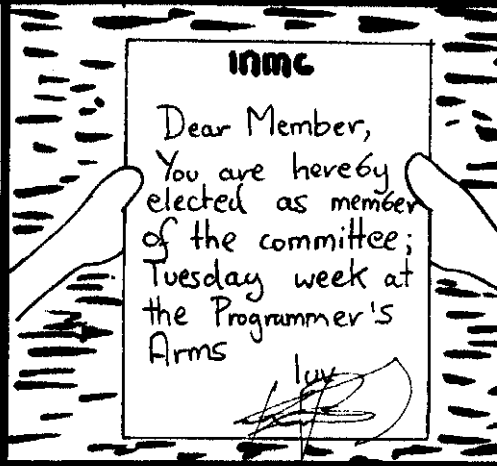

WHICH IT HAS YET TO PASS

OINK

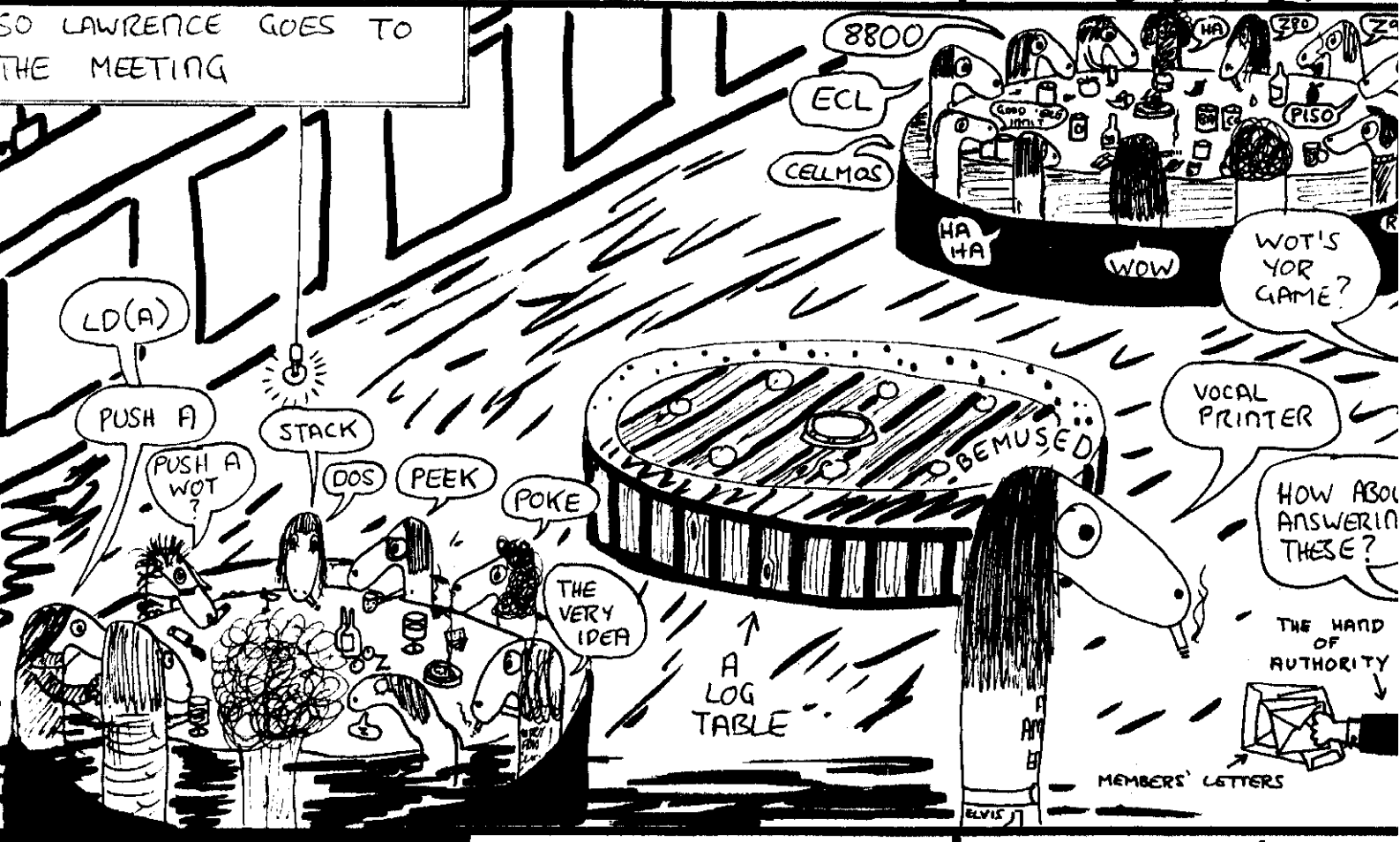
WHEN








SO LAWRENCE GOES TO THE MEETING





MONTHS LATER




Letters offering suggestions for getting the vocal printer up and running should be sent to lawrence c/o inmc and will be answered first.