80-BUS NEWS

APRIL – JUNE 1982 VOL.1 ISSUE 2

The Magazine for NASCOM & GEMINI USERS

£1.50

April – June 1982. **80-BUS NEWS** Volume 1. Issue 2.

CONTENTS
========

 Published by Interface Data Ltd and printed by Excelsior Photoprinting Ltd., High Wycombe.

SUBSCRIPTIONS

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

Subscriptions to 'Subscriptions' at the address below.

EDITORIAL

Editor : Paul Greenhalgh Associate Editor : David Hunt

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

ADVERTISING

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

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

EDITORIAL

Welcome to Issue 2 of 80-BUS News.

We have been very pleased to receive all of the letters congratulating us on the first issue. It is obviously very encouraging that so many people have taken the time to comment on the format and content of the magazine and we will endeavour to make some of the modifications that have been suggested.

There have also been, inevitably, one or two letters with an amount of criticism, but this criticism has been mainly aimed at the inefficiency of the INMC80 magazine, which we have, as you know, taken under our wing. Steps are being taken to rectify the shortcomings in these areas, as we are well aware of them. One problem has been in maintaining an accurate mailing list, as until now this has been handled manually. We are about to take a somewhat retrograde step and computerise!! This will mean some considerable fun during the changeover period and we would ask you to have a little (lot of?) patience as we move the sizeable mailing list from paper to disk.

Another complaint has been at the irregularity of issues. Well, again we know, we apologise, we have had a little hiccough at the start, but we have made a promise to produce a given number of issues within a given time period, and this we will do !! (We will, we will, we will.)

In the last editorial I commented on the amount of material we had received (and published) on disk systems. Well, there seems to have been a natural reaction to this, and this magazine escapes with hardly a word about disks. (Well, nearly.) But there are articles about disk systems arriving for the next issue already, so anti-disk readers had better get scribbling quickly if they want the magazine to maintain a nice even balance ! We have also received a couple of hardware articles and so Issue 3 is well underway.

A number of people have written asking 'Whatever happened to the INMC program library?' Not an easy one this. The program library was, to put it mildly, too successful. Vast numbers of programs used to arrive at the INMC, with no-one with the time to even look through them, let alone sort out which of the 27 versions of Hangman, or 18 versions of Life was the best. In addition some programs arrived hand-written, or as carbon copies, or printed on an arthritic antique teletype, or on Nascom 1 format tapes (single or 3.83642 times speed!), and even the occasional Nascom 2 tape (which was about the only promising media, until it was discovered that the sender had forgotten to plug in the cassette lead and all that was there was Joey the budgie saying rude things). All in all a disaster. Then one day a volunteer was found to sort it all out (mad fool) and so off he went with van-fulls of paper and tape ..... never to be seen again (not so mad). And that brings us almost to date. So, in answer to this recurring question, all I can say is that we do hope that one day we will be able to resurrect the library, but in the meantime, please don't call us.....(P.S. Anyone seen the van driver?)

One popular item in this sort of magazine is the review. If anyone has purchased any software or hardware that they think everyone else should either buy or steer well clear of, then please send us a review. (We pay, you know?)

And finally a little glimpse into a few items that we hear may actually have appeared at last, or may appear in the next few weeks. Anyone who manages to find any of them please write in to let us know if they work! I/O Research's Pluto; Nascom's AVC; Gemini's GM816 I/O Board, GM813 combined CPU-I/O-64K RAM, & GM822 Real Time Clock; Nascom's, Gemini's and Hi-Soft's PASCALs. Plus, has anyone any comments to make on Nascom's N3, Gemini's Galaxy 1, EV Computing's IEEE488 or Microde's Static RAM board. We look forward to hearing from you.

LETTERS TO THE EDITOR

HISOFT PASCAL - 1

Readers may recall that in the Jan-Mar '82 issue of 80-Bus News a letter from Mr Holy (in fact written seven months earlier) raised three points concerning Hisoft to which I would reply as follows:-

1. We now always publish our address and telephone number, although I agree we have not yet been listed in the telephone directory. With a new business which was likely to change premises I did not wish to inconvenience customers by a change of phone number in the directory.
2. There is no discrepancy between the workspace addresses given in the documentation and those actually used by the Pascal compiler. Unfortunately Mr. Holy had been sent the list of addresses for use under NASMON not Nas-Sys. This was easily rectified.
3. Owing to my absence abroad I was not able to reply to Mr Holy's letter as promptly as possible.

Unfortunately Mr Holy's letter was published some seven months after it had been written but I guess it is just one of those that slipped through the net and was published long after it had been dealt with satisfactorily.

Further information about Hisoft Pascal can be obtained from the advert which appears elsewhere in this issue.

D.G. Link of Hisoft.

HISOFT PASCAL - 2

I am referring to my letter published in 80-Bus News vol. 1, issue 1, regarding the Hisoft Pascal compiler.

It appears that a rather unfortunate error has been made by your magazine, in that the above letter was published nine months after it was written. This has put the letter almost completely out of context, particularly as in the meantime Hisoft had contacted us and had been very helpful indeed.

I can now state that, as far as we have been able to determine, the Naspas-3 compiler contains no 'bugs' at all. We are very pleased with it and with the assistance which we have recently received from Hisoft.

With hindsight, it would appear that our difficulties were due to -

1) a page or two missing from the original documentation and
2) the apparent inability of some sections of the run-time code to operate when interrupts were present. The latter point is relatively unimportant as far as Nascom users are concerned since most Nascoms do not use interrupts.

P. Holy of Ringdale Engineering Co. Ltd

HISOFT PASCAL - 3

Users of HISOFT NASPAS 3 (with the trigonometric extensions) will probably be aware of a few minor bugs in the compiler. With the permission of the authors of this compiler, I publish here details of how to fix these bugs. This fix applies to copies of the compiler purchased after 1st May 1981. One of these fixes may not be necessary for the later compilers, but I include it for completeness.

I deal with four distinct bugs. These are:

1) COS(0) = -1 instead of 1
2) Faulty handling of explicit string parameters e.g. PROC1('Pascal');.

3) Oddities in the handling of REALs far down the Procedure/Function stack.
4) Faulty handling of ARRAY types in PEEKs. This has been corrected in the later versions of the compiler, but here is the correction anyway.

First of all, load the compiler master tape, and relocate as usual, using an address for the Runtimes routines which is 10 bytes less than normal, and for the compiler which is 35 bytes less than normal (at least, in both cases). Before doing anything else it is advisable to have pencil and paper to hand. You should be familiar with using the A command of the Nas-Sys to calculate hex addresses and the M command to modify memory. COMP will indicate the Compiler address, RUN will indicate the Runtime address. The appropriate hex values should be substituted for these labels.

1) Calculate RUN + OCA4H. At this address, you should find CD XX XX (using M command). Replace XXXX with RUN + OFE1H. You are now setting up a jump over the runtimes to a patch at their end. At RUN + OFE1H enter the following code -

CB 74 C2 XX XX 26 40 E3 E1 C9

where XX XX is RUN + ODE2H. Remember ALWAYs to enter the low order byte first.

2) At COMP + OB6CH change OC 06 02 to CD YY YY where YYYY = COMP + 285FH.
At COMP + 1E95H change 06 02 EB to CD ZZ ZZ where ZZZZ = COMP + 285CH.
At COMP + 1D95H change EB C1 C3 UU UU to OO C1 C3 VV VV,
where VVVV = UUUU + 1.
At COMP + £285C enter the following code -

EB 18 01 OC 06 02 F5 79 32 FE OC AF 32 FF OC F1 C9

3) At COMP + 1FB2H replace the current three bytes with CD SS SS where SSSS is the address of the byte after the C9 in the previous section.
At that address, add the following code -

DD CB 01 C6 21 TT TT C9 where TTTT = COMP + 1FD5H

4) At COMP + 1EDEH change bytes from CA NN NN to 28 29 23. If they are already in position, then your compiler has had the PEEK bug corrected.

Having done all this, now make a tape of the relocated compiler and runtime support routines. Remember that the compiler requires four parameters on entry, so you can't write a Generate tape. Make allowance for the increased length of the compiler and runtimes when saving on tape, and remember that any compilers relocated by the master tape will again require the same procedure to be gone through. In all cases, the addresses are entered into memory with the low order byte first.

I am greatful to the authors of this compiler for details of these fixes, and for permission to bring them to you.

Rory O'Farrell, Ireland.

PEN - 1

I run CP/M 2.2 on a Nascom 2 with Gemini DD/DS drives and an IVC. When using PEN I sometimes suffer from not remembering which case the keyboard is in. The differences between "d" and "D" can be somewhat frustrating.

My original idea was to use the cassette LED to indicate the current case, but I found that whenever the keyboard was addressed the LED was put out. Since I have an N2 keyboard I have no need for the Control/Backspace toggle provided in the BIOS and so I decided to patch the BIOS and use this function to gain positive control of the case lock. With the patch below the Control/Backspace input always sets the case lock to small letters, the Control/Enter input remains the same - it flips the case lock each time. The last patch causes the case shift to be locked on capitals whenever the system is booted from cold.

The patch is a listing of DDT as it is displayed on the screen.

```
DDT MOVCPMV.COM
DDT VERS 2.2
NEXT  PC
3100 0100
-S2931
2931 36 3A
2932 FE .
-S2935
2935 36 32
2936 FE .
-S2968
2968 3E 3A
2969 01 E9
296A 18 1E
296B 02 21
296C 3E 3E
296D 02 01
296E 21 .
-S29E9
29E9 00 01
29EA E1 .
-GO
SAVE 48 MOVCPM.COM
```

Steve Willmott, West Drayton.

PEN -2

This small routine will enable NASPEN users to indent any text from the left hand margin without having to insert the indentation directly into the NASPEN buffer. It does this by detecting all CARRIAGE RETURNs (ODH) sent to the printer (IMP assumed) and printing a pre-determined number of spaces before returning to NASPEN. The code can be placed in any convenient part of memory. For those with IMPRINT, code 02 (bi-directional printing) can also be trapped when it occurs at the end of text. This allows unidirectional printing to continue till the IMP print buffer is empty.

```
        CP 2            ;omit if not required
        RET Z           ;this too
        SCAL £6E
        CP £D           ;is it a CR
        RET NZ          ;if not return
        LD  A £20       ;space
        LD B 8          ;indent of 8 spaces
LOOP    SCAL £6E
        DJNZ LOOP       ;print spaces
        RET
```

Enter the routine into memory, say at £0C80, then change the NASPEN printer reflection to jump to the routine. i.e. modify £101D (DF 6E C9) to C3 80 0C. The indent can be changed by loading B with the required number of spaces. Then warm start NASPEN.

R. Mohamed, Glasgow

NASCOM BASIC

This piece of information may well be common knowledge among the wiser Nascomers, but I feel sure that many people will be glad of it.

The USR routine to scan the keyboard under Nas-Sys 3, given in Appendix I of the Nascom BASIC manual, only detects the initial pressing of a key, and returns a 0 if the latter is subsequently held down and the routine re-run. The solution is to load the keyboard repeat counter (KCNT) with 1, before calling the input routine RKBD. The subroutine becomes:

```
0C80    21 01 00        LD HL, 1        ; set keyboard
0C83    22 2C 0C        LD (KCNT),HL    ; counter to 1
0C86    DF 7D           SCAL RKBD       ; scan keyboard
0C88    38 01           JR C,CHAR       ; skip if char.
0C8A    AF              XOR A           ; clear A
0C8B    47      CHAR    LD B,A          ; char in B
0C8C    AF              XOR A           ; clear A
0C8D    2A 0D E0        LD HL,(E00D)    ; get add. in HL
0C90    E9              JP (HL)         ; jump and return
```

In BASIC, this is:

```
10 DOKE 4100,3200: FOR I=3200 TO 3216 STEP 2
20 READ A: DOKE I,A: NEXT
30 DATA 289, 8704, 3116, 32223, 312
40 DATA 18351, 10927, -8179, 233
```

Michael D'Arcy, Bristol

EPROM Erasers & IMP ribbons

A cheap alternative to forking out £40 odd for an EPROM Eraser is the ordinary UV Sunlamp. Having no idea whether the power or UV frequency would be suitable, I did some experiments and found that 20 min at 1" from the UV bulb did the trick. The lamp I used was the small Boots one and this has 2 IR heating elements which are an essential part of the circuit and cannot be disconnected. To avoid the EPROM being cooked I mounted it on a wet sponge and directed a fan at it - These measures kept the EPROM fairly cool. Some experiments will probably be needed, as the lamp may differ from mine. I can't guarantee the EPROM won't be damaged by heat but they are cheap enough to take the risk - I have now erased mine many times without trouble however.

Does any reader have any information on how to re-ink ribbons for the Imp - i.e. type of ink and where it can be obtained? (I know it can be done with the correct ink despite the manual recommending this is not attempted). Incidentally I note that the article in INMC 80-4 on IMPRINT mentioned oiling the cam. I believe this may be inadvisable since ordinary oil can cause the nylon to distort. My interpretation of the manual is that only IBM 22 or equivalent grease should be used on the nylon bits (oil is OK on the metal rail). I had trouble in getting this locally and got it direct from the Great IBM (Greenford) by post.

P.A.Cooper of Brentwood, Essex

ERROR !

With reference to Dr. Dark's Diary in the last issue of 80-Bus News, the section titled 'Another Nas-Sys 3 fix' contains an error. The location that requires alteration is B15B.

Whilst on the subject of the Bits & P.C's BASIC Programmers Toolkit, have you ever tried running the ROM version of it in RAM? It doesn't like being rehoused. This is unless the firing pins are removed from a few bombs that have been placed. To defuse these place NOP's in the following locations:

B020 and B021
B23E and B23F
B247 and B248

Don't forget to also alter the reset jumpt at B000 - 003 to the toolkits new location +3. Hope this is of help to someone.

Mel Warwick, Grantham, Lincs

ZEAP MOD

If you have the cassette version of ZEAP 2.0 on your Nascom you can add the facility of a tabulator function when you use the Auto Input Mode. It is annoying to have to hit the space bar to move the cursor to the next field every time you do not have a label to type in. When adding the function below, typing "enter" causes the assembler to figure out where on the line the cursor is and moves it to the mnemonic field if the cursor position before was in the label field or to the next line if it was in the mnemonic field. If you type a ";" the cursor moves to the comment field and ";" is typed out, except when the cursor position before was in the label field. Then ";" is typed out at the first position in the label field.

Change these positions listed below and type in the following code. It is assumed that the modifications described in INMC News issue 7 have been made. The free space will begin at 2050H after a cold start. Address 2022H and 202AH contains the first position of the mnemonic field and 203FH the first position of the comment field counted from start of the line.

```
1006    4A
1493    49 20
182B    4A
1C36    CD F9
1C38    1F 18 02
1FF8    C9 2A 87 0F DF 66 DF 7B
2000    4F FE 1B 28 3D FE 0D 28
2008    09 FE 3B 28 03 F7 18 EE
2010    06 05 11 C0 FF 19 E5 DF
2018    7C D1 EB B7 ED 52 26 3B
2020    7D FE 0D 30 15 7C B9 28
2028    02 06 0D 3E 17 F7 3E 12
2030    F7 10 FD 7C B9 20 C7 F7
2038    18 C4 7C B9 20 05 06 23
2040    18 E9 F7 DF 6A C9 00 00
2048    00 B6 06 00 00 00 00 FF
```

Mats Olofsson, Sweden

GEMINI DRIVES ON AN N1

Just before Xmas '81 I took delivery of a shiny, brand new 100% certified functional, never before used and carefully wrapped Gemini G805 single drive disk unit with D-DOS for my Nascom 1.

Of course, after specifying to my dealer that it was for use with a N-1 the unit was supplied with a plug (on the ribbon cable) to fit a Nascom 2 PIO. There now follows a warning to any naive person who is considering investing in a G805 and who

also still thinks that items of the genus computer were designed to fit together effortlessly.

The really BIG error was, of course, in the literature. Not once did any of the manuals state that the D-DOS software was written for a N-2 and that it would only work on a N-1 if (and only if) the N-1 was running at 4MHz, all you'd get at 2MHz would be a clunk, a wheerr (if you get my meaning) and a system crash. [Ed. - We can think of no reason why D-DOS/G805 shouldn't be run at 2MHz. Perhaps your Nascom clock is running very slow, as below about 1.85MHz the software loop isn't fast enough to get the data.] This fact took me a couple of hours of grief and worry, while standing ankle deep in bits of my favourite computer, to find out (not to mention a peak rate telephone call to EV Computing in Manchester).

When the thing was running 'right proper like' another problem raised its head (as usual). The original address of the D-DOS software is B000H which messes up the memory map of a 48K machine (brag, brag) very nicely. To overcome this problem I moved D-DOS up to D000H. After disassembling the first 1K of D-DOS by hand, I came up with the following -

B000 C300B4	B010 C370B2	B013 C3A5B2	B016 C331B0	B019 C3DAB2	B01C C3A1B3
B01F C373B1	B022 C304B2	B025 C339B2	B028 C324B1	B02E C307B1	B03F CD07B1
B042 CDE3B0	B04C C24BB0	B053 CDC4B0	B06F CD60B0	B07E CD57B0	B086 CD60B0
B095 CD57B0	B09D CD57B0	B0B3 CD57B0			

From the above, I think you can see that the only thing you need do to D-DOS to move it to another memory location is to alter the addresses of all the JUMPS and subroutine CALLS to the new address. For example, to run D-DOS at D000 change all the jump and call addresses from BXXX to DXXX. To access a disk under Nas-Sys 3, all I have to do is type 'D' and 'NL' and I'm straight into D-DOS.

Also of some minor interest is the fact that D-DOS is no longer in EPROM but in a 2K block of non-volatile RAM which is write protected and it seems happy there!

D.G. Richards of Glamorgan, S.Wales

AND FINALLY, THANKS.

Many thanks for a fine magazine - it was the quality of this, with all its information on hardware etc that finally persuaded me to purchase a Nascom 2. Even if I hadn't bought a Nascom it is worth buying the mag. for the 'Teach Yourself Z80' series alone - many thanks to Dave Hunt. [Ed. - Spare the blushes Dave, I haven't published the letter that slates you something rotten!] It took me quite a few readings to understand B2HEX, but I finally got the jist of it. I then found that in my Nas-Sys 1 the routine was different (but much easier to work out). I even plucked up courage to single step thro' the routine, and then wished I hadn't - it was only a few days later that I came across the fact that certain Nas-Sys routines cannot be single stepped! Anyway, I am still looking forward to the rest of the series. [Ed. - Fool!]

Finally, are there any other readers in the King's Lynn area who would like to make contact?

Paul Tostevin, 8 Sidney St., King's Lynn, Norfolk, PE30 5RH. Tel. 5174.

EDITOR'S NOTE - Letters are very welcome on any Nascom/Gemini related topic and the author of any letter published will receive £3. We also would very much like to hear from you about any computer clubs which have been set up, or which are in the process of being formed. Perhaps representatives of the various clubs could write giving details of their meetings and activities as their are probably lots of potential members reading this.

INPUTting and READing Double Precision Constants

by Mike York

Some of the purchasers of my Double Precision Package (DPP) extension of the Nascom ROM BASIC (marketed by myself at 9 Rosehill Rd, London SW18 and also by Business & Leisure and the Microvalue group as "MathsPak") have pointed out that it does not support the INPUTing or READing of double precision constants. As things stand, double precision constants can only be entered as a numerical constant in an expression to be evaluated by a call to the DPP. This is inconvenient when it is required to enter a large quantity of double precision conctants.

Although I now recognise this to have been an error on my part, it was originally a conscious decision to leave out INPUT and READ facilities as non-essential when trying to minimise the memory requirements of the package. I had intended that INPUT and READ could be implemented in BASIC rather than as part of the M/C package and thus would occupy no space when not required. However, the BASIC routane required is rather longer than I anticipated and it would probably, with hindsight, have been better to include it in the DPP from the start. Still, for those of you who wish to see such a routine, here it is:

```
10 WIDTH 80
20 PRINT "FACILITY FOR INPUT OR READ OF A ";
30 PRINT "DOUBLE PRECISION FLOATING POINT ";
40 PRINT "DECIMAL CONSTANT"
50 PRINT "INTO A VARIABLE."
60 PRINT "GET THE DECIMAL STRING INTO A$ ";
70 PRINT "(INPUT, READ OR WHATEVER) AND GOSUB ";
80 PRINT "2000."
90 PRINT "THE BINARY EQUIVALENT IS RETURNED";
100 PRINT " IN A$ READY FOR USE IN THE DPP."
110 PRINT
120 LINES 37:LIST 2000
2000 REM CONVERT ASCII DECIMAL TO DP BINARY
2010 XP=0:PT=-1:MH=0:LH=0:SN$=""
2020 K=0:D=0
2030 L=LEN(A$)
2040 K=K+1:IF K>L GOTO 2190
2050 DG$=MID$(A$,K,1)
2060 IF DG$=" " GOTO 2120
2070 IF DG$="+" OR DG$="-" GOTO 2160
2080 IF DG$="." GOTO 2110
2090 IF DG$="E" GOTO 2180
2100 GOTO 2040 : REM NEXT CHAR
2110 PT=K-1 : REM NOTE POSITION OF DECIMAL POINT
2120 REM DELETE DECIMAL POINT & BLANKS
2130 A$=LEFT$(A$,K-1)+RIGHT$(A$,L-K)
2140 K=K-1:GOTO 2030
2150 REM SAVE SIGN, DELETE IT AND LEADING BLANKS
2160 SN$=DG$:A$=RIGHT$(A$,L-K):GOTO 2020
2170 REM SPLIT MANTISSA AND EXPONENT
2180 XP=VAL(RIGHT$(A$,L-K)):A$=LEFT$(A$,K-1)
2190 K=K-1:IF PT<0 THEN PT=K
2200 IF K<8 GOTO 2240
2210 IF K>14 THEN K=14:A$=LEFT$(A$,K)
2220 REM LH IS LEAST SIGNIFICANT HALF
2230 LH=VAL(RIGHT$(A$,K-7))/10↑(K-7)
2240 IF K>7 THEN K=7
```

(continued elsewhere!)

Doctor Dark's Diary — Episode 11

Matters arising.

I was delighted to read, in the first issue of 80-Bus News that all articles are to be paid for, although I don't grudge the ten I have churned out for almost free, over the past however long it has been. The thought occurs to me, however, that this encouragement is bound to result in a great increase in the number of items submitted, and if they are better than mine, I shall no longer appear! So, without further ado...

In my last article, I said that my CP/M version of Nas-Sys 1 would only send to disk, and recover, files of up to 16K. This turns out to be nonsense. CP/M is much more clever than I had thought, in some ways, and just opens a new "extent" on the disk when necessary. So even the simple disk operating routines I produced for MONITOR.COM work with large files. The only time a problem could occur is if a disk read or write error occurs whilst reading or writing the first record of an extent other than the first one. If this fairly unlikely event takes place while writing to the file, it will make a mess of the file. It would be possible to write a version of my extra code that could not foul up in this way, if there was any demand for it. So let me know, folks, if you regularly use MONITOR.COM for huge files, and have been getting inexplicably strange results. In case you are wondering if MONITOR.COM is of any practical use, in the light of these horrifying revelations, I am in fact using it now, to produce this article, using my botched version of Naspen (see last article for how to convert Naspen to work to a screen at 0F800H). Of course, if this gets printed, I may be able to afford Diskpen (or is it called Gempen? The adverts don't seem to discriminate between the two at all.) Or maybe a free copy of Gempen will just turn up magically... [Ed. - No chance! And to answer your question, Diskpen runs on a Nascom, under CP/M, using the 48x16 display; Gempen runs under RP/M or CP/M on a Nascom or Gemini with the Gemini GM812 IVC (80x25).]

The address of Aid to Industry Systems, who made the EPROM emulator board I use as a programmable character generator, is:-

4 Dursley Close, Yate, BRISTOL, BS17 4EL

Now you will be able to write to them direct, and save me a spot of postage! I have not seen their advertisement in the glossy magazines recently - I hope they are still in operation, because their board is a useful one, and is reasonably priced as well.

My thanks are due to Dave Hunt, for his answers to my question about using CB to communicate with other Nascoms. I had suspected that it might not be practical, and have been trying to come up with some sort of alternative to using CB. Perhaps it is possible to put the tape interface signal onto the telephone system by means of a small speaker, and a microphone? One thing is certain; no system anyone cares to invent will cost as much as a British Telecom modem does...

I don't suppose any of you are at all surprised to hear that the Pilot interpreter I was writing for use with CP/M fell by the wayside. I became interested in something else for a few weeks, and when I returned to the job I found that I had forgotten how it was supposed to work, even though the source file was full of comments. In the event of there not being letters of protest about this situation, I shall possibly not write any more about Pilot interpreters, although I will certainly not say definitely that the project is abandoned. The language seems to be of no interest at all to most teachers, for whom it was designed, as they are all learning BASIC and using some peculiar machine with an owl on it. The fact that BASIC is not in the least suited to their purposes does not deter them in the least. A volume of the CP/M user library is devoted to Pilot interpreters and the like, if you are still interested, although I have not seen them. My next attempt to write the definitive version will be written in Pascal, I suspect, as this is more sensible than doing it in assembly language. (I was originally going to ignore P. J. Brown's advice, in "Writing Interactive Compilers and Interpreters", on the grounds (or excuse) that being in machine code, my version would run fast. See later for why this is no longer a problem.)

The most surprising thing in the last issue was P. Holy's letter about Hisoft. [Ed. - See 'Letters' in this issue too.] You may remember me waxing ecstatic over their Z80 editor-assembler package for use with CP/M, ("the editor is a joy to use" - shock horror probe!) and saying what good software it is. I have written to them twice, asking fairly difficult questions, and they answered both of my questions promptly. They were very patient when my own carelessness resulted in my being unable to operate the system properly, too! And they put their telephone number at the top of their letters to me... In fact, my confidence in them is such that I ordered their new Pascal 4 compiler for CP/M systems as soon as I saw the advert (which I notice has their Ansaphone number in it) in Personal Computer World. The compiler output is fast machine code, as can be seen by the P. C. W. benchmarks for the earlier version for use with Nas-Sys. And at £40, it is incredibly cheap. A proper review will trip from my Naspen as soon as I have had a good go with the compiler. Unfortunately, I saw the advert too late for this episode's deadline.

Did you see the nice things Uncle Dusty wrote about my item in Micro-Power? The "copyright notice" that MONITOR.COM produces when given the Y command refers only to the part I wrote, not the whole thing. It is there primarily so that people can see how to add their own commands to the system - I'm a great believer in learning by disassembling!

A particularly nasty gremlin.

I recently joined the Taunton Computer Club, which meets at the Somerset College of Art and Technology on Tuesday evenings, from 6pm to 9pm (at which latter time the "serious" members transfer to the staff bar for further learned discussion, or something...) and took Marvin in to give a sort of demonstration. Everything went well for about an hour and a half, then the system went haywire. After a few minutes head-scratching (my head, not the ones in the disk drive, which I clean much more carefully!) and reset pushing, the system settled down and ran for the rest of the evening without further problems. So I assumed it to be mains noise, caused by all the other machines on the same ring (four RM*8*Z's, an A*pl*, and a couple of Si**la*rs, if you must know!) and carried on demonstrating my latest bizarre programs.

At home again, the same problem kept cropping up, and finally I had to do something, because it was driving me daft, and it managed to erase an important file from a disk. So I stripped the system down, and found that the cooling fan had extracted all the chalk dust from the atmosphere of the classroom, and stuck it to the back of the processor board. The lessons here seem to be that if you fit a fan, a filter needs to go in too, and it isn't always the electricity board's fault. Mind you, it often is their fault, as has been pointed out to me recently, in a letter from a member of the Merseyside Nascom Users Group. They (the electricity people, not MNUG!) are rumoured to be in the habit of constantly sending thumping great pulses down the line to operate their remote controlled equipment. Own up, any SWEB employees who are reading this, and feel they can comment on these libels...

Further delusions of grandeur.

Ever since I saw the article in Personal Computer World a while back on how to interface a Z80 and memory to a 6800 based system, I have been thinking about adding more processing power to a Nascom. Not, I hasten to add, because of any lack of power in the basic system. Possibly you will have heard that some programmers have developed multi-programming systems for Nascoms. Their software will run more than one job at once, and I take my hat off to them, in a figurative way. They have done something on a micro that some mainframe people would have us believe can not be done. But is it something that needed to be done? I think (and this is definitely a matter of opinion) that when the system is to run another job, given the cost of a Z80, it is a better idea to add another processor. I picture an add-on board carrying a processor, some memory and a simple control program. The processor, I suppose, will probably have to be a Z80. The main reason for this is that the Z8000 is still too expensive. There are several even more exotic possibilities, such as

bit-slice processors, but I am still reading about them. It remains to be seen whether they too are ruled out by their cost, of course. All this extra hardware is still at the "thinking about it" stage, and should really wait until some minor speed problems on some of my other boards have been fixed. Recommended reading, if you are at all interested in either sixteen bit hardware or bit-slices, is "Modern Microprocessor System Design" by Daniel R. McGlynn, published by Wiley-Interscience and not cheap! Of course, in the event of boards with extra processors appearing, someone will need to write the software to coordinate the tasks they are each running...

Something useful (at last!) for CP/M hackers.

The subroutine below is one that I have found very useful in programs that send a lot of text to the screen. The usual output routine sends all text up to the delimiter, which is a dollar sign, direct to the screen, without any regard to what is happening to the words at the end of each line. It is, of course, possible to write your program in such a way that all the output fits the screen nicely. It also happens to be boring work to do this, and the program will be no good at all on a system with a different screen width. The routine that follows will output the contents of a text buffer of any length, which is terminated with a 00 byte, without breaking any words. The text must not contain new line characters, or the output will be somewhat bizarre, to say the least.

```
SYSTEM  EQU   £0005
PRTBUF  EQU   £09
WIDTH   EQU   48              ;Or 80, or whatever it is.
LF      EQU   £0A             ;Line feed character.
CR      EQU   £0D             ;Carriage return character.
SPACE   EQU   £20
DOLLAR  EQU   £24
OUTPUT  LD    HL      OUTBUF  ;Point to the start of the text.
OUTP02  LD    D       E       ;Set CP/M's text pointer.
        LD    E       L
        LD    B       WIDTH   ;Set B to screen width given.
OUTP04  LD    A       (HL)    ;Get a character from the text.
        OR    A               ;Test for end of text.
        JR    NZ      OUTP08  ;Jump if it is not.
OUTP06  PUSH  HL              ;Save the text pointer.
        CALL  OUTP40          ;Call subroutine to output one line.
        POP   HL              ;Get text pointer back.
        LD    (HL)    £00     ;Remove dollar sign inserted by OUTP40.
        RET                   ;Return to caller.
OUTP08  INC   HL              ;Advance lookahead pointer.
        DJNZ  OUTP04          ;Loop until looked 1 line ahead.
        LD    A       (HL)    ;Check for terminator after line.
        OR    A
        JR    Z       OUTP06  ;Jump if it is.
OUTP10  LD    A       (HL)    ;Read a byte of text.
        CP    SPACE           ;Suitable place for end of line?
        JR    Z       OUTP14  ;Jump if it is.
        DEC   HL              ;Move pointer left.
        INC   B               ;If this not done, no CR LF needed.
        JR    OUTP10          ;Loop round to find space.
OUTP14  CALL  OUTP16          ;Prints the line so far.
        LD    (HL)    SPACE   ;Repair damage!
        INC   HL              ;Advance pointer to next word.
        JR    OUTP02          ;Loop until all text printed.
OUTP40  PUSH  BC              ;Save CR LF flag which is in B.
        LD    (HL)    DOLLAR  ;Insert a CP/M print terminator.
        LD    C       PRTBUF  ;CP/M routine number.
```

```
        CALL  SYSTEM          ;Print the line up to the $.
        POP   BC              ;Get CR LF flag back.
        LD    A       B       ;Transfer it to A.
        OR    A               ;Test to see if it is zero.
        RET   Z               ;Return if it is, no CR LF needed.
        LD    DE      OUTP50  ;Point to CR LF text.
        LD    C       PRTBUF  ;CP/M routine number.
        CALL  SYSTEM          ;Print a CR LF.
        RET                   ;Return to caller.
OUTP50  DEFB  CR, LF, DOLLAR
OUTBUF  DEFM  "Here is a typical line of text which under"
        DEFM  "ordinary circumstances would be printed with"
        DEFM  "the word ORDINARY broken. As you can see, this"
        DEFM  "has not happened."
        DEFB  £00
```

And anybody who thinks that a Nas-Sys version of that routine would be useful is at liberty to write one. I have decided not to, as I have got to get this article finished soon, or it will miss the deadline.

And finally, something totally silly.

This is a conversion of the lost entry to the long forgotten Christmas game contest in BASIC. Probably the best thing about it is the apalling acronym that gives it its name. I insist that the game itself was actually my brother's idea. I think you will find that it is capable of breaking the ice at parties, and could easily be modified to create even more complex situations, that we must not discuss here, as this magazine is frequently read by persons of tender years...

```
1000 PRINT "Social Contact Recreation Under Micro-control"
1010 INPUT "How many players ";NP
1020 IF NP > 2 THEN 1040
1030 PRINT "Don't be silly..." : GOTO 1010
1040 DEF FNR(X) = INT(RND(1)*X)+1
1050 PRINT "Here we go then..."
1060 FOR P = 1 TO NP
1070 GOSUB 5000 : CLS
1080 PRINT : PRINT "Player number ";P;
1090 PRINT " put your ";
1100 GOSUB 6000
1110 PRINT "hand on player" : PRINT "number ";
1120 Q = FNR(NP) : IF Q = P THEN 1120
1130 PRINT Q;"'s ";
1140 GOSUB 6000 : GOSUB 7000 : PRINT "."
1150 NEXT P
1160 GOTO 1060
4990 REM DELAY SUBROUTINE
5000 FOR I = 1 TO 5000 : NEXT I : RETURN
5990 REM PRINT LEFT OR RIGHT
6000 IF FNR(2)=1 THEN PRINT "left "; : RETURN
6010 PRINT "right "; : RETURN
6990 REM PRINT A PART
7000 RESTORE 7500 : B = FNR(4)
7020 FOR I = 1 TO B : READ I$ : NEXT I
7050 PRINT I$; : RETURN
7500 DATA foot, knee, elbow, shoulder
```

I hereby nominate this program de facto winner of the Silliest Misuse of Unusual Technology (SMUT) award for this year, unless you know better, of course...

MORE LETTERS & SOME REPLIES

Disks, C.B. & Amateur Radio

So, the good old INMC is no more! Still, times change and we must change with them. When the club started most users had 1K Nascom 1's and vague aspirations to 16K and Tiny Basic. Now it seems that everyone has a 64K Nascom 2 or equivalent Gemini system with disks (except me!). For a hobby machine the cost of a dual disk systems seems excessive although I recognise that the systems on offer do represent good value for money. How many home, as opposed to business users, really need the speed of disks? Maybe the best way ahead for those of us with limited resources lies with fast mini cassette tapes or the new microfloppies. I have seen no reviews of the mini cassette systems that at least two firms are selling. Have any readers tried these out, and are they practical devices? I will be very interested to try Uncle Clive Sinclair's microfloppies on my Nascom 1. At rumoured price of about £50.00 for 100K, even the four second average head seek time won't put me off!

I was particularly interested in Dave Hunt's article on CB and computers. It does seem to me that he is skating on thin ice here, both legally and ethically. The Home Office definition of CB in the UK is as a 'short range radiotelephone service' and that is surely a reasonable use for it. All the talk about 'burners' and halfwave antennas is just RF megalomania. It reduces what could be a useful public service to a rat race where the 'breaker' with the most power can shout down others until they rush out and fill the dealer's pockets to try and get above the cacophony for a while. I personally agree that the licence restriction on antennas is a shame, particularly directional aerials. A good beam antenna is worth its weight in watts, and owing to its directional properties, doesn't interfere with other local traffic. Incedentally, why do people buy LINEAR amplifiers for CB? One of the many advantages of FM is that the output is constant. This means that the simpler, cheaper and more efficient class C power amp will do just as well.

Dave's idea for frequency-agile CB for data transmission is ingenious, I must say. Presumably, the list of random numbers would have to exclude channels 9 and 19. Also, to add to the already onerous legal problems, modifying the sets as he describes would mean that they would have to be re-certified by the Home Office.

So, is that the end of the story for data transmission by radio? No! The most glaring error in the article is where Dave says 'Amateurs are restricted to Baudot code at a maximum of 50 Baud'. Wrong! Footnote 18 to the Amateur Radio Licence Schedule states 'Data transmission may be used within the frequency bands 144 MHz and above provided (a) the station callsign is announced in morse or telephony at least every 15 minutes and (b) emission is contained within the bandwidth normally used for telephony.'. This means that there is no obstacle to transmitting 300 Baud CUTS format. I should add that amateurs are permitted 100 Watts of carrier power at the antenna and there is no limit (apart from what your neighbours will tolerate!) on the size of antenna that you may use.

The Radio Amateur's Exam (RAE) is not beyond the capability of any Nascom user. After all, most of us have built our own machines and know which end of a soldering iron to hold! Country wide communication on 2M (144 MHz) is commonplace and you have the comfort of knowing that you are not acting illegally or spoiling other people's use of our crowded radio spectrum. Although the equipment is expensive, it's also better made and more versatile. Anyway, you can always build your own gear (unlike CB where rigs have to be approved by the Home Office, amateurs just have to ensure that they are operating within the terms of the licence). Just as a final tempter, the licence is actually £2.00 cheaper!! So Dave, why not throw away your Children's Box [Ed. - I thought CB stood for Chicken Brain!] and do the job properly?

I hope that sets things straight. I don't want to put down Dave's idea. At Busby's current telephone rates it has to be a good thing, but hopping all over the jam-packed 27 MHz band ain't the way to do it!

Keep up the good work and 73's from

Pete Kendall (G6ADF)

DH replies:

Although I agree with almost everything Pete says, I had to come back on this didn't I? I'll comment where I think clarification is necessary on a paragraph by paragraph basis.

Yes, the times they are a'changin', as they say. As was pointed out in the last issue, the content of the magazine reflects the contributions made to it. If the majority of contributors have disks, then the articles reflect that. This is certainly the case at present, and we'd welcome any articles on either mini cassette systems or Uncle Clive's microfloppies (when [Ed. - if] they appear) to redress the balance. If you are using a data loading system which is fast and cheap then let us know. On the subject of sending us articles, can we have them in machine readable form if possible. We don't mind either Nascom or Gemini tape formats, or any IBM 3470 (or is that 3740) compatible disk format, we think we can read most, either 5.25" or 8".

Now on to my article in the last issue 'Breaking Computers'. As I think Peter realised, it was written 'tongue in cheak'. Dave Hunt, him speak with forked tongue, etc. As to the ethics and legality of the suggestions I made, I hope I made it totally clear that they weren't either legal or particularly ethical. Regarding the use of 'linears' when a cheap class C PA would do equally well if not better, it seems most of the linears come from Italy (surprise surprise, not the Far East). The Italians use SSB quite a lot and a linear is necessary for this purpose. You might like to note that it is legal to import them and sell them, it's the owner who breaks the law by using them.

Pete attributes the frequency-agile channel changing idea to me. Sorry, that's not correct, it arose out of a discussion with several interested parties over a year ago, and I don't remember whose idea it actually was. Secondly, contrary to Pete's comment, above, the port controlled channel selector was about the only 'legal' part of suggestions. My copy of MPT1320 dictates the frequencies to be used (spurii, RF power, and a lot of other things), but says nothing as to the physical changing of channels. So it seems to me that computer controlled channel selection is totally in order. A second point stemming from the MPT1320 spec., is that the rigs do not have to be certified by the Home Office. The supplier, importer, constructor or manufacturer simply has to certify that the rigs meet the spec.

When I first read Pete's letter it was the next paragraph which prompted me to write a reply. I also had a couple of other letters on this point. Pete refers to a 'glaring error', now fair's fair, you can NOW send data on 144MHz and above (although, as we shall see, this is not technically open to all Amateur Radio Licence holders). But at the time I wrote that piece, way back in February, the great February 12th debacle was about to break on the amateur radio fraternity to cause many whoops of joy, or wailing and gnashing of teeth, depending upon whether you had an 'Amateur B' or 'Amateur A' Licence. Anyway, up to that date, data transmission as such was a definite 'no go area' as the earlier schedule was then still in force.

I won't dwell on the 'Great Home Office Cock-Up', as a long discussion about radio topics, even about the wrong doings of those who consider themselves our masters and who know better than we mere mortals will 'bore the bytes' off the computer public who are reading this.

However, it's interesting to note that the revised schedule of the 19th March still technically prohibits data transmission (and also, if interpreted strictly, now prohibits RTTY as well) by holders of 'Amateur B' Licences. The classes of emission we are interested in, either automatic on/off keying of a carrier or modulating tone or a.f.s.k. (automatic frequency shift keying) of modulating tones, A2B, F1B, F2B, G1B or G2B, remain the domain of the 'Amateur A' Licence holder as it would appear that our friends at Waterloo Bridge House can't tell the difference between morse telegraphy and data transmission. As the new classifications of modes includes the emission type 'D' this problem could be overcome by including the following modes in the amateur schedules for use by both 'Amateur A and B' Licence holders:

A2D amplitude modulated double sideband a.f.s.k by data
J2D amplitude modulated single sideband suppressed carrier a.f.s.k by data
F2D frequency modulated a.f.s.k. by data

I know the Home Office say that it was not the intention to prohibit data transmission (and RTTY) by 'Amateur B' Licence holders, but on paper they have, blame the loon who wrote the schedule! Perhaps they'll get it right on their third try due in September.

As Pete notes, an aerial input power of 100 Watts and no limitations of the type of aerial used should be adequate for some sort of data transmission country wide. I also agree with his comment that anyone who knows which is the hot end of a soldering iron stands a reasonable chance in passing the RAE exam.

The underlying purpose of my original article was to desuade people from attempting to try data transmission on CB, whilst the recent inclusion of data transmission within the scope of the amateur radio licence (albeit very badly defined) opens up a wholly practical field for experiment. Lastly, sorry Pete, your advice to me to put away my Chicken Brain set comes far too late. I'm still waiting for the HO to perform a bit of 'digitus extractus' in my case (Pete will know what I mean). I can listen to Brum via 'VA' at present, and it should be Ok simplex on SSB when I get my big beam up, so I'll give you a call some time.

More about Amatuer Radio and Non-disks

The following letter on the subject was recieved from C. A. Graham close to the copy date. From the latter half of his letter, it seems he reads the editorals, and has a form of precognition of what Peter Kendall is going to ask.

Radio Data Transmission

I was interested to read Dave Hunt's article on data transmission on the CB channels, but concerned that one or two of the comments made in the article, with reference to the Amateur Transmitting licence, were not quite correct.

It is true that the licence restricts teleprinter operation on the h.f. bands to the CCITT code No.2 (Murray code), at a rate of either 45.5 or 50 baud. However, amateurs are [NOW - D.H.] also allowed to transmit data on all frequency bands above 144 MHz, provided identification (i.e. callsign) is sent in morse or telephony at least once every 15 minutes and that a bandwidth no greater than that used for telephony be occupied.

I have conducted a number of experiments into this form of transmission on channels within the 144-146 MHz band, with Steve, G6BLF, and found that 100% reliable service can be achieved over a distance of 4-5 miles at a transmission rate of 300 baud. The equipment at both ends of the link was a Nascom 2 linked to a 10 watt FM transceiver, in my case, a home built synthesizer controlled rig; and the interfacing required?..... None!! (except a couple of leads and level controls).

The Nascom's audio tape output is coupled to the microphone socket on the transceiver and the loudspeaker output from the latter coupled to the Nascom audio tape input. This arrangement has the advantage that the Nascom thinks it is sending files to and receiving them from a tape recorder; thus programs and data may be sent from within Nas-Sys, BASIC or any other program that uses audio tape I/O. The audio shaping in the transceiver takes care of bandwidth occupancy and with a little optimization might well allow a 1200 baud rate, although this was not achieved in my case - an error rate of approximately 2% being recorded at this speed.

The question of mutual interference with other stations does arise, especially in an area of high activity. However, it is usually possible to find a clear channel (there is even a data-transmission calling channel reserved) and most other stations are content to listen to the proceedings without feeling the need to put a carrier up. If this does happen, another band may be sought; possibly the 70 cm band. This has the advantages of low occupancy and small antenna size at high gain and directivity (you can squirt your signal in one direction, and receive best

from that direction). Antenna sizes are typically those of domestic TV aerials, and consequently most inconspicuous (compare that with your "Silver Rod" lightning conductor!).

Another development on the Amateur Radio scene is the data-handling repeater. This is a land based device which receives data signals on one channel and re-transmits them on another, with a considerable improvement in range (up to 20 miles). One such repeater is GB3MT (Bolton) which will handle ASCII & CCITT No.2 codes and even permit interconversion, i.e. an incoming ASCII (CUTS) transmission being recoded and transmitted in CCITT code and vice-versa. Another facility offered is that of "electronic letterbox". This allows the reception, storage and later retransmission of messages and data. More of these devices are being planned!

Amateur TV

Television transmissions are permitted by the Amateur licence on a number of bands from 432 MHz upwards, and I have made use of the video output from the Nascom 2 to transmit messages and graphics displays on this band over distances of up to 6 miles with 4 watts PEP. A light-pen project is planned shortly to allow onscreen message writing!

Digital Tape Storage

With so much information on floppy disk systems having appeared in the first edition of 80-BUS News (and jolly interesting it was too!), I feel I must take up the challenge thrown down in the editorial, and write something about my (non-disk) system which uses digital tape storage. I decided to go for this form of mass storage about a year ago, when disk storage seemed too expensive (it still does) and under-developed for the Nascom, and a device called CFS appeared on the market, produced by Grange Electronics. This unit is based upon the Philips DCR which takes mini-cassettes of the kind used in pocket dictaphones.

I don't intend to inflate this into a full review of the CFS as I am not even sure that it is still being sold, so, briefly, it provides 96k storage (48k per side) in 24 x 2k blocks + directory block (1 each side) on each mini-cassette tape. The unit plugs into the PIO port on the Nascom 2, and is driven by a 2k operating system called CASSOP (the source listing of which is readily available). This O.S. allows tape formatting, file writing, reading, renaming and deletion, and contains subroutines which are accessible from external software. I have so far successfully interfaced several system programs with it, including NAS-PEN, Xtal Basic, Nascom Basic, ZEAP2 and a data-file handling program, via Nas-Sys.

All my software now resides either on mini-cassette, or in EPROM held on the Gemini EPROM card and "paged" in and out of the memory map by a boot-loader based upon David Parkinson's excellent scheme outlined in INMC80-4. This loader copies Nas-Sys into RAM and then optionally overlays the R,W and V routine addresses in the jump table with addresses of routines in the loader which communicate with the tape operating system. If the overlay is not invoked, normal audio tape I/O (or data transmissions) may take place.

One shortcoming of the CASSOP system is that no "interlock" is provided to query a "Prepare Tape" command, which is accidentally invoked. Thus the directory can be scrubbed in no time flat (certainly before you realize what is going on and hit the reset button!). Since this has happened to me (twice), I have written a 2k tape salvage program which runs under Nas-Sys 3 (or 1 with simple changes) and allows access to the individual blocks on tape, with screen editing of directory, data blocks and read/write/verify etc. I wonder how many other CFS users have had similar problems and could use this program.

All in all, I think the tape system works very well, giving named file facilities and operation under software control, even though it can't match the random access speed of disks; AND it cost about £170 - a lot cheaper than any disk systems around at the moment (although I wonder if the Sinclair Microdrives could be "bent" to work on a Nascom ??).

Bits & P.C.s Basic Toolkit

Just one last item: since getting a 64k RAM card, I have been trying EPROM based software in RAM and much of it works. However, some authors cannot resist putting little "fixes" in their programs to prevent them from running in RAM. Now I may be a bit naive, but I can't for the life of me understand why they bother, unless it is that they reckon EPROM based programs are harder to copy than RAM based programs. Anyway, the Bits & P.C.s Basic toolkit is one of these programs, and there seem to be three such "fixes". To eliminate these, place zeroes (NOPs) in the following locations B020, B021, B23E, B23F, B247, B248. If the program starts on a different 4k boundary, change the B's to the appropriate value.

I hope that some of the information above will be of interest to you and your readers. Good luck with the magazine.

Clive A Graham G3XIG

BASIC Mods. & Microtype Cases

As a user not used to machine code, it may seem dumb to suggest, but, how about having all the readers rewrite the BASIC by offering their optimized Z80 solutions to the comprehensive listings of the 8080 codes available in the Microsoft BASIC? It may well be that this is what Crystal has already done, but knowing how well some of my better informed friends write in M-C, I have no doubt that there has to be a better (or even best) solution to every one of the seven hundred routines of the original Nascom BASIC, as well as to the forty-odd additional general and DOS routines which I have seen added since.

If this were undertaken by a group of readers I feel sure that this would be one of the largest and most interesting projects in microcomputing undertaken in recent years.

For those of us with Microtype cases for Nascoms 1 & 2, could I suggest that someone makes an 'extension ring' to fit between the upper and lower halves. Together with additional connectors there seems no reason why perhaps five or six cards can not be fitted - the fan is quite large enough to cope, the only limitation being the power supply which I believe could be uprated rather than replaced.

Bert Martin

DH back again.

Yes, a nice idea, although I doubt that the whole readership would either feel competent nor want to take part. This is the sort of thing that could easily be handled within a group. It sounds like the sort of thing that NAS-TUG, the Nascom Thames Valley User Group would dearly love to have a go at (how about it Mike?). I might add that Carl Lloyd-Parker is something like half way through the job of tearing the BASIC apart, converting it to Z80 mnemonics, commenting and labelling it, so far he has 170K of commented source, and he says that's less than half of it. Personally I doubt that the finished code would be much if any shorter as the Microsoft BASIC is the product of many man-years work and must be fairly well optimized. Optimization to Z80 codes alone does not save all that much space. If any readers feel like having a go, drop us a line. If we get more than one reply we'll put you all in touch.

The dear old Microtype case is fairly easily adapted and the idea of producing a 'skirt' to fit between the top and bottom halves is a good one, if not original. A couple of years ago I remember seeing one with the top and bottom halves separated by 2" pillars and the resulting gap filled with fine black plastic mesh apparently bought from an ironmonger.

MONITOR.COM and other scribblings

by Jeremy Gugenheim

Well me dearios, at last I have put Nas-Pen to printer, and what goodies flow forth?? Read on, and if you have disks, CP/M, and MONITOR.COM (the best thing since sliced Nascoms) all your old Nas-Sys software can burst back into fruity life!! Firstly, one and a half mods to MONITOR.COM itself (Gasp!), this one can only be done if your MONITOR.COM came from Nas-Sys 1 (Tee hee!).

Do 'DDT MONITOR.COM', then 'S2F5' and replace

```
79 DF 60 EF 08 08 0D 00
```

with

```
EF 2E 00 79 DF 68 DF 6A
```

This gives you back your checksums in the Tabulate routine, separated from your bytes proper by a space and a full stop.

To change the cursor character, do your DDT etc etc, and change the byte at 0877H, which should start out as 5FH. It is quite alright to replace it with a carriage control character, they won't control the carriage, try 07H, or even B5H if you've got the graphics chip.

Something for all you dedicated Nas-Crunchers (compatible) is replacing the Y (copyright) command with the repeat keyboard routine from INMC-6 (compatible). You can't type it in directly so you need to find some mug who is willing to work out the new addresses for you. If you are too lazy even for that, then, well, I've done it for you.

Do all your DDT and stuff, and 'SA67' this lot in...

```
0A67 21 C3 0A DF 72 21 80 0A
0A6F 22 7B 0C 21 80 02 22 2E
0A77 0C 21 50 00 22 30 0C C9
0A7F 00 DF 61 30 07 2A 2E 0C
0A87 22 2C 0C C9 2A 2C 0C 2B
0A8F 22 2C 0C 7C B5 C0 21 02
0A97 0C 01 00 08 16 FF 7D FE
0A9F 06 20 02 16 BF FE 09 20
0AA7 02 16 C7 7E A2 28 06 0E
0AAF 01 7A 2F A6 77 23 10 E4
0AB7 79 B7 C8 2A 30 0C 22 2C
0ABF 0C DF 61 C9 76 70 00 .
```

For those of you who just groaned because I'm too mean to type in the source, I'm not, it's exactly the same as the INMC-6 (compatible) one with the addresses changed and a RET NOP replacing the original SCAL MRET. So, now the clever ones amongst you can work out how, by using the Y0 and Y1 commands, it is possible to turn the repeat on and off. You have 55 bytes to play with, that should be ample, or you could even make it execute automatically after a cold start, but note that after the workspace has been initialised.

Once you've MONITOR.COM installed, use the READ command to get ZEAP into your memory. DON'T let the GENERATE command have its evil way here, or you'll really be in trouble, 'cos MONITOR.COM uses the old screen to hold bits of CP/M that it requires for disk transfers. If you have Generated ZEAP then you'll have to disable the Generate command. How do you do that? Simple, you can even use Nas-Sys for this one.

Use Modify to change the two bytes at 0621H

from	21 7A
to	DF 5B

I'm sure you can guess what that does, and if you can't, you can look it up in the Nas-Sys manual under 'how to end a program'. Excuse the digression. Once you have Zeap in your memory use the Modify command on the following:

	Address	Was	change to
	1849	F4	04
and at	15EB, 1644, 1853, 185B, 1B36, 1B94, 1BFD, 1C0E, 1C17, 1C20, 1C2C		
	1C82, 1CBF, 1CC8, 1CD2, 1DE2, 1E20, 1E30, 1E40, 1E73, 1EE7, 1FBA		
	change	0B	FB

All the above changes move the screen to the place CP/M expects to find it except 1849, which took rather a while to find ... it's used to locate the cursor on the screen, and is the 2's complement of the number they actually mean...

I'm now working on disk save/get routines from within ZEAP, though where I'll put them I don't know; probably I'll remove the +/- option adjust. that gives me the two command letters I require, and a bit of room inside ZEAP. I guess one of the routines will have to appear at 2000H, so I'll have to move the start of text to allow for that. Full marks to the writers of ZEAP for making that bit easy, if nothing else. By the way, all this surgery upsets the routine that checks that all's well within ZEAP, so you'll get Error 90 every time you do a cold start. The way around that is to find a unused byte, such as 1FFDH (I hope) and fiddle with it till ZEAP starts OK.

I've also got Nas-Pen (compatible) working (guess what I wrote this on) under CP/M but I can't for the life of me remember how I did it. Still, someone else has done it, and documented it too, so find that and the world is yours.

PiP PiP Chaps.

INPUTting and READing Double Precision Constants (continued)

```
2250 REM MH IS MOST SIGNIFICANT HALF
2260 MH=VAL(LEFT$(A$,K))
2270 XP=XP-K+PT : REM ADJUST EXPONENT
2280 U=USR(DP)MH+LH&A$ : REM MANTISSA
2290 IF SN$="-" THEN U=USR(DP)-A$&A$ : REM SIGN
2300 REM NOW FLOAT IT
2310 IF XP=0 THEN RETURN
2320 IF XP<0 GOTO 2350
2330 FOR I=1 TO XP:U=USR(DP)A$*10&A$:NEXT I
2340 RETURN
2350 FOR I=1 TO ABS(XP):U=USR(DP)A$/10&A$:NEXT I
2360 RETURN
```

-- o0o --

THE KIDDIES GUIDE TO Z80 ASSEMBLER PROGRAMMING

by D. R. Hunt

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

Part: The Sixth

Getting stuck.
Nobody's perfect, we all make mistakes don't we!

Well it had to happen, four episodes and all I get is fan mail, I had to go and blow it on the fifth now didn't I. Yet, strange, only two letters and one phone call to tell me. Either no one out there is reading this rubbish, or no one out there understands it, or you are all so poor (having taken out a second mortgage to buy the darned thing in the first place), you can't afford a stamp. Well, having been caught by the way B2HEX worked half way through writing the last episode, I wasn't thorough enough in checking the thing myself. When making use of tricks to do useful things in programs, I said you needed to know exactly how an instruction worked, and then proceeded to waffle on about how the DAA instruction went about its 'doings'. Got it wrong didn't I?. Made it up from what I thought it did, didn't I? Didn't look in the Zilog bible, did I? Oh what the heck, I made a dogs' breakfast of it and now I've proved what I said in the first episode, that I'm not qualified to write this stuff anyway. Call that an excuse? Yeah!! Want to make something of it?

So it's sackcloth and ashes time, I consulted the Zilog bible (not the little one in the Nascom manual, the big fat 'Zilog Programming Manual'), and intoned one hundred times as a penance, "The DAA instruction works as follows .....". Now I must impart the truth and explain exactly how it does work. Fortunately (for me) the remainder of my description of the B2HEX routine is correct, even down to the introduction of the ADC 4OH as a 'fiddle factor'. It was only the way the DAA dealt with it which was wrong.

So here goes. Firstly, the invisible 'Half Carry' flag is not affected by the DAA instruction itself, in fact, it is the preceding arithmetic instruction which sets the H flag for the DAA instruction to use. The instructions which affect the H flag are ADD, ADC, INC, SUB, SBC, DEC and NEG. The DAA instruction works conditionally in the following manner:

Preceding operation	Result of preceding operation in A and F				Action taken and result of using DAA	
	Condition of C flag before DAA	HEX value in upper digit (bits 4, 5,6,7)	Condition of H flag before DAA	HEX value in lower digit (bits 0, 1,2,3)	Number added to A by DAA	Condition of C flag after DAA
	0	0 - 9	0	0 - 9	00	0
	0	0 - 8	0	A - F	06	0
	0	0 - 9	1	0 - 3	06	0
ADD	0	A - F	0	0 - 9	60	1
ADC	0	9 - F	0	A - F	66	1
INC	0	A - F	1	0 - 3	66	1
	1	0 - 2	0	0 - 9	60	1
	1	0 - 2	0	A - F	66	1
	1	0 - 3	1	0 - 3	66	1
SUB	0	0 - 9	0	0 - 9	00	0
SBC	0	0 - 8	1	6 - F	FA	0
DEC	1	7 - F	0	0 - 9	AO	1
NEG	1	6 - F	1	6 - F	9A	1

Now we can see the affects of the DAA instruction, or at least it will become clear if you can unscramble the above table. Remember, the DAA instruction is to enable the use of packed BCD arithmetic. All right, what is packed BCD arithmetic. Well you all know we've got eight bits available in the accumulator, and to date we've been plugging it with data in HEX, two digits at a time, the characters 0 thro' F. Now BCD stands for Binary Coded Decimal, and uses four bits to represent a decimal number, 0 thro' 9. As we've got eight bits in the accumulator we can accommodate two decimal numbers, giving a decimal number range of 0 thro' 99. The packed bit in 'packed BCD' simply means that there is more than one digit.

OK, so let's add two numbers in decimal:

```
 15
+27
---
 42
```

Now lets add the packed binary representations of the two numbers

```
 0001 0101        see that 0001 is the 1 and 0101 is the 5
+0010 0111
----------
 0011 1100        Something went wrong, the result is 3C
```

Well it's obvious what has happened, the numbers were added in pure binary and not in packed BCD, now the Z80 doesn't have an 'ADD packed BCD' instruction, but the DAA instruction is provided to affect the result in such a way that the effect of having an 'ADD packed BCD' instruction IS provided. The reason for doing it this way is because only one 'correction' instruction is required for seven operations, whereas seven additional instructions would be needed in the Z80 if this were to be implemented directly. Add up the numbers, low nibble first, 0101 + 0111 = 1100 = C, notice there is no 'carry' into the high nibble, so the invisible 'half carry' flag, H, is not set. By the same process, 00001 + 0010 = 0011 = 3, again, there is no carry, so the C flag wasn't set. The result is 3C with neither the H or C flags set. Now to determine what the DAA instruction will do with it. By inspection, it looks as if the second row in the table satisfies these conditions, the upper digit is between 0 and 8 and there was no C, the H flag is 0 and the lower digit is between A and F. So the table says the DAA instruction will add 06 to the 3C in the accumulator and there will be no carry. Let's try it:

```
 0011 1100
+0000 0110
----------
 0100 0010    Well the result is now 42 which is what we wanted.
```

Got it, w - e - l - l, I know it's difficult, but if you indulge in a bit of practice you'll soon see how it works. Go on try a few random two digit numbers and see if you can work it out from the table. Anyway, if you relate the above to the previous episode you'll see how it all falls into place. As I said then, the trick is really knowing how the instructions work, and making use of what is provided.

So, having put the lid on the DAA instruction and how B2HEX works, onto the subject of todays lecture on the painful art. A thought has just passed its way through my feeble brain! If you've been waiting for me to get on with the job of explaining this opaque business, and have been moving towards the light at the rate at which I write this stuff, aren't you fed up with it yet? If not, there are still some masochists out there!

I thought perhaps describing the bones of writing a games programme wouldn't be a bad idea this time. 'Battleships' I thought. But then at the time of writing (late May), I thought that might not be too appropriate. Then I thought of a new game 'Bomb the Argies', but couldn't quite work out the rules, in fact there didn't seem to be any. Someone might accuse me of being tasteless (same as the writers of the game 'Three Mile Island' were likewise abused after their game came on the market three weeks after a certain nuclear power station nearly blew up. Must admit it was quite good, particularly the colour graphics of the glowing reactor, played on an Apple by the way.). So we need a program which will be short, with well defined rules and totally uncontentious, taking these criteria into account, it's easy to deduce that it will probably also be extremely boring.

Right back in the mists of time, shortly after I acquired a Nascom 1, I remember writing a machine code version of Hangman in the hope that I might justify the time I was devoting to the machine by entertaining No. 1 Brat, at that time aged about seven or eight. Now Hangman's hardly contentious, the only people that program could upset is the anti-capital punishment lobby, and then only figuratively. So I rummaged about for it, I found a Nascom 1 format tape which I couldn't read, but no sign of the source. It then occurred to me that it was written in pre-assembler days, and if there ever was a source, it would have been handwritten and not machine readable anyway. So I might as well write a new one. So 'Simple Hangman' saw the light of day last evening. It's printed somewhere in the mag. It's not buried in this article so that whoever does the paste up on this issue isn't going to get a headache trying to fit a monumental chunk of DH's stuff in one place. Apart from that, it runs stand alone, so anyone wishing to use it can do so. Rewriting Hangman has served a secondary purpose, as it has entertained No. 2 Brat, aged about seven and a half, over the last few days. That also explains the choice of words in the word table. I offer this explanation, just in case someone decided to draw Freudian conclusions about me from the word table. Anyway, If anyone wanted to draw conclusions, they'd do better reading this rubbish. Don't write to tell me your findings, I know, I know!!!

Ok then, let's think about the special parameters required in writing a program like this. Now the first one is that people reading this article will have different systems, notably, Nascom's running NAS-BUG 1, 2 or 4 (heaven forbid), Nascom's running NAS-SYS 1 or 3, Gemini's running RP/M, or Nascom's or Gemini's running CP/M, or some other combination I haven't thought of. To complete the 'foul-up', I'm running a homebrew mixture which features a Nascom 2 with Gemini CP/M and a number of other assorted odds and sods. Now that adds up to a lot of system incompatibility one way or another. How to get round that one? Well, the only areas that need concern us are where the systems are different: the memory maps, the input and output of characters and the way the program is exited. If I provide 'user patch areas', for the input/output and exit areas, and direct all input/output to the program via these patches then that problem is solved. If I assemble the whole lot starting at 0C83H and provide a 'start jump' into the program at either 100H (for RP/M - CP/M) or at 0C80H (for NAS-BUG or NAS-SYS), then the memory mapping problems are also overcome. True under a CP/M - RP/M environment it wastes nearly 3K of space, but we can't have everything. Another criteria regards memory mapping is that the program shouldn't be larger than OFFFH, otherwise it won't fit on a minimum system Nascom 1 or 2. Of course, the real reason for providing these facilities is that it allows me to be lazy and not have to provide half a dozen different versions.

To make life easier, I'm going to concentrate on only two 'generic' versions. A version which will run under NAS-SYS 1 or 3, which I will call the 'NAS version' from now on, and a version which will work under CP/M - RP/M, which I will call the 'CP/M version' from now on. Differences for the NAS-BUG version simply require the input patch to be changed, such that it saves all the registers, CALLs $KBD, CALLs $CRT, then restores all the registers, whilst the output patch again saves all the registers, CALLs $CRT then restores the registers. The exit should be either an absolute jump to PARSE (but watch out as this does not reset the monitor stack), or an absolute jump to 0000H to reset the system.

So let's look at the patch areas. The first patch, (although it's not obviously so) is the start jump. For either NAS or CP/M versions the three bytes will be the same, C3 ODD3, but for the CP/M version, this jump will be located at 100H, whilst the NAS version will have the jump at OC80H. (Don't forget that my assembler prints absolute addresses the 'right way round' and not 'low byte first' as they would actually require to be loaded, so the above would be loaded as C3 D3 OD.) The next patch area is the one concerned with getting an input. As printed, it contains the CP/M version. For the NAS version this becomes a system call to 'BLINK'. However, unlike CP/M function 1 (the input function), the keyboard input is not automatically echoed to the display, so the call to 'BLINK' is followed by a system restart to 'ROUT'. Now 'BLINK' corrupts HL and DE, and my notes about the patches say that all registers must be preserved, so HL and DE must be 'PUSHed' before the system calls and 'POPped' afterwards. The next patch is the 'output to display', with the NAS version, this couldn't be simpler, a system restart to 'ROUT', as 'ROUT' also kindly preserves all the registers there is no need to PUSH and POP them, so this patch is all of two bytes long and 14 NOPs to fill the empty space. The last patch is also dead easy, a system call to 'MRET' and a NOP to fill the one remaining empty space. So having patched the area it should look like this:

```
OC80    C3 ODD3             JP START        ; Skip round patches and the text

OC83    F5          GETCHR: PUSH HL         ; Save registers
OC84    D5                  PUSH DE
OC85    DF                  RST SCAL        ; Internal subroutine call
OC86    7B                  DEFB BLINK      ; Keyboard entry routine
OC87    F7                  RST ROUT        ; Display the character
OC88    D1                  POP DE          ; Restore registers
OC89    F1                  POP HL
OC8A    C9                  RET
OC8B    00 00 00 00         DEFB 0,0,0,0    ; Pad to 16 spaces
OC8F    00 00 00 00         DEFB 0,0,0,0

OC93    F7          OUTCHR: RST ROUT        ; Display a character
OC94    C9                  RET
OC95    00 00 00 00         DEFB 0,0,0,0    ; Pad to 16 spaces
OC99    00 00 00 00         DEFB 0,0,0,0
OC9D    00 00 00 00         DEFB 0,0,0,0
OCA1    00 00               DEFB 0,0

OCA3    DF          EXIT:   RST SCAL        ; Internal subroutine call
OCA4    5B                  DEFB MRET       ; Return to NAS-SYS
OCA5    00                  DEFB 0          ; Pad to 3 spaces
```

Good, getting there, now there are a couple of others before we go any further. The CP/M version uses code 1AH to clear the screen, and the line feed following a carriage return, code OAH, is explicit and not implied as it is in the NAS version. Well, the 'clear screen' character, labelled CS only appears once at OCD8H and this is easily changed to OCH for the NAS version. The other one, the line feed, again appears only once at ODD1H and this should be changed to OOH. By the way I haven't tried this program under the NAS regime, so if it doesn't work, drop me a line with the correct answer, to collect your 'Prize Dodo of the Month' medal.

So on to the program. We all know the rules of 'Hangman' don't we, so there's no need to reiterate them here. I'm one of those people who isn't into flow charts, so I'm not going to provide one, the program flows in what I would call a linear fashion, executing in a straight line, skipping the bits that aren't required as determined by the conditions set from the previous operation.

Program flow is as follows:

1) Put up the title
2) Initialize the 'letters tried' buffer and the 'trys' counter
3) Throw a random number from 1 to the maximum number of words
4) Locate that word and copy it to a 'word buffer'
5) Display the 'I've chosen a word' message
6) Test to see if the maximum number of trys has taken place
6a) If not the maximum number of trys, go on to 7
6b) If so, display the 'lose' message
6c) Display the word
6d) Display the 'another go' message
6e) Get an input, validate it and restart or exit as appropriate
7) Display the 'what letter' message and get a letter
8) Check the 'letters tried' buffer
8a) If not in the 'letters tried' buffer, go on to 9
8b) Display the 'letter tried' message then back to 6
9) Put the letter in the 'letters tried' buffer
10) Count this try
11) Scan through the word to see if the letter's there
11a) If not found go on to 12
11b) If found flag it by making bit 7 high (adding 80H)
12) Test the word to see if all the letters now flagged
12a) If not all flagged, then on to 13
12b) Display 'you've won' message, then back to 6c
13) Display the 'trys' message, and the number of trys so far
14) Display the word, flagged characters as letters, unflagged as '-'
15) Display two newlines then back to 6

As you can see the philosophy is simple, and presents no problems to the programmer.

Now onto the bits in detail, from the above, it's obvious we need three workspaces, one to hold the word, one to hold the letters tried and one to hold the number of trys taken. We also need two others, a three byte workspace for the RANDOM routine (which must be primed with three numbers, any numbers), and space for the program stack.

In order then, the workspace 'RING' is the workspace for the RANDOM routine, it is primed with the numbers 01H, 02H and 03H, although so long as this work space contained any old rubbish it wouldn't matter (it's random after all). The only thing the RANDOM routine doesn't like is if all three workspace bytes are 00H.

WBUF is the 'word buffer', this must be one byte longer than the longest word it is expected to hold, so this is what limits the word length to 9 characters, and subsequently allows me to indulge in a nice fiddle of which more anon.

TRYS is the one byte store where we keep count of the number of trys taken. To simplify matters, the contents will be in packed BCD, as we are hardly likely to indulge in more than 99 trys.

Next comes the 'letters tried' buffer, CHRTRD, which will be the maximum number of trys plus one long. Now in the system equates I defined the maximum number of trys, NTRYS, as being 12H (HEX, as the comparison will be done in packed BCD, 12H is the representation of 12 in packed BCD), therefore if we make the length of CHRTRD equal to NTRYS in HEX, it's got to be longer than the maximum number of trys without us having to worry exactly how long it is, let the assembler take the strain, that's what I say.

The last work space is the stack space. This started at 30H bytes long. Having written the program I then loaded it under the debugging tool, filled the stack space with 00's and ran the game some half a dozen times, trying all combinations of winning and losing. Having done that, I stopped the game and examined the stack space to see how many of my 00's had been overwritten. Eighteen bytes had been overwritten, indicating a stack depth of 9 (as each stack operation takes two bytes) so I set the stack space to 18 (12H). As I haven't tried the NAS version I don't know if it requires more stack space, although it's unlikely. Anyway I've deliberately placed the stack next to CHRTRD, and that will always have some spare space in the end of it, so the stack can probably come down something like 24 bytes without crashing into the used part of CHRTRD.

Next comes the messages. I've had to write my own string output routine as I am can't use the string displaying facilities of either NAS or CP/M because I've restricted my output patch to single character by character operation. My string print routine is a simple little subroutine buried somewhere towards the end of the program and labelled SNDTXT. The way this works is to 'point' HL at the start of a string of text and call SNDTXT. SNDTXT then marches through the text byte by byte directing it out through the output patch. When SNDTXT encounters a 00 it stops and returns to the next instruction in the main program. The choice of 00 as a text delimiter was deliberate, as it is very easy to test for. Having loaded a character into A, A is ORed with itself, now if this is anything other that 00, the character is unchanged and the Z flag is reset. If it is 00, then the Z flag is set. In fact I've used this scheme of using 00 as a delimiter throughout the program, for marking the end of CHRTRD and WBUF. You might like to note that ORing A with itself will also always clear the C flag. Handy thing to remember if the C flag starts getting in the way of some arithmetic operation at any time. Anyway, using 00 as a delimiter makes life very easy.

You might have noticed that the messages do not contain any 'newlines' within them, although MSG1 and MSG2 could easily have been run together in this fashion. This is because the NAS version requires only ODH to be sent, whilst the CP/M version requires ODH OAH. By making the 'newline' a small message in its own right, it is easily patched. The penalty is that the program 'grows' bit, as additional calls have to be made to display the 'newlines'. To rationalize the additional calls to send newlines I've written pair of 'nested' subroutines called CRLF1 and CRLF2. CRLF1 simply points HL to the newline message then calls SNDTXT, hence one newline is sent. CRLF2 calls CRLF1 and then drops through to CRLF1, hence two newlines are sent.

The next bit is obvious, set the stack, then on to initializing the workspaces. XORing A with itself will always result in A being 00 and the Z flag being set, a simple one byte method of clearing A. The 00 thus generated is put in TRYS, to set it to 0 and into the first space in CHRTRD, because it is empty and 00 is the delimiter. Then comes the displaying of the title using SNDTXT bit. I require two newlines after the title, so CRLF2 is called.

Now comes the picking of a word from the table of words. The first thing is to pick a random number. A subroutine is used to do this (even though it's only used once within the loop) as this is a stand alone module. All it needs to know is where the three byte workspace RING is located, and for A to contain the maximum number (n) on entry to the subroutine. It returns A containing a random number between 0 and n-1. It's a rather complicated routine and I'm going to gloss over it until some later episode when I've worked out what makes it tick. Having chosen a random number, the next thing is to locate the word picked, from the start of the word table. This is done by putting the number in B, pointing HL to the start of the words and testing them byte for byte. If the character picked up is a letter then it is ignored. Where the character is an 00, then the little test mentioned above detects this and the character is counted by the DJNZ instruction. Effectively, this counts the 00 word delimiters downwards from n to 0 in B.

By having counted down to 0, the loop leaves HL pointing at the start of the word in question. DE is then loaded to point to WBUF, and the characters are copied from the location pointed to by HL into the location pointed to by DE. HL and DE are both incremented to point to the next character in the word table and WBUF resepctively, whilst B is incremented to count the letters. Notice the copy process is so arranged that the 00 word delimiter gets copied into WBUF along with the word. That saves me having to make the deliberate effort to put an 00 at the end of the word in WBUF. By copying the 00 across to WBUF, B now contains a count one count greater than the length of the word. So long as we remember this that's ok.

Now to the cheeky bit. We want to tell people how many characters there are in the word (remember we've got the count plus one in B). Now as the word length cannot exceed 9, dictated by the length of WBUF and hopefully no-one has added any words longer than nine characters, and as I suspect that the program might get upset if we set up a word of no length, consisting of 00 only, the word length cannot be less than 1, this means we have a figure between 02H and 0AH in B. If we add 2FH to this, we end up with the ASCII character for the numbers 1 to 9. So, what about stuffing this value into the text string that is to display it. Well what about it? Now I'm sure I've mentioned (and if not, I'm about to) that writing code that alters itself is bad practice, if only for the reason that you can't put it into EPROM (there are other reasons, but they're a bit deep). So how do I defend myself for suggesting naughty practices to you. Simple I don't, I say tut - tut, shake my head and look the other way. Who the hell wants Hangman in EPROM anyway. So the next bit does just that and displays the message.

Here starteth the main loop. This is where we test to see if the trys equals the maximum set and send a 'lose' message, otherwise the program goes on to get an input character, decide if it's been used before, and if not, to compare and fill in the blanks in the word. Sounds simple, and it really is.

Ok, so compare the number of trys taken (from TRYS) with NTRYS. If it's not there yet, the program skips past the 'lose' bit and carries on. Have you noticed my use of uninspired labels for the main program flow. Simply CONT and LOOP. Sorry, but having said labels should be meaningful, there's not much else you can do in a linear flow program. Anyway, if the number of trys equals NTRYS, display the 'sorry you lose' message. Now here we have a label RSTART, which is where the program restarts in the event of a win. Note that the program goes through this bit, win or lose. This displays the word, but we can't simply point HL at it and call SNDTXT as it will have some or all bit 7's set, which will display funny graphics characters. Instead, LOOP4 gets the characters one by one (up to an 00, as before) these are ANDed with 7FH which is a bit mask to strip the flag bit (see last episode) and then sends them to OUTCHR one by one.

In the event of a win or a lose, the next thing is to discover if the game is to continue. The appropriate message is displayed and the input routine, GETCHR, is called. When a character is returned (caused by you hitting a key), the character is compared with 'Y', if it is that character and only that character (no mucking about with lower case here) then the program goes right back to the beginning and starts all over (no finesse about that bit either). Any character other than 'Y' will cause the program to jump to the EXIT routine and finish the game.

If the number of trys doesn't equal NTRYS, then we must display a message inviting an input character, and then call GETCHR to get an input. Having got the input character, the character is compared with 03H (which just happens to be control/C, the CP/M escape character. For the NAS version the byte at 0E55H could be changed to 1BH to change the tested into and 'ESCape' character to be consistent with other 'NAS type' programs.). If the character is 03H (or whatever) then the program jumps to the EXIT routine to finish the game. Otherwise the character is taken as valid, so a newline must be sent. Now the CRLF1 subroutine will corrupt both the AF

and the HL registers, HL doesn't matter at this point, but A contains the character, so as we need the character in the B register for the following part of the program we might as well tuck it out of sight now before the newline is displayed.

Now to find out if the character has been used before, and this is done by scanning through the used letters buffer CHRTRD. Remember the letter is already in B. The characters are picked up one by one into A and compared with B until either an 00 is detected indicating the end of the buffer or a compare is found. If a compare is found a message to announce that fact is displayed, and the program goes back to the start of the main loop to get another character.

If the character isn't found in CHRTRD, then it must be put there, the scanning routine was so arranged to leave HL pointing at what was the buffer delimiter, the 00. There is a LD (HL),B instruction, but when I wrote the program I forgot it existed, so I copied B into A again and used LD (HL),A to copy it into CHRTRD. Silly of me. This sort of oversight becomes apparent when you read through the listing. However, having already edited the listing into DISKPEN printable format, it's too late to do anything about that now. I dare say there are other instances where I've wasted a couple of bytes in this fashion. I'll offer a £5.00 prize for the maximum number of 'oversights' of this sort found in the program. Don't cheat just 'cos there's some loot in it. I'm sure there are much more efficient and space saving (if less transparent) methods of writing this program. I'll be the judge of how much code juggling you are allowed to indulge in to save a few bytes. Anyway, having put the character in CHRTRD, HL is incremented by one and a new 00 placed at the end of the buffer to delimit it. Now I hope you see why we put an 00 at the start of the buffer when the program started.

The next step, as we've accepted the letter is to count it. This is done by getting the contents of TRYS into A, incrementing A by one, performing our old friend DAA on it to keep a packed BCD number, and putting the new number back in TRYS.

Now to look for the letter in the word. The scan procedure is almost identical to the previous one. If a compare with B is found then the letter is 'flagged' to make it visible. This is done by setting bit 7. There are a number of ways to do this, I favour the simplest to understand which is to add 80H to the letter. Having flagged the letter, it is put back in it's place in the word.

Having flagged (or not flagged) the letter it's time for another scan, this time to see if all the letters in the word are flagged. This works in the same way as the previous two scans except in this instance the letters are compared with 80H and the result of this decides whether it is flagged or not. The compare instruction is effectively a subtract instruction which does not affect the contents of the A register, so if the character is less than 80H, that is, unflagged, there will be a carry. If the character is equal to or greater than 80H then there will be no carry, indicating that the character is flagged. As soon as a character that is less that 80H is detected the old trick of ORing A is performed. If there is no Z, then the character must be a letter, in which case there is at least one letter remaining which is unflagged. If the OR A results in a Z then the character must be 00 indicating the end of the buffer, which means all the letters have been flagged, and therefore you win. Think about this step carefully because this is where the decisions are taken.

In the winning case a message is displayed to say 'you win' and the program jumps back to RSTART to display the word and ask if you want to continue.

In any other instance, the 'number of trys' message is displayed followed by the number of trys from TRYS. Now as TRYS is a packed BCD number, ideal for display by our old friend B2HEX. So just to prove it has it's uses in programs it has been included for the sole purpose of displaying TRYS. Notice the output part of B2HEX has been altered slightly to accomodate OUTCHR. Having displayed TRYS, the next thing is a

number of spaces, these are contained in a message and displayed in the normal way through SNDTXT. Now to display the word, this is done in a similar fashion to the one already described in LOOP4. The only difference is that each character is tested to see if it is flagged. Each flagged character is 'visible' so it is ANDed with 7FH to strip the flag and then displayed directly, each unflagged character is replaced by a '-' which is displayed instead. A space is placed between each character. When 00 is detected, the loop cops out in the normal way, and a call made CRLF2 to send two newlines. After that the program jumps back to LOOP3 to start the whole process again.

Well that just about wraps up that program, with the above comments and the liberal sprinkling of comments all over the program itself, you should be able to follow it. Now to improvements, well obviously if you have an assembler you can reassemble the whole thing into it's right places for either NAS or CP/M versions. If you're going to have a go at the NAS version, then might I suggest that you put the ORG at 1000H as you could unwittingly end up with problems if you extended the words table from its current end at OFFCH. You see, NAS-SYS corrupts the bytes OFFEH and OFFFH during its 'E'xecute procedure as it uses them as intermediate stack space, so if you had data in those two bytes then some funny things could happen. There is no easy way of making the program avoid those bytes so best to reassemble the whole lot above it. If you've got an assembler it implies that you aren't stuck for RAM space, so what the odds. The words table may be extended to 255 words (the largest random number that can be generated), make sure NOWRDS contains a count of how many words there actually are. There is scope for tidying up the displays although the method used here was chosen so I didn't have complications with the output routine. In fact it was deliberately made a simple as possible. I remember my handwritten version kept the screen in one place and didn't scroll. It achieved this by moving the cursor about and overwriting the text for each successive turn round the loop. There is one version in Basic I've seen which incorporates some spectacularly gruesome graphics, complete with swinging man and stretchy rope. Lot's more fun to do in assembler. People reassembling this for a NAS regime could save quite a bit of space by dispensing with the input/output patches and calling B2HEX direct from NAS-SYS. As should be apparent B2HEX had to be included in the program as there is no CP/M equivalent within the system.

I hope that you've noticed that by feeding all input and output through simple user patch areas this program could be made to work with very little modification on almost any Z80 based machine. It could also be modified fairly easily to run an any 8080 based machine simply by substituting absolute jumps for the relative jumps I've used (and modifying the one or two DJNZ loops used). This technique of providing user patches is going to play an ever more important part in programs published in this mag due to the diversity of machines we intend to support in future. So if you are thinking of writing a program for inclusion in the library then keep in mind that flexibility is the name of the game, and user patches will become very useful.

Note that the RANDOM routine works completely stand alone, all it needs is its three byte work space and to be fed with the maximum number to pick. I'm told this particular routine is very nicely random, quite a bit better than the ones found in a good many Basics.

So here endeth the lesson, as I said, I haven't tried it under a NAS regime, but it should work. I know it works under CP/M, and as it only uses two system calls it has no excuse for failing under RP/M. The 3K waste space penalty is 'a nothing' when used with disks, for the RP/M tape types, sorry, but it takes about 45 seconds to load when it should only take 4 or 5. I hope you will have a go at it, and that it gives you as much fun getting it to work as it gave me in writing it.

```
                        TITLE SIMPLE HANGMAN

                        ; Demo program for 'Dodo's' guide part 6
                        ; CP/M - RP/M version

                        ; D. R. Hunt        29/05/82
                        ; Revision   0.1    02/06/82

                        .Z80
0000'                   ASEG

                        ; System equates for CP/M - RP/M
0001                    CONIN   EQU 1           ; Get a character from keyboard
0002                    CONOUT  EQU 2           ; Send a character to display
000A                    LF      EQU 0AH         ; Send a line feed
001A                    CS      EQU 1AH         ; Clear screen character
0000                    WBOOT   EQU 0000H       ; Warm boot jump address
0005                    BDOS    EQU 0005H       ; BDOS jump address

                        ; Equates for the program
0003                    CTRLC   EQU 3           ; The exit character
000D                    CR      EQU 0DH         ; New line character
0012                    NTRYS   EQU 12H         ; The max. number of trys (must be BCD)

                                ORG 0100H       ; Start of code for CP/M - RP/M
0100    C3 0DD3                 JP START        ; Skip round patch area and text

                                ORG 0C83H       ; Start of program, to be compatible
                                                ;   with Nascom min. config. memory.

                        ; ***********************************************
                        ; ************* USER PATCHES ****************
                        ; ***********************************************

                        ;  The first patch goes to the keyboard and waits for a key
                        ;   press. The character is returned in A. The patch must
                        ;   preserve all registers and end with a RET. The patch may
                        ;   be up to 16 bytes long.
0C83    E5              GETCHR: PUSH HL         ; Save all the registers
0C84    D5                      PUSH DE
0C85    C5                      PUSH BC
0C86    0E 01                   LD C,CONIN
0C88    CD 0005                 CALL BDOS       ; Wait at keyboard for an input
0C8B    C1                      POP BC          ; Get the registers back again
0C8C    D1                      POP DE
0C8D    E1                      POP HL
0C8E    C9                      RET
0C8F    00 00 00 00             DEFB 0,0,0,0    ; Pad the space to 16 bytes

                        ; The second patch is to send a character to the display.
                        ;  The patch must preserve all registers and end with a RET.
                        ;  The patch may be up to 16 bytes long.
```



```
0C93    F5              OUTCHR: PUSH AF         ; Save all the registers
0C94    E5                      PUSH HL
0C95    D5                      PUSH DE
0C96    C5                      PUSH BC
0C97    5F                      LD E,A          ; Put the char in E (for CP/M or RP/M)
0C98    0E 02                   LD C,CONOUT
0C9A    CD 0005                 CALL BDOS       ; Send the character to the display
0C9D    C1                      POP BC          ; Get the registers back again
0C9E    D1                      POP DE
0C9F    E1                      POP HL
0CA0    F1                      POP AF
0CA1    C9                      RET
0CA2    00                      DEFB 0          ; Pad the space to 16 bytes

                        ; The third patch area is to exit from the program.
                        ;  No special precautions are required. The patch may be
                        ;  up to 3 bytes long.
0CA3    C3 0000         EXIT:   JP WBOOT        ; Jump to CP/M exit

                        ; ***********************************************
                        ; ************* WORKSPACES ****************
                        ; ***********************************************

0CA6    01 02 03        RING:   DEFB 1,2,3      ; The ring counter used by RANDOM
0CA9                    WBUF:   DEFS 10         ; The temporary word buffer
0CB3                    TRYS:   DEFS 1          ; Trys taken
0CB4                    CHRTRD: DEFS NTRYS      ; Table of letters tried
0CC6                            DEFS 12H        ; The stack space
0CD8                    STACK:

                        ; ***********************************************
                        ; ************* MESSAGES ****************
                        ; ***********************************************

0CD8    1A 20 20 20     TTLMSG: DEFB CS,"             SIMPLE HANGMAN",0
0CDC    20 20 20 20
0CE0    20 20 20 20
0CE4    20 20 53 49
0CE8    4D 50 4C 45
0CEC    20 48 41 4E
0CF0    47 4D 41 4E
0CF4    00
0CF5    49 20 68 61     MSG1:   DEFB "I have just thought of a "
0CF9    76 65 20 6A
0CFD    75 73 74 20
0D01    74 68 6F 75
0D05    67 68 74 20
0D09    6F 66 20 61
0D0D    20
0D0E                    CHRCNT: DEFS 1
0D0F    20 6C 65 74             DEFB " letter word",0
0D13    74 65 72 20
0D17    77 6F 72 64
0D1B    00
```

```
OD1C  50 6C 65 61  MSG2:   DEFB "Please guess a letter ",0
OD20  73 65 20 67
OD24  75 65 73 73
OD28  20 61 20 6C
OD2C  65 74 74 65
OD30  72 20 00
OD33  59 6F 75 27  MSG3:   DEFB "You've used this letter already",0
OD37  76 65 20 75
OD3B  73 65 64 20
OD3F  74 68 69 73
OD43  20 6C 65 74
OD47  74 65 72 20
OD4B  61 6C 72 65
OD4F  61 64 79 00
OD53  54 72 79 20  MSG4:   DEFB "Try number ",0
OD57  6E 75 6D 62
OD5B  65 72 20 00
OD5F  20 20 20 20  MSG5:   DEFB "      ",0
OD63  20 00
OD65  53 6F 72 72  MSG6:   DEFB "Sorry, you've run out of trys",0
OD69  79 2C 20 79
OD6D  6F 75 27 76
OD71  65 20 72 75
OD75  6E 20 6F 75
OD79  74 20 6F 66
OD7D  20 74 72 79
OD81  73 00
OD83  54 68 65 20  MSG7:   DEFB "The word was ",0
OD87  77 6F 72 64
OD8B  20 77 61 73
OD8F  20 00
OD91  44 6F 20 79  MSG8:   DEFB "Do you want another go (Y/N) ? ",0
OD95  6F 75 20 77
OD99  61 6E 74 20
OD9D  61 6E 6F 74
ODA1  68 65 72 20
ODA5  67 6F 20 28
ODA9  59 2F 4E 29
ODAD  20 3F 20 00
ODB1  43 6F 6E 67  MSG9:   DEFB "Congratulations you've won !!!",0
ODB5  72 61 74 75
ODB9  6C 61 74 69
ODBD  6F 6E 73 20
ODC1  79 6F 75 27
ODC5  76 65 20 77
ODC9  6F 6E 20 21
ODCD  21 21 00
ODD0  0D 0A 00     CRMSG:  DEFB CR,LF,0
```



```
                ; ********************************************
                ; **********  START OF THE PROGRAM  ************
                ; ********************************************

ODD3  31 0CD8   START:  LD SP,STACK
                ; Initialize the TRYS counter and the letters tried buffer
ODD6  AF                XOR A           ; Clear A
ODD7  32 0CB3           LD (TRYS),A     ; Save it in TRYS
ODDA  32 0CB4           LD (CHRTRD),A   ; Save it in CHRTRD as well
                ; Put up the title message
ODDD  21 0CD8           LD HL,TTLMSG    ; Point to message
ODE0  CD 0EE9           CALL SNDTXT     ; Send it
ODE3  CD 0EDF           CALL CRLF2      ; Send two newlines
                ; Get a random number
ODE6  3A 0F31           LD A,(NOWRDS)   ; Get the number of words
ODE9  CD 0F07           CALL RANDOM     ; Think of a random number
ODEC  3C                INC A           ; INC by 1 as it can never be 0
ODED  47                LD B,A          ; Put the number in B
                ; Select the word from the table
ODEE  21 0F32           LD HL,WORDS     ; Point to start of words
ODF1  7E        LOOP1:  LD A,(HL)       ; Get a character
ODF2  23                INC HL          ; Point to next character
ODF3  B7                OR A            ; Test it for 0
ODF4  20 FB             JR NZ,LOOP1     ; It isn't 0 so get next
ODF6  10 F9             DJNZ LOOP1      ; It is 0 so DEC B and get next ..
                                        ;  ... until B is 0
                ; Copy the word to the buffer
                ;  HL already points to it, DE points to buffer,
                ;  count the letters in B. (B is now 0)
ODF8  11 0CA9           LD DE,WBUF      ; Point DE to WBUF
ODFB  7E        LOOP2:  LD A,(HL)       ; Get the character
ODFC  12                LD (DE),A       ; Put it in WBUF
ODFD  23                INC HL          ; Point to next character
ODFE  13                INC DE          ; Point to next in WBUF
ODFF  04                INC B           ; Count it
0E00  B7                OR A            ; Test it for 0
0E01  20 F8             JR NZ,LOOP2     ; It isn't 0 so get next
                ; Convert the count in B to an ASCII character
                ;   and stuff it in the message. (Self modifying code ...
                ;   ... naughty naughty.)
0E03  3E 2F             LD A,30H-1      ; Put 30H - 1 in A ...
                                        ;  ... because B's one too big
0E05  80                ADD A,B         ; Adding B to A gives the number
0E06  32 0D0E           LD (CHRCNT),A   ; Put it in the message
                ; Send the message
0E09  21 0CF5           LD HL,MSG1      ; Point to the message
0E0C  CD 0EE9           CALL SNDTXT
0E0F  CD 0EDF           CALL CRLF2      ; Send two newlines
                ; This is the start of the main loop
                ;  Test to see if maximum number of trys have been made
0E12  3A 0CB3   LOOP3:  LD A,(TRYS)     ; Get the number of trys so far
0E15  FE 12             CP NTRYS        ; Compare with the maximum
0E17  20 32             JR NZ,CONT2     ; Not there yet, so skip the lose
```

```
0E19  21 0D65          LD HL,MSG6      ; Point to 'Lose' message
0E1C  CD 0EE9          CALL SNDTXT
0E1F  CD 0EE2          CALL CRLF1      ; Send a newline
0E22  21 0D83  RSTART: LD HL,MSG7      ; Point to 'The word was' message
0E25  CD 0EE9          CALL SNDTXT
                ; Display the word
0E28  21 0CA9          LD HL,WBUF      ; Point to the word
0E2B  7E       LOOP4:  LD A,(HL)       ; Get a character
0E2C  B7               OR A            ; Test for end of word
0E2D  28 08            JR Z,CONT1      ; If it is, skip the rest
0E2F  23               INC HL          ; Point to next character
0E30  E6 7F            AND 7FH         ; Strip off any flags
0E32  CD 0C93          CALL OUTCHR     ; Send the character
0E35  18 F4            JR LOOP4        ; Go and get the next
0E37  CD 0EDF  CONT1:  CALL CRLF2      ; Send two newlines
                ; Ask if the game is to continue
0E3A  21 0D91          LD HL,MSG8      ; Send the 'More?' message
0E3D  CD 0EE9          CALL SNDTXT
0E40  CD 0C83          CALL GETCHR     ; Get an input
0E43  FE 59            CP "Y"          ; Compare against 'Y'
0E45  CA 0DD3          JP Z,START      ; If 'Y' start again ...
0E48  C3 0CA3          JP EXIT         ;  ... else exit.
                ; Ask for a letter
0E4B  21 0D1C          CONT2:  LD HL,MSG2      ; Point to the message
0E4E  CD 0EE9          CALL SNDTXT
                ; Get a letter
0E51  CD 0C83          CALL GETCHR
                ; Test the character for an exit
0E54  FE 03            CP CTRLC        ; Test against 'exit character'
0E56  CA 0CA3          JP Z,EXIT
                ; Save the letter and send a newline
0E59  47               LD B,A
0E5A  CD 0EE2          CALL CRLF1      ; Send a new line
                ; Now check to see if this letter has been used before
0E5D  21 0CB4          LD HL,CHRTRD    ; Point to the store of tried letters
0E60  7E       LOOP5:  LD A,(HL)       ; Get a letter from CHRTRD
0E61  B7               OR A            ; Test it for end
0E62  28 0F            JR Z,CONT3      ; It is end so skip the rest
0E64  23               INC HL          ; Point to next if CHRTRD
0E65  B8               CP B            ; Test it against B
0E66  20 F8            JR NZ,LOOP5     ; Its not the same so get the next
0E68  21 0D33          LD HL,MSG3      ; It is the same so tell 'em ...
0E6B  CD 0EE9          CALL SNDTXT
0E6E  CD 0EE2          CALL CRLF1      ; ... send a newline ...
0E71  18 9F            JR LOOP3        ; ... and go round again
                ; Save the letter in CHRTRD and stick a new 0 on the end
0E73  78       CONT3:  LD A,B          ; Put the letter in A
0E74  77               LD (HL),A       ; Save it in CHRTRD
0E75  23               INC HL          ; Point to next in CHRTRD
0E76  36 00            LD (HL),0       ; Stick a 0 there
                ; Count the try in the TRYS counter
0E78  3A 0CB3          LD A,(TRYS)     ; Get the trys in A
0E7B  3C               INC A           ; Count up one
0E7C  27               DAA             ; Convert to packed BCD
0E7D  32 0CB3          LD (TRYS),A     ; Save it again
```



```
                ; Scan through the word to see if the character is there
0E80  21 0CA9          LD HL,WBUF      ; Point to the word
0E83  7E       LOOP6:  LD A,(HL)       ; Get a character
0E84  B7               OR A            ; Test it for end
0E85  28 09            JR Z,CONT5      ; Is end, so skip the rest
0E87  B8               CP B            ; Compare with the letter
0E88  20 03            JR NZ,CONT4     ; No compare, so skip flagging it
0E8A  C6 80            ADD A,80H       ; Set bit 7 to flag it ...
0E8C  77               LD (HL),A       ; ... and put it back
0E8D  23       CONT4:  INC HL          ; Point to next character
0E8E  18 F3            JR LOOP6        ; Go round again
                ; Test the word to see if all characters flagged (win)
0E90  21 0CA9  CONT5:  LD HL,WBUF      ; Point to the word
0E93  7E       LOOP7:  LD A,(HL)       ; Get a character
0E94  23               INC HL          ; Point to next
0E95  FE 80            CP 80H          ; Test if greater than 7FH
0E97  30 FA            JR NC,LOOP7     ; It is, so get next
0E99  B7               OR A            ; It's not so test for end
0E9A  20 0C            JR NZ,CONT6     ; It's not end, so skip the rest
                ; All flagged, so tell 'em you've won
0E9C  21 0DB1          LD HL,MSG9      ; Send 'Win' message
0E9F  CD 0EE9          CALL SNDTXT
0EA2  CD 0EE2          CALL CRLF1      ; Send a newline
0EA5  C3 0E22          JP RSTART       ; Go back to 'The word was' message
                ; Not all flagged yet, so report results
0EA8  21 0D53  CONT6:  LD HL,MSG4      ; Send 'Trys' message
0EAB  CD 0EE9          CALL SNDTXT
0EAE  3A 0CB3          LD A,(TRYS)     ; Send the number of trys
0EB1  CD 0EF2          CALL B2HEX
0EB4  21 0D5F          LD HL,MSG5      ; Send 'Spaces' message
0EB7  CD 0EE9          CALL SNDTXT
                ; Print the word double spaced with dashes where not flagged
0EBA  21 0CA9          LD HL,WBUF      ; Point to the word
0EBD  7E       LOOP8:  LD A,(HL)       ; Get a character
0EBE  23               INC HL          ; Point to next character
0EBF  B7               OR A            ; Test for end
0EC0  28 17            JR Z,CONT9      ; Is end, so skip rest
0EC2  FE 80            CP 80H          ; Isn't end, so test if flagged
0EC4  38 07            JR C,CONT7      ; Isn't flagged, so skip display
0EC6  E6 7F            AND 7FH         ; Strip the flag ...
0EC8  CD 0C93          CALL OUTCHR     ; ... and display it
0ECB  18 05            JR CONT8        ; Skip round 'unflagged' case
0ECD  3E 2D    CONT7:  LD A,"-"        ; Not flagged, so send a '-' instead
0ECF  CD 0C93          CALL OUTCHR
0ED2  3E 20    CONT8:  LD A," "        ; Send a space to double space it
0ED4  CD 0C93          CALL OUTCHR
0ED7  18 E4            JR LOOP8        ; Go and get the next
0ED9  CD 0EDF  CONT9:  CALL CRLF2      ; Send two newlines
0EDC  C3 0E12          JP LOOP3        ; Go back to start of main loop
```

```
                        ; ***********************************************
                        ; ************* SUBROUTINES *********************
                        ; ***********************************************

                        ; This subroutine sends one or two newlines
0EDF    CD 0EE2         CRLF2:  CALL CRLF1
0EE2    21 0DD0         CRLF1:  LD HL,CRMSG
0EE5    CD 0EE9                 CALL SNDTXT
0EE8    C9                      RET

                        ; This subroutine sends a line of text up to a 0
                        ;  Enter with HL pointing to start
0EE9    7E              SNDTXT: LD A,(HL)       ; Get the character
0EEA    23                      INC HL          ; Point to the next
0EEB    B7                      OR A            ; Test it for 0
0EEC    C8                      RET Z           ; Return if so ...
0EED    CD 0C93                 CALL OUTCHR     ; ... otherwise send it
0EF0    18 F7                   JR SNDTXT       ; Get the next

                        ; This subroutine outputs A in HEX
                        ;  Originally from NAS-BUG, rewritten by
                        ;  Richard Beal and dealt with at length
                        ;  in episode 5 of Dodo's guide.
0EF2    F5              B2HEX:  PUSH AF         ; Save the byte
0EF3    1F                      RRA             ; Move high nibble ...
0EF4    1F                      RRA             ; ... into low nibble
0EF5    1F                      RRA
0EF6    1F                      RRA
0EF7    CD 0EFB                 CALL B1HEX      ; Call output routine
0EFA    F1                      POP AF          ; Get the byte back
                        ; Output low half of A in HEX
0EFB    E6 0F           B1HEX:  AND 0FH         ; Strip the high nibble
0EFD    C6 90                   ADD A,90H       ; Do the 'fiddles' as ..
0EFF    27                      DAA             ; .. described in episode 5
0F00    CE 40                   ADC A,40H
0F02    27                      DAA
0F03    CD 0C93                 CALL OUTCHR     ; Display the nibble
0F06    C9                      RET

                        ; This is the random subroutine.
                        ;  Originally written by Howard Birkett, rewritten by
                        ;  Richard Beal and ultimately pinched by me!
                        ;  Detail explanation is complicated, we'll leave
                        ;  it until some other time

                        ; Call with A = n
                        ; Return    A = 0 to n-1
0F07    B7              RANDOM: OR A            ; If the number is 0 reject it
0F08    C8                      RET Z
0F09    C5                      PUSH BC         ; Save the registers
0F0A    D5                      PUSH DE
0F0B    E5                      PUSH HL
0F0C    57                      LD D,A
0F0D    ED 5F                   LD A,R
```



```
0F0F    47                      LD B,A
0F10    21 0CA6         RND1:   LD HL,RING
0F13    0E 03                   LD C,3
0F15    3A 0CA8                 LD A,(RING+2)
0F18    E6 42                   AND 42H
0F1A    C6 3E                   ADD A,3EH
0F1C    17                      RLA
0F1D    17                      RLA
0F1E    CB 16           SHIFT:  RL (HL)
0F20    23                      INC HL
0F21    0D                      DEC C
0F22    20 FA                   JR NZ,SHIFT
0F24    10 EA                   DJNZ RND1
0F26    3A 0CA6                 LD A,(RING)
0F29    92              RND2:   SUB D
0F2A    30 FD                   JR NC,RND2
0F2C    82                      ADD A,D
0F2D    E1                      POP HL          ; Restore the registers
0F2E    D1                      POP DE
0F2F    C1                      POP BC
0F30    C9                      RET

                        ; ***********************************************
                        ; ************* TABLE OF WORDS ******************
                        ; ***********************************************

                        ; NOWRDS contains the number of words in the table.
                        ;  The first byte must be 0, and each word must end in 0.
                        ;  No word should be more than 9 characters long.
0F31    23              NOWRDS: DEFB 35
0F32    00              WORDS:  DEFB 0
0F33    4D 4F 55 53             DEFB "MOUSE",0
0F37    45 00
0F39    48 41 54 00             DEFB "HAT",0
0F3D    52 41 42 42             DEFB "RABBIT",0
0F41    49 54 00
0F44    44 4F 4C 4C             DEFB "DOLL",0
0F48    00
0F49    43 41 54 00             DEFB "CAT",0
0F4D    48 4F 55 53             DEFB "HOUSE",0
0F51    45 00
0F53    54 4F 59 53             DEFB "TOYS",0
0F57    00
0F58    50 45 47 00             DEFB "PEG",0
0F5C    44 4F 47 00             DEFB "DOG",0
0F60    50 52 41 4D             DEFB "PRAM",0
0F64    00
0F65    50 49 43 54             DEFB "PICTURE",0
0F69    55 52 45 00
0F6D    44 49 4E 4E             DEFB "DINNER",0
0F71    45 52 00
0F74    43 4C 4F 43             DEFB "CLOCK",0
0F78    4B 00
0F7A    52 41 44 49             DEFB "RADIO",0
0F7E    4F 00
```

```
OF80    42 4F 58 00          DEFB "BOX",0
OF84    43 55 50 00          DEFB "CUP",0
OF88    47 4C 41 53          DEFB "GLASSES",0
OF8C    53 45 53 00
OF90    57 41 54 43          DEFB "WATCH",0
OF94    48 00
OF96    48 4F 52 53          DEFB "HORSE",0
OF9A    45 00
OF9C    50 4F 53 54          DEFB "POSTER",0
OFA0    45 52 00
OFA3    56 41 53 45          DEFB "VASE",0
OFA7    00
OFA8    42 4F 4F 4B          DEFB "BOOKS",0
OFAC    53 00
OFAE    50 45 4E 00          DEFB "PEN",0
OFB2    50 45 4E 43          DEFB "PENCIL",0
OFB6    49 4C 00
OFB9    52 55 42 42          DEFB "RUBBER",0
OFBD    45 52 00
OFC0    4B 4E 4F 42          DEFB "KNOB",0
OFC4    00
OFC5    42 41 53 4B          DEFB "BASKET",0
OFC9    45 54 00
OFCC    42 55 47 47          DEFB "BUGGY",0
OFD0    59 00
OFD2    43 55 50 00          DEFB "CUP",0
OFD6    4D 4F 4E 45          DEFB "MONEY",0
OFDA    59 00
OFDC    50 45 4E 00          DEFB "PEN",0
OFE0    42 41 47 00          DEFB "BAG",0
OFE4    52 45 43 4F          DEFB "RECORDER",0
OFE8    52 44 45 52
OFEC    00
OFED    42 49 4C 4C          DEFB "BILLS",0
OFF1    53 00
OFF3    4F 50 45 52          DEFB "OPERATION",0
OFF7    41 54 49 4F
OFFB    4E 00

                             END
```


Macros:

Symbols:

0EFB	B1HEX	0EF2	B2HEX	0005	BDOS
0D0E	CHRCNT	0CB4	CHRTRD	0001	CONIN
0002	CONOUT	0E37	CONT1	0E4B	CONT2
0E73	CONT3	0E8D	CONT4	0E90	CONT5
0EA8	CONT6	0ECD	CONT7	0ED2	CONT8
0ED9	CONT9	000D	CR	0EE2	CRLF1
0EDF	CRLF2	0DD0	CRMSG	001A	CS
0003	CTRLC	0CA3	EXIT	0C83	GETCHR
000A	LF	0DF1	LOOP1	0DFB	LOOP2
0E12	LOOP3	0E2B	LOOP4	0E60	LOOP5
0E83	LOOP6	0E93	LOOP7	0EBD	LOOP8
0CF5	MSG1	0D1C	MSG2	0D33	MSG3
0D53	MSG4	0D5F	MSG5	0D65	MSG6
0D83	MSG7	0D91	MSG8	0DB1	MSG9
0F31	NOWRDS	0012	NTRYS	0C93	OUTCHR
0F07	RANDOM	0CA6	RING	0F10	RND1
0F29	RND2	0E22	RSTART	0F1E	SHIFT
0EE9	SNDTXT	0CD8	STACK	0DD3	START
0CB3	TRYS	0CD8	TTLMSG	0000	WBOOT
0CA9	WBUF	0F32	WORDS		

No Fatal error(s)

Colour your Computer Green (for envy?)

by D. R. Hunt

What to date has been much speculated upon, is highly colourful, and so far has remained completely invisible? What is it, that by the time you get round to reading this, you should just about be able to rush out and buy? (At least don't blame me if you can't.) What is he talking about? (Does he ever know what he's talking about?) Of course, it's the Nascom Advanced Video Card!!

At a recent Nascom dealer meeting it was there in all it's glory, and for the first time we were allowed to poke it a little bit. No, they weren't generous enough to donate one to a worthy cause (me) so what follows is a description of what it is and what it does rather than a review of how good (or bad) it is.

First impressions are certainly good, and it could prove itself extremely useful in the educational field and, to a lesser extent (as far as the colour graphics is concerned), in the business field. The business area will be much more interested in it's 80 x 25 screen format. For the home user, well I don't know. I still maintain that any colour graphics is a facility that most could do without and on balance, will remain without. Although the initial cost is modest, there are hidden overheads. For instance, in the average home there is only one colour display device, the TV, and the only reason that your home computer is allowed in the house at all is because the 'Mrs' is able to sit in front of the box watching Coronation Street, whilst you play with your toys. Suggest that the colour TV should be connected to the computer instead of the aerial, and you, the computer, the newly acquired colour board and any other bits of your assorted iron-mongery, are liable to find themselves out in the street. The alternative, buy another colour TV. But if you are going to do that, then, realising that the rotten performance of a colour TV connected to a colour computer is almost all down to the poor bandwidth of the TV colour demodulating circuits (for a TV picture it doesn't need to be better than about 1.5MHz), then, why not a colour monitor. You then find that what was once a relatively cheap colour facility has turned into a wallet depleting demon, at a cost of several times the original cost of the computer. Some home users will do this. But I think the majority will stay with playing Space Invaders and Galaxians in black and white. Perhaps I'm wrong, it remains to be seen.

What does it consist of and how does it work. Firstly it should be made clear that except when used with CP/M business type systems, Nascom consider the AVC as a peripheral rather than the main display device. This means that two monitors are really required, although the Nascom video output can be directed through the AVC if required. If two monitors are used, a B & W one would be used for displaying the programming details from the standard Nascom video, and a colour one to display the colour results. This is a useful scheme and one commonly adopted in colour software development. Anyway, the AVC itself is best considered as three planes of dynamic RAM arranged one above the other. The video controlling being achieved by Motorola MC6845 CRTC processor. Each plane is 16K and each plane deals with one of the three primary colours. Each plane is identically memory mapped to the screen in a similar fashion to the existing video RAM is mapped to the screen on the Nascom. The only difference that need concern you is that there is 16K of video RAM in each plane as opposed to the 1K of RAM in the Nascom. Using 16K of RAM allows a resolution of some 390 dots horizontally and 256 dots vertically. For full colour use, the three planes are effectively placed in parallel, providing three outputs one representing RED, one representing GREEN and one representing BLUE. If these output were fed directly to what is known as an RGB monitor (that's one with three inputs, one red, one green and one blue), then that's all there is to it. On the Nascom card further options are allowed, the RGB signals may be fed to an optional PAL encoder and then to an optional high bandwidth UHF modulator to provide a composite UHF TV signal. There is also a monochrome (B & W) monitor output.

On the monitor screen, the RED output lights up the red dots, the GREEN output lights the green dots, etc. Additive colour mixing takes place on the screen, so that if, for instance, the RED and BLUE signals were on together then a sort of

mauve colour, magenta, would be produced. By combinations of the three primary colours eight colours can be produced, ranging from white, where all three signals are present at once, through magenta, cyan, yellow, red, green and blue, to black, where all three signals are off. Further colours are available by mixing the proportions of the above eight colours within a certain area. For instance a green dot surrounded by a number of red dots would result in a brownish red colour, the 'brownness' being proportional to the number of green dots within a given area. This of course implies a coarser resolution than is obtainable with pure colours, but is ideal for backgrounds etc. Some 4000 shades are obtainable in this way.

The three 16K RAMs are memory mapped into the computer in much the same way as the already existing video RAM. However, this does not gobble up vast acres of user RAM, as the colour RAM is on different 'pages'. In other words, they are 'paged' in place of the user RAM, which is simultaneously 'paged' out, thus overlaying user RAM whilst being addressed, when video update is complete they are 'paged' out again and the user RAM is 'paged' back in untouched. For those who are worried about this paging scheme conflicting with the existing user RAM paging scheme, don't. The AVC uses different ports to address the pages and no conflict arises. Unfortunately the ports that Nascom have adopted clash with those used by Gemini's IVC and, although both the IVC and the AVC can be set to use alternative ports, this means some aggravation for those who already have an IVC, want to add an AVC for colour, and want to use the standard software drivers available for each card. Shame.

Nascom have indulged in bit of cleverness in the flexibility in which the three colour RAMs can be arranged. As has already been mentioned, the three RAMs are effectively laid one on top of the next. Now imagine moving the top RAM sideways (to the right or left, it doesn't matter), and dropping it down one layer so that it butts against the one on the middle layer. Now double the addressing speed to this 32K video RAM and a resolution of 780 x 256 results. Now two colour layers are available, the bottom one of highish resolution (390 x 256) and the top of very high resolution. The backgrounds could be produced by the lower layer to quite acceptable resolution whilst very fine detail would be comfortably resolved by the upper 32K layer. The outputs of the two RAM planes may be directed to any two of the three colour outputs producing an effective result equivalent to the highest resolution yet seen on a colour card at this sort of price in any four colours of the eight colours previously available. It was interesting to note that it was mentioned that by increasing the onboard 16MHz crystal to 20MHz, a screen format of 100 x 25 could be achieved. It was not stated whether an equivalent increase in graphics resolution to 926 x 256 dots could be achieved at the same time (should be possible).

So far we have dealt with the graphics capability. The Nascom AVC is not fitted with a character generator. Instead a clever piece of software looks up the bit patterns of alpha-numerics from a table and transfers them to the appropriate places in the RAM planes. The 360 x 256 mode produces a 40 x 25 screen format in eight colours whilst the 80 x 25 screen format is catered for by the 720 x 256 mode in four colours. This method of character generation has both advantages and disadvantages. One of the 'prettiest' advantages demonstrated was the ability to select character sizes and aspect ratios at will. So for instance italics could be mixed into ordinary text (in contrast colours if desired) simply by stating the 'slope' angle of the characters to be displayed. Alpha-numeric characters could also be placed at odd angles on the screen, and of course things like sub-scripts and super-scripts are no problem at all. The potential for this sort of character generation is quite considerable. However, there are two penalties. Speed and system RAM overhead.

The speed of screen scrolling suffers quite a bit because instead of having to only copy the character bytes from one line to the next, whole chunks of bit patterns have to be copied. Nothing too upsetting though. At first sight, 'soft-scrolling' would appear to be easy, and it is. Unfortunately, again, because of the enormous amount of 'bit shunting' required to achieve this, it is also painfully slow, too slow to be useful in fact. Another problem arises when high

speed character update to the screen display is required. Something like Naspen (or Diskpen) could not be made to work as it updates the whole screen content every time a key press is made (in the Insert mode). Without a radical redesign, 'Pen' would refresh the whole screen far too slowly to be practical. Maybe this is an argument for redesigning 'PEN', but that's a different story.

The other penalty is the RAM overhead used by the character generation software. It uses 1K of bit patterns for the characters and another 1K of user definable bit patterns for the user defined character set. That's 2K for starters. Another 2.5K is used for the 'getting and putting' of the character bit patterns and to make the AVC behave sensibly as a display device. In fact it's configured to look a bit like a Lear-Siegler ADM-3A terminal, a definite plus point. It's all quite clever, and they've crammed quite a lot into the 2.5K of control software. However, it's 4.5K of RAM space and the Nascom with its 48K RAM card (unless you have two RAM boards, or Gemini's 64K job) is not over generous when it comes to running CP/M type disk systems. Perhaps Nascom could supply the software package in EPROM to reside above the top of user RAM, the CP/M BIOS 'hooking' into the EPROM package. This would not increase the BIOS size (in fact it could make the BIOS smaller), but then if it's in EPROM, where's the RAM for the user defined character set. No doubt Mike Hessey and his lads have worked their way round that one.

To prove the point that the card could be made to perform adequately in a business type environment that old faithful Wordstar was demonstrated, and it worked well. Noticeably slower on scrolling and repositioning text than the Gemini IVC (but then the Gemini IVC works differently and doesn't have colour), but anyone used to seeing Wordstar on a terminal being driven by something like a DEC-10 would be immediately convinced of the speed at which the AVC could be made to perform. Sadly it left me cold. Nothing to do with the AVC, it's simply that I consider Wordstar as being one of the most unnecessarily complex and frustrating lumps of software around (wait for the defensive letters to come pouring in after that comment).

As mentioned earlier, there are options that can be supplied for the AVC. As standard the PAL encoder is not fitted, although that should not be expensive, a tenner or so I would think. Enough software is supplied to make it work in either colour graphic or alphanumeric mode, but an enhanced software package will be available to enable things like the rotation of solid objects and some sort of picture 'zooming'. The standard software seems to link into Nas-sys with minimal difficulty, probably by using the 'U' command functions.

So to sum up on first impressions. It's good, and at about £155.00, not too expensive for the sort of market where most Nascom's are used. It's speed is not overly impressive so don't think you can do high speed animation with it. (For those who saw the Horizon programme on computer graphics, don't forget that "Carla's Island" used the whole resources of the Cray One computer, and could still only run at one frame in eight seconds; and that "Teapot" required several million pounds worth of DECs working in parallel and many tens of man-years of software development.) Within it's limitations (which aren't many from the point of view of the potential users) it performs well and achieves its original aims entirely. (By the way, the rumours were true ... the AVC is 10" x 8".)

To change the subject, we owe our apologies to Nascom. In the last two issues we could have created the impression that the Nascom FDC card was simply the pre-receivership Nascom FDC card put into production. Nascom tell us that this is not the case, and that the card is a redesign and several significant design changes have been made. Also, arising from the last issue, Nascom would like to point out that the Nascom disk system is now available with the TEAC FD-50F drives giving some 700K of formatted space per drive, and that judging from advertised prices, the price advantage still lies with Nascom rather than Cumana as was implied. Lastly, our overview of disks systems should have drawn a distinction between DCS-DOS, DCS-DOS2 and NAS-DOS. NAS-DOS and DCS-DOS2 are related and both contain enhancements over the original DCS-DOS. Our apologies to Nascom and Dove Computing Services for these inaccuracies.

```
;-----------------------------------------------
;
;       PolyDos 2.0 DUMP utility
;       by Anders Hejlsberg, June 1982
;
;-----------------------------------------------

        REFS    SYSEQU          ;Get symbols from SYSEQU
        REF                     ;Get all of them

        ORG     1000H           ;Define program origin and
        IDNT    $,$             ;load/execute addresses

        LD      B,110B          ;Type and drive optional
        LD      DE,(CLINP)      ;Pick up command line ptr
        LD      HL,FCB          ;Point to FCB
        SCAL    ZCFS            ;Convert file name
        SCAL    ZCKER           ;Check for error
        LD      (CLINP),DE      ;Save new command line ptr
        SCAL    ZRDIR           ;Read directory
        SCAL    ZCKER           ;Check for error
        SET     4,B             ;Copy dir info to FCB
        SET     5,B             ;Include locked files
        SCAL    ZLOOK           ;Look up file
        SCAL    ZCKER           ;Check for error
        LD      HL,POUTT        ;Point to output table
        SCAL    ZNOM            ;Activate printer
        CALL    TOPPG           ;Move to top of form
        LD      DE,0            ;Init dump address
        LD      C,E             ;Init page number
D1:     LD      HL,(FCB+FNSC)   ;Get sector counter
        LD      A,H             ;Zero?
        OR      L
        JP      Z,D11           ;Yes => done
        DEC     HL              ;Decrement
        LD      (FCB+FNSC),HL   ;Save it
        PUSH    DE              ;Save dump address
        PUSH    BC              ;Save page number
        LD      HL,BUFFER       ;Point to RAM buffer
        LD      DE,(FCB+FSEC)   ;Get sector address
        LD      A,(DDRV)        ;Read from dir drive
        LD      C,A
        LD      B,1             ;One sector
        SCAL    ZDRD            ;Go read
        SCAL    ZCKER           ;Check for error
        INC     DE              ;Point to next sector
        LD      (FCB+FSEC),DE   ;Save address
        POP     BC              ;Restore page number
        POP     DE              ;Restore dump address
D2:     PUSH    HL              ;Save buffer pointer
        PUSH    DE              ;Save dump address
        LD      HL,(CLINP)      ;Point to heading
        LD      A,(HL)          ;Load first character
        OR      A               ;Empty?
        JR      Z,D5            ;Yes => skip
        LD      HL,PLCT         ;Point to PLCT
        LD      A,(HL)          ;Load it
        INC     HL              ;Point to PPOS
        OR      (HL)            ;Zero if PLCT=PPOS=0
        JR      NZ,D5           ;Not at top of form => skip
        RST     PRS             ;Print DUMP message
        DB      'DUMP V1.0'
        DB      TAB,0
```

```
        LD      HL,PCPL         ;Length of user heading
        LD      A,(HL)          ;is PCPL-PBMG-24
        INC     HL
        SUB     (HL)
        SUB     24
        LD      B,A             ;Put length in B
        LD      HL,(CLINP)      ;Point to heading
D3:     LD      A,(HL)          ;Get character
        INC     HL              ;Point to next
        OR      A               ;End of string?
        JR      NZ,D4           ;No => skip
        DEC     HL              ;Back to the null
        LD      A,' '           ;Load a blank
D4:     RST     ROUT            ;Print character
        DJNZ    D3              ;Loop
        RST     PRS             ;Print PAGE message
        DB      ' PAGE ',0
        LD      A,C             ;Get page number
        ADD     A,1             ;Increment (must ADD)
        DAA                     ;Keep it in decimal
        LD      C,A
        SCAL    ZB2HEX          ;Print page number
        SCAL    ZCRLF           ;Do CR/LF
        SCAL    ZCRLF           ;Do CR/LF
D5:     POP     HL              ;Get dump address
        PUSH    BC
        SCAL    ZTBCD3          ;Print dump address
        POP     BC
        EX      (SP),HL         ;Save addr and get buffer ptr
        PUSH    HL              ;Save buffer pointer
        LD      B,16            ;Print 16 bytes
D6:     LD      A,B             ;First or ninth byte?
        AND     7
        JR      NZ,D7           ;No => skip
        SCAL    ZSPACE          ;Print a blank
D7:     LD      A,(HL)          ;Get byte
        SCAL    ZB2HEX          ;Print it
        SCAL    ZSPACE          ;Print a blank
        INC     HL              ;Point to next byte
        DJNZ    D6              ;Repeat 16 times
        SCAL    ZSPACE          ;Print a blank
        POP     HL              ;Restore buffer pointer
        POP     DE              ;Get dump address
        LD      B,16            ;Print 16 characters
D8:     LD      A,(HL)          ;Get byte
        CP      ' '             ;Control character?
        JR      C,D9            ;Yes => skip
        CP      7FH             ;Graphic or DEL?
        JR      C,D10           ;No => printable
D9:     LD      A,'.'           ;Not printable
D10:    RST     ROUT            ;Print it
        INC     HL              ;Point to next byte
        INC     DE              ;Increment dump address
        DJNZ    D8              ;Repeat 16 times
        SCAL    ZCRLF           ;Do CR/LF
        INC     E               ;Need a new sector?
        DEC     E
        JP      Z,D1            ;Yes => go get it
        JP      D2              ;Print next line
D11:    CALL    TOPPG           ;Move to top of page
        SCAL    ZMRET           ;Back to PolyDos

; If paginated output is requested, ensure
; that the printer is at the top of a form
```

```
TOPPG:  LD      HL,(CLINP)      ;Point to heading
        LD      A,(HL)          ;Load first character
        OR      A               ;Empty?
        RET     Z               ;Yes => return
        LD      HL,PLCT         ;Point to PLCT
        LD      A,(HL)          ;Load it
        INC     HL              ;Point to PPOS
        OR      (HL)            ;Zero if PLCT=PPOS=0
        RET     Z               ;At top of form => return
        LD      A,FF            ;Move to top of form
        RST     ROUT
        RET

; Printer output table

POUTT:  DB      ZPOUT,0

; Workspace

FCB:    DS      20              ;FCB buffer
BUFFER: DS      256             ;Sector buffer

        END
```

DUMP is a utility program for PolyDos 2.0. It is used to output hex listings of disk files to the printer. To run DUMP, use the following command line:

    $DUMP filename heading

where filename is a PolyDos file name (drive and extension are optional), and heading is any string of ASCII characters, seperated from the file name by at least one blank. Each line output shows the address of the first byte (starting with 0000H), thenthe values of 16 bytes in hex, and finally the same values in ASCII, if they are printable. Note that as the lines are 72 characters wide, the minimum printer line width is 72 (48H).

```
DUMP V1.0       PolyDos 2.0 DUMP utility                    PAGE 01

0000  06 06 ED 5B 19 C0 21 E6  10 DF 85 DF 8A ED 53 19  ...[..!.......S.
0010  C0 DF 83 DF 8A CB E0 CB  E8 DF 86 DF 8A 21 E4 10  .............!..
0020  DF 71 CD D3 10 11 00 00  4B 2A F4 10 7C B5 CA CE  .q......K*..|...
0030  10 2B 22 F4 10 D5 C5 21  FA 10 ED 5B F2 10 3A 01  .+"....!...[..:.
0040  C0 4F 06 01 DF 81 DF 8A  13 ED 53 F2 10 C1 D1 E5  .O........S.....
0050  D5 2A 19 C0 7E B7 28 3E  21 17 C0 7E 23 B6 20 36  .*..~.(>!..~#. 6
0060  EF 44 55 4D 50 20 56 31  2E 30 09 00 21 12 C2 7E  .DUMP V1.0..!..~
0070  23 96 D6 18 47 2A 19 C0  7E 23 B7 20 03 2B 3E 20  #...G*..~#. .+>
0080  F7 10 F5 EF 20 50 41 47  45 20 00 79 C6 01 27 4F  .... PAGE .y..'O
0090  DF 68 DF 6A DF 6A E1 C5  DF 66 C1 E3 E5 06 10 78  .h.j.j...f.....x
00A0  E6 07 20 02 DF 69 7E DF  68 DF 69 23 10 F1 DF 69  .. ..i~.h.i#...i
00B0  E1 D1 06 10 7E FE 20 38  04 FE 7F 38 02 3E 2E F7  ....~. 8...8.>..
00C0  23 13 10 F0 DF 6A 1C 1D  CA 29 10 C3 4F 10 CD D3  #....j...)..O...
00D0  10 DF 5B 2A 19 C0 7E B7  C8 21 17 C0 7E 23 B6 C8  ..[*..~..!..~#..
00E0  3E 0C F7 C9 8F 00 00 00  00 00 00 00 00 00 00 00  >...............
00F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

BOOK REVIEWS

by Rory O'Farrell

Writing Interactive Compilers and Interpreters, by P. J. Brown
publ. John Wiley

All fans of Professor P.J. Brown will be happy to know that the latest edition of his most readable guide to the complexities of compiler design and construction is now available in a paperback edition, at a most reasonable price reduction on the hardcover edition.

Pascal from BASIC, by P.J.Brown,
publ. Addison Wesley 1982 (cost approx £5 in paperback)

This book has recently appeared on the bookstands. This author's work will need no recommendation to those who have read his previous work, mentioned above. This new book is a detailed and extensive guide to the problems of converting from BASIC to Pascal - which is a problem that will confront more and more microcomputer users as time goes by. In this book, in his usual humorous way, Professor Brown discusses very fully the different approach needed to write successful Pascal programs from that used for BASIC. He is alive to the advantages of Pascal, but does not hesitate to deal with its disadvantages as well (Gasps of horror.. surely Pascal can't have any? Well, Heloise, you are a big girl now, and there are things you should know. Of course these are only talked about in the proper place (not in front of BASIC programmers)). In dealing with the language, he deals with the 'standard' Jensen and Wirth Pascal, which is substantially that proposed for the ISO standard. In restricting his discussion to this definition of the language, he is able to avoid heavily system dependant and non-standard extensions, referring you for these to the detailed manual which has been supplied with your particular implementation (we hope). The book is illustrated with comparative examples of Pascal and BASIC programs, with humorous cartoons as chapter headings. The book is well printed and typeset, with only three misprints coming to my notice. These were on page 81, line 7 of text, which should read

```
ecount['p',true] {p' omitted}
```

and line 35 of text, which should read

```
for row:='a' to 'z' do {z' omitted}
```

and page 134, last line, which should read

```
score:1..50;{a dartboard etc} {. omitted}
```

In the course of the book, he makes helpful hints to the would-be Pascal programmer, with sound advice on how to approach the problems of writing a program, and useful little points of style to help overcome some of the infuriating syntax errors of Pascal. For example, did you know that you don't preceed an ELSE with a ';' in an IF THEN ELSE construction? I know that it is in the syntax diagrams, but I hadn't realised that it could be as formally stated as that!

Without any reservation, I wholeheartedly recommend this book to the would be user of Pascal, more particularly if he is trying to convert from BASIC. Because of the language similarities, it will not be irrelevant to those of us who learned to program in FORTRAN II (They are now at FORTRAN 77 or 80, so you can figure out how long ago that was!). I think that in writing this book, which is light and easy to digest, but not trivial, Prof. Brown has done a service to the microcomputer fraternity, having written a very valuable contender for the title of 'Computer Book of the Year'. He might even end up being mentioned in our prayers, along with Niklaus Wirth!

Pascal - The Language and its Implementation, ed. D.W.Barron,
published John Wiley 1981

This is collection of papers on the subject of the programming language Pascal and the problems of its implementation. Some of the papers have been previously published, but are not elsewhere available. This book substantially arises from the proceedings of a symposium on the same subject. It carries a reprint of Niklaus Wirth's 'Pascal-S: A Subset and its implementation', which is useful as a complete example of a compiler for study.
This latter is also the subject of -

Programming Language Translation, by R.E.Berry
published Ellis Horwood (distrib. John Wiley) 1981

In this volume, Berry deals with the problems of translating the source language into the version of the program which can be understood by the target machine. The author gives the text of the Pascal S compiler in toto, and makes lavish use of this in his discussions, concluding with the most useful blow by blow account of the purpose of each proceedure, and the uses it makes of the various data structures of the compiler in the organisation of the code. I would recommend this book over the preceeding for those who are interested in getting an idea of what happens (and how) in a compiler.

Pascal Programs for Scientists and Engineers, by Alan Miller,
publ. SYBEX

This is a collection of assorted programs in Pascal which may be of use to people handling data or figures. It is a well laid out and very readable book, but quite expensive, so one ought really to see it first before buying it, to make sure that you needed it. The programs given include Mean and Standard deviation, Vector and Matrix operations, solution of simultaneous Linear equations, curve fitting, sorting, integration, Bessel functions, non linear curve-fitting. It does not (unfortunately!) include a Fast Fourier Transform, which is a pity.

Software Tools in Pascal, by Kernighan and Plauger,
publ. Addison Wesley 1981

Further to my notes in the last 80-BUS News, I have now received a copy of this. It is a revision of the earlier book, Software Tools, using Pascal as the language of implementation rather than RATFOR and PL/I. In doing this, it gains in readability and ease of application. Its philosophy on the construction of a series of tools of general application, which are well documented and modular, so that any maintainence or modifications are easy to implement, has been one of the most outstanding breakthroughs of the last decade. Nobody who has read these authors has been unaffected by their approach.

Pascal Implementation, by Daniels and Pemberton,
publ. Ellis Horwood (John Wiley),

This is a fully commented listing of the P4 Pascal assembler, complete with the necessary interpreter to run the compiler. It does not, contrary to my impressions from the advance publicity, list the Pcodes produced by the compiler compiling itself. In consequence, it would be necessary to cross compile the compiler on another machine, having a full Pascal implementation. The source for the P4 compiler takes up in excess of 160 Kbytes, so cross compiling on a micro-computer will be difficult, to put it mildly.

The book comes in two volumes, one of 160 pages comprising the notes and commentary, and one of 82 pages comprising the listings of the compiler and interpreter. Due to currency differences, I cannot give an exact price, but it is expensive - say in the £22 (sterling) region. I think that it would not prove very valuable to the average microcomputer user - I'd suggest P.J. Brown's 'Writing Interactive Compilers and Interpreters' or the same author's 'Pascal from BASIC' as being of more general use. If you are interested in the design of compilers, then a detailed study of one of the important modern compilers for a modern structured language may well prove enticing. See it first! After all £22 is a lot of money for a book, even in this inflationary age. A note in the book says that machine readable versions of the source programs are available from the publisher. These would be of interest if one had access to a larger machine to cross compile the compiler, subject to compatibility of formats and pricing. Most probably the source will be available only in professional tape and floppy disk formats.

CLASSIFIED ADS.

One Nascom RAM A board with 32K RAM and ROM (4x2708) ZEAP. Works perfectly at 2.5 or 4MHz. £75 the lot. Phone 0903-204521 evenings.

Nascom 1, 32K RAM, 8K ROM BASIC, graphics, Nas-Bug & Nas-Sys, sound, many extras including tapes (games, assembler, etc). Built in professional case. All documentation and INMC mags. £250 ONO. Tandy thermal printer (works with Nascom) £70 ONO. Both £300 ONO. Phone Southend (0702) 76205 - Evenings.

Teletype KSR33 printer. Excellent condition. RS232 i/f. Demonstration on Nascom 2. £70. Keyboard plus case. 80 keys. £10. Tel. Crowthorne 6894.

Nascom 2, 4MHz, 1200 baud, 32K RAM, keyboard case, cassette player, programs, books, graphics. £300. 56 2114's 200nS, 4 2716 5V EPROMs £60. 061-773-6487.

Teletype. Creed 444 (recent model) in excellent condition. Ideal for program listings. Includes software and serial interface for immediate connection to your Nascom 2. All this for only £75. Pick up from Chelmsford (Essex) or Crewe (Cheshire). Phone Mark Hughes on (0270) 582301.

Bits and P.C.s Toolkit in EPROM. £15. Mr Trewartha. Tel. 0482-43998.

IBM 3982 heavy duty Golfball printer with split platen, pin feed platen, 6 fabric ribbons, 6 carbon film ribbons, 2 golfballs (Courier 72, US ASCII) and box of wide pinfeed paper. £250 ONO. Phone 02407-2117. (Bucks.)

CLUBS
=====

There is now a Nascom Owners Club in Northern Ireland. The club meets every 2nd Wednesday in each month at Newburn Electronics, 58 Manse Road, Ballycarry, Co. Antrim. Tel. Whitehead 78330.

REVIEW OF EXTENSION BASIC by LEVEL 9 COMPUTING

by David J. Plews

(Ed.'s Note - As our printer currently has no 'at' sign, for '%' read 'at'.)

WHAT YOU GET :

One TDK D C46 cassette with Extended BASIC (hereinafter XBASIC) and relocator program and a basic demonstration program (side 1 has copies at 1200 baud, side 2 at 300 baud), one 28 page manual.

WHAT YOU NEED :

A NASCOM 1, 2 or 3 running under NAS SYS 1 or 3 with the ROM BASIC and a minimum of 16K of RAM.

LOADING :

The 1200 baud version loaded error free first time.

RELOCATING IT :

The loaded program was a full 8K. This consisted of 4K for the XBASIC program itself, a couple of hundred bytes for a relocator program and the rest was an advertising and copyright message!! To relocate the XBASIC into memory the program is executed at 1000 HEX. There then follows an advertisement and a prompt asking you where you would like XBASIC to sit in memory by entering the start address of a 4K block. I have 32K of RAM so it's 8000 HEX. There is a delay of less than 1 second while XBASIC is relocated and control is then returned to the monitor. The relocated version can then be stored on tape. There then follows a tedious rigmarole of starting XBASIC up to work with ROM BASIC. The ROM BASIC is cold started and the top of user RAM is set by the 'Memory size ?' prompt so that XBASIC is not overwritten. The appendicies help out here, for 32K RAM it's 32679. After entering this you then have to re-enter the monitor and initialise XBASIC by executing it at its start address, i.e. for me E8000. After another short commercial control is again returned to the monitor. BASIC is then warm started with a Z, and away you go!! If you have to go into monitor at any time subsequently, to return to the full 12K basic system warm start with a Z and then enter SET. You get another commercial and you're off. If you get mixed up and forget whether you're running under XBASIC , SET- will check for you.

COMMANDS :

So what do you actually get in the 4K program? Well XBASIC gives you 32 (yes 32 !!) extra commands running under ROM BASIC, and you can add your own !! The new commands are:

AUTO	BREAK	CALL	CHECK	COPY	DEC	DELAY	DELETE
EDIT	FIND	GET	HEX	IF..THEN..ELSE		INKEY	INLIN
LINE	LIST	PLOT	PRINT%	PUT	REPEAT..UNTIL		RENUMBER
REDUCE	SET	SPEED	TEST	TRACE	VDU	WHILE..WEND	
WRAP	XLIST	XREF					

Now to describe them all if the editor will let me!!

AUTO - Automatically generates line numbers after each ENTER. You set the starting number and subsequent increment. Needs no further comment.

CHECK - This looks through a program to see if there are any calls to unreferenced line numbers.

DELETE - This deletes lines in a program. The two arguments needed are inclusive.

EDIT - This command is a real gem!!! On entering the EDIT mode you can write a single program line which can fill the entire screen, i.e. over 700 characters!!! The demonstration program has a few examples in it. These long lines can be edited as normal while in the EDITmode. O.k. a line 700 characters long with multiple statements isn't exactly elegant but it IS exceedingly useful. (Also see WRAP.)

LIST - XBASIC now prints graphics characters and not keywords when you list a program. It is able to distinguish between keywords and graphics characters you type in.

REDUCE - This removes accessory material from the program i.e. spaces and REM statements. There are three modes of action:-
a) removes spaces except those in REM statements DATA statements and within quotes
b) removes REM statements (if the REM statement is at the beginning of a line then the text is removed but the REM remains so you still have to delete those line numbers)
c) combines a) and b).

RENUMBER - Renumbers a line from the specified start and increments each subsequent line as specified. A CHECK is performed first and other safeguards are included so that the program is not corrupted.

FIND - This finds the lines where the following string argument occurs, scrolling them up from the bottom of the screen.

TRACE - This is another of those commands you only dream about!! The command is turned on by the argument 1 and off by the argument 0. The variables you want displaying are stored in a subsequent string expression , you can have up to 20, i.e. the number you can squeeze onto a line!!! Following RUN you single step through the program using ENTER. The number of the current line being interpreted is displayed at the top of the screen followed by the current values of the variables asked for. Program debugging becomes so easy!! Alternatively, instead of having to hold down the ENTER key, argument 1 can be replaced with 'n' where 'n' is a delay of about 'n'msec.

XLIST - This lists a single specified line followed by the numbers of lines making references to it. Another very useful little thing.

XREF - As XLIST but the specified line is not listed.

DEC - This converts a hexadecimal number to a decimal number.

HEX - This converts a decimal number to a hexadecimal number.

CALL - This calls a machine code subroutine. The first argument is the subroutine's address in decimal. Using subsequent arguments it is possible to pass values to the subroutine.

GET - This returns the ASCII value of the next key pressed, or if no key is pressed it returns 0. No more mucking around with machine code subroutines using DOKEs and DEEKs.

INKEY - As for GET but the keyboard is scanned until the next key is pressed and its ASCII value is returned.

INLIN - This returns an entire screen line containing the cursor after ENTER. This can be used instead of INPUT as it allows the use of the cusor keys to edit the line while waiting for ENTER.

TEST - This tests to see whether a specifed key is pressed down, returning a value 0 if not and 1 if it is. The code specifying the key to be tested is NOT the ASCII value, but a hardware generated number related to the keyboard wiring. A full table is supplied in the manual's appendices.

LINE - Yet another superb command. This uses the Nas Graph 'pixels' to draw a line between two points X1,Y1 and X2,Y2. The argument 0 resets the line, 1 sets it and 2 inverts it. It's very fast!!

PLOT - This sets, resets or inverts the point X,Y.

PRINT% - This commences a print at the specified point and has been included, so the manual says, for easier conversion Of TRS 80 programs to NASCOM's. Without the '%' PRINT acts as normal.

PUT - Arguments which are numbers are printed as their equivalent ASCII characters and strings are printed as messages.

WRAP - This is yet another gem!! When enabled by a '+' it prevents word wrap round i.e. stops a word being printed half at the end of one line and the other half at the beginning of the next. In conjunction with the EDIT command it makes text handling a real doddle!! To disable it the argument is '-'.

VDU - This prints strings and evaluates string expressions and prints them at the coordinates specified as per ordinary SCREEN coordinates. The top line can be written to.

IF..THEN..ELSE - Speaks for itself.

REPEAT..UNTIL - Loop until condition is true.

WHILE..WEND - If the condition is true then loop. These commands give truely structured programming in BASIC, and they are VERY easy to use. You don't need to know Pascal!!

BREAK - This enables or disables the break action of the ENTER key. So what!!

COPY - This is very similar to the C command of NAS SYS except the arguments are in decimal. It is possible to over write your program!! It might be useful for fast action graphics games.

DELAY - The argument 'n' causes a delay of nmsec.

SET - Already explained.

SPEED - This allows control of the repeat keyboard speeds. The first argument delays the start of the repeat and the second argument is the interval between repeats.

You might have gathered by now that I'm rather impressed with XBASIC and blind to its faults. It does in fact have a few bad points :

1) There is no APPEND which I think is a major ommission. The manual explains how it can be done by saving a listing to cassette, which is almost an admission of the ommission!

2) The starting up of XBASIC is a real bind. The manual indicates that the ROM version of XBASIC does all the initialisation for you. It should be quite easy to modify the tape version to do this too.

3) The manual is very clear and the appendicies are very helpful, but I would have liked an assembly listing of XBASIC, the extra fee would have been well worth it.

4) Despite there being some extremely useful new commands there are in my opinion some unnecessary ones viz BREAK,XREF or XLIST (they do almost the same thing), CHECK (RENUMBER does this any way) and PRINT%, PUT and VDU are nice but a bit extravagant. I've already mentioned I would have liked an APPEND, but another command I would have really liked to see is 'FIND string and REPLACE with'. I KNOW how useful this would be but I've never seen it mentioned anywhere. Is it really that difficult to do? Well I'll soon find out when I start to attempt adding (and replacing?) commands to XBASIC.

5) The demo program is a bit simple.

CONCLUSION : XBASIC is exceptionally good value for money. The extra commands it gives to ROM BASIC are on the whole very useful, lucidly explained in the manual and easy to use. Further, it is possible to add your own commands to produce a very personal BASIC. XBASIC is available from Level 9 Computing, 229 Hughenden Road, High Wycombe, Bucks., on tape price £15.00 or in ROM price £25.00.

RANDOM RUMOURS (& TRUTHS)

by S. Monger

Promises, promises, but where are the goods? Why do we keep hearing about all these wonderful new products only to find that they are not yet available (designed)? Last issue I mentioned the Nascom AVC, the Gemini (nee Quantum) I/O board and the Gemini RTC. All 'imminent'. As I write now the Gemini I/O board is available in limited quantities and the NM AVC and GM RTC are 'nearly' available. Please don't taunt us so much! (By the way, I was right (of course) about the AVC being 10"x8", and the reason for conflicting reports on whether there is text handling or not is because (1) it CAN handle text but (2) this text must be created under SOFTWARE control.)

Since I last wrote several items have appeared, some pre-announced and others surprises. Firstly Nascom have released 'MicroEd'. This is not a compact word processing package as the name implies, but is a Nascom 2 in a smaller box than Nascom 3, without the expansion frame and fitted with 8K of static RAM. This machine is intended as a competitor in the education market. How it competes against the newer, more compact, cheaper, more powerful (stand-alone) machines aimed at that market we will have to see. Looking at Nascom 3, shouldn't this be called Nascom 4??

From IO Research (formerly IO Systems) we can now obtain 'Pluto'. This board brings more RAM and power to your system than you dare imagine! 192K of dynamic RAM and a 16 bit (internal) 8088 processor. 'Pluto' works in much the same way as the Gemini GM812 IVC - you check a status port, send commands to a data port, then let the card get on with all the work. Quite a neat card, but at £399 (+ VAT) I'll wait for ERNIE to buy it for me. (And wait, and wait, and wait ....)

Then there is Gemini's GM813 combined Z80A CPU - I/O - 64K RAM board, all on an 8"x8" card. It has taken a while to actually materialise, but it is definitely now available. Richard Beal has been at work again and has produced RP/M V2.0 for the card. This apparently tidies up one or two 'features' of RP/M VO.1 whilst retaining full upwards compatibility, and adding a parallel printer driver and enhanced editing amonst other things. This board provides a lot for the money (£225 b&t + VAT) as far as 80-BUS/Nasbus cards go, but as you are reading this then presumably you already have an N1, N2, or GM811 and don't want another master CPU board (do you?). I don't, anyway! (Sounding a bit mean, aren't I?) [Ed. - No comment.]

On the software front Nascom have announced the 'real' availability of Pascal, and Gemini have announced impending availability of it too (COMPAS). As it happens both versions are written by the same guy! The first is BLS Pascal with the name changed to Nascom Pascal (and one or two mods.) and it runs under Nas-Sys. Gemini's version is somewhat more powerful, larger, (and more costly!) and runs under CP/M. As 80-BUS News seems to get its fair share of articles about Pascal I am sure that we will be hearing more of both of these, and also the independant supplier's new one, Hisoft Pascal 4. Imported from the same company (Polydata) by Gemini is also Polytext. No it isn't a wall filler, but a text editor that runs under Polydos. Rather nice too, with some features that Naspen/Diskpen/Gempen are all definitely lacking.

Talking of software, there is currently a Lucas/Nascom applications note (AN006) doing its round of the dealers and it contains generalised Centronics driver routines for use with the Nascom. Looks VERY similar to the now superceeded routines incorporated in the original SYS program written nearly two years ago for the Henelec driven Gemini G805 disk system. Fair enough ?
..... By the way, it seems to have grown a '(c) Lucas Logic' !!

✲ REAL ADVENTURE ✲
for nascoms

Adventures are possibly the most fascinating and addictive computer games, but until now REAL adventures have only been available for machines with disks or a lot of memory - and at high prices.

Not any more! Level 9 adventures need no disks: just Nas-Sys and 16K or 32K of memory, depending on version. The versions differ only in the size of text descriptions and in price: only £8 for 16K and £10 for 32K games.

In the games, the program acts as your eyes and ears to a fantastic world of weird settings, creatures from myth and legend, subtle puzzles, great risks and greater rewards... and, of course, magic.

Each adventure has over 200 individually described locations, and large numbers of treasures and creatures. In fact, a game may take you weeks to solve - but we provide a save command so that you can return to everyday life for a while and continue the game later; and we'll give a hint if you get stuck. However, playing the games could not be simpler - you just type in English phrases to tell the program what you want to do.

Level 9 adventures are written in a super-compact language called 'a-code' and use text compressed to a fraction of normal. This makes them possible.

Colossal Adventure ..... 16K/32K Machine Code ..... £8/£10

A complete, full scale version of the original classic mainframe game "Adventure". You'll find all of the treasures, creatures and rooms that you will have seen hinted at in computer magazines.

And the standard end-game, with only 2 locations, has been enlarged by a new ending with over 70 extra rooms! No one else gives you this bonus.

Adventure Quest (ready mid-August) . 16K/32K Machine Code ..... £8/£10

From the great forest, across the burning desert, up orc mountain, braving fire, marsh and illusion on a quest to save Middle Earth from Tyranny.

Use swords, spells and subtlety to combat opponents from dogs to demons, from ghosts to ghouls, and from wizards to 200-foot worms.

****

FREE P&P. NO VAT. Money back if not happy. Supplied on high quality tapes with full instructions. Send order or large SAE for details of our range of games (Asteroids, Missile Defence etc.) to:

Level 9 Computing
229 Hughenden Road, High Wycombe, Bucks. HP13 5PG

Note to Nascom 1 owners: you will need Nas-Sys & Cottis Blandford interface

Futura Software

Introducing the FUTURA graphics chip for the NASCOM.

It has 128 special graphics characters including :-
Arcade specials -
space invaders, galaxians,
pacmen & maze makers,
Adventure - treasure chests,
wizards, dwarfs, monsters,
Space Fantasy - robots, UFO's,
droids, aliens, starfighters,
Plus - skiers, race cars,
aircraft, rockets & bombs,
laser fire, flashes, explosions,
space debris, trees, mushrooms
and lots more ...

Only £5·99

*** Free Competition Win £25 ***
SAE for details.
Closing date 31/12/82.

Please add 50p P+P per order.
Send to : FUTURA SOFTWARE
63, Lady Lane, Chelmsford, Essex, CM20TQ.

PARKSTONE ELECTRICS

NASCOM
AUTHORISED
DEALER

GEMINI PRODUCTS
ALSO STOCKED

Computer Systems Built
for Special Applications

Personal PEARL
now available
on NASCOM
Create all your business
programs for less than £200

PARKSTONE
ELECTRICS
18, Station Road,
Lower Parkstone,
Poole, Dorset, BH14 8UB.
Tel: Parkstone (0202) 746555

syrtis SOFTWARE
presents two
★ASTOUNDING★
NASCOM 1 & 2 PROGRAMS
Adventure 16k
(NASCOM APPROVED)
&
Zap Z80 Macro Assembler

Adventure 16k

The mainframe game Adventure has become a classic. Now, for the first time, you can play this 16K version on your Nascom. Descend into the mysterious Colossal Cave, marvel at the wonders within, collect the treasures that lie buried deep inside; keep your wits about you as you face various hazards such as killer dwarves, fierce green snakes all preventing exploration of the cave.
As you wander around the scores of 'rooms' (directing your quest with English words as commands) you become deeply absorbed in this new and fascinating version of the 'Adventure' game. As demonstrated at the 1981 PCW Show and Compec '81.

ONLY £15

Zap Z80 Macro Assembler

We think this is the most advanced assembler yet available for the Nascom. Specially designed to get optimal use for programmers using the NAS-SYS series of monitors, source programs can be stored much more compactly due to its remarkable text compression feature. Its powerful macro facility allows easy duplication of repetitive sequences of instructions or the implementation of 'extra' instructions. Among its many features are Conditional assembly, Multi-line statements, Full error descriptions, Assembly to memory, memory plus displacement, or nowhere. Extra mnemonics including RCAL, SCAL, ROUT, CLA etc. Powerful pseudo-ops.
When writing Z80 assembly programs you need powerful software tools. The ZAP Z80 Assembler is one of these tools; it makes the assembly writer's task a great deal easier.

ONLY £15

A comprehensive manual, with examples, is supplied.

Other programs include Breakout, Games Tape, Text Editor, Relocater. Programs are supplied on cassette (please state tape format when ordering), and run on Nascom 1/2 in Z80 machine code, under NAS-SYS. Send cheque or PO. including .55p post and packing, or SAE for futher details, to:

Syrtis Software·23, Quantock Rd·Bridgwater Somerset TA6 7EG

Gemini Microcomputers

New Lower PRICES

GM 802 RAM Board

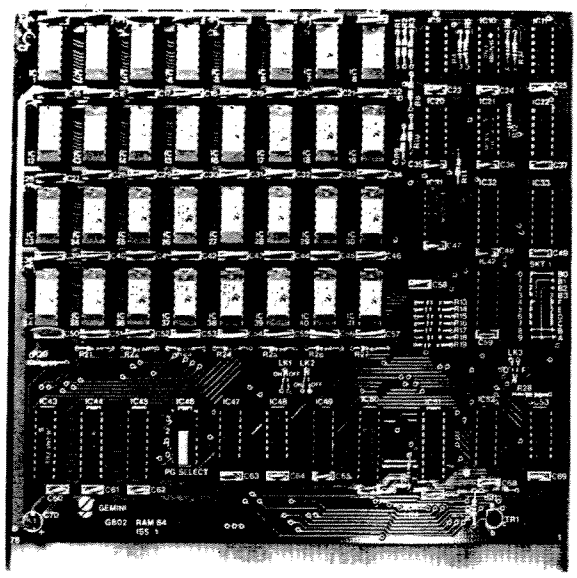
***64K Dynamic RAM**
***4MHz Operation**
***RAM Disable Function**

The Gemini GM 802 is a Dynamic random access memory card with a capacity of 64K bytes, allowing the total memory capability of the Z80 to be implemented on a single card. The card utilises an active delay line to give full 4MHz operation with no wait states required.

The GM 802 also includes logic for a pagemode operation which, when used with the appropriate software, allows up to four memory cards to be fitted in a single system.

Additionally this RAM card supports the 80 BUS/ NASBUS RAMDIS signal. This allows the user to have ROM in the system, the RAMDIS signal ensuring that there is no bus contention with the 64K RAM.

GM 802 – 64K RAM B & T – £125 + VAT



NOW £125

GM 812 IVC Board

***80 x 25 display format** ***160 x 75 pixel graphics**
***On-board Z80A** ***light pen input**
***Programmable character generator** ***Flicker-free display**

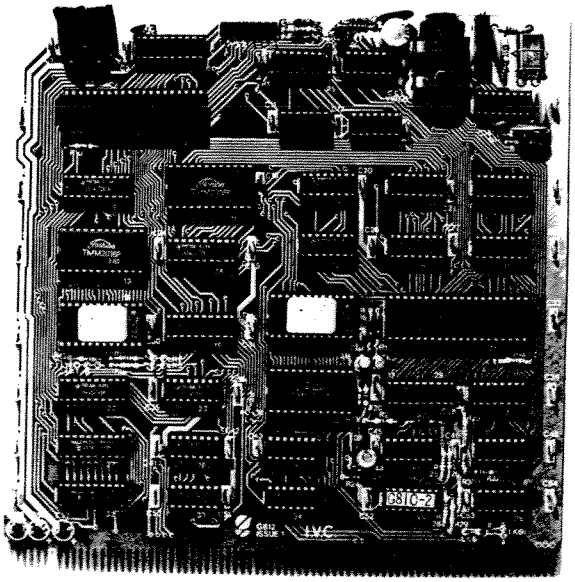
The GM 812 Intelligent Video Controller (IVC) board is an 8" x 8" 80-BUS/NASBUS compatible video display card. It features its own on-board Z80A processor to allow the video section of the computer to provide a variety of complex video functions without imposing any memory or processing overhead on the main CPU. Communication between the GM 812 and the host system is through the bus via two Z80 I/O ports. All reading and writing to the display is transparent, providing a flicker free display.

The standard screen format of the GM 182 is 80 x 25, which is the preferred format for most applications. The card also has an adjustable dot clock to provide alternative formats. The standard character set of 128 characters provides all upper and lower case alphanumerics plus additional characters. Lower case characters have true descenders. An additional 128 character shapes may be defined under software control. These can be used to provide inverse characters or pixel graphics characters giving a resolution of 160 x 75. Additionally the user may define his own characters for special applications.

The GM 812 includes both light pen and ASCII keyboard inputs. The light pen input can resolve a single character on the screen. The keyboard input provides a 32 character buffer to allow 'type ahead' without loss of characters.

The GM 812 is ideally suited for use in either a Multiboard system, or in a Nascom system running CP/M.

GM 812 – IVC Board B & T – £125 + VAT



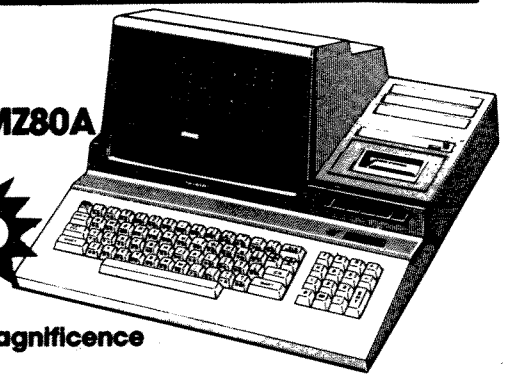
NOW £125

AVAILABLE FROM

YOUR MicroValue DEALER

MicroValue

THE NEW SHARP MZ80A



£475 + VAT

Electronic magnificence from Sharp

Z80A C.P.U. • 48K RAM • 4K ROM • Industry standard QWERTY keyboard with numeric pad • 9" GREEN C.R.T. • 1200 bd cassette • Music & sound • Real time clock • Enhanced BASIC • Full editing facilities • Internal expansion

FREE SOFTWARE! Home budget, bank reconciliation, SPACE INVADERS, STAR TREK, SCRAMBLE, bank loan calculator, mortgage calculator + 7 other games.
Educational — Geography, Maths, Spelling + 4 part BASIC tutorial.

Quantum HI-RES FOR MZ80K

High resolution plotting on your MZ80K down to a resolution of a single dot within a character cell.

A new BASIC is supplied with the following additional commands: LINE, WIPE, G SET, G RESET.

NEW FROM Quantum

Quantum HI-COPY FOR MZ80K

This combination of hardware & software not only allows printing of the full Sharp character set, but allows a full High Resolution print of the actual screen if used with the Hi-Res graphics option.

Available in 2 versions

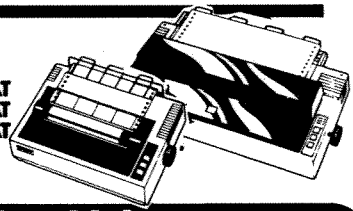
QUANTUM GP100A HI-COPY
SEIKOSHA GP100A, Interface, ROMS & screen dump BASIC.
£300.00 + VAT including printer

QUANTUM EPSON HI-COPY
Interface & ROMS & screen dump BASIC suitable for use with any EPSON PRINTER.
£90.00 + VAT excluding printer

SPECIAL OFFER **Quantum Hi-Res Only £70.00 if purchased with Quantum GP100A Hi-Copy**
Quantum MZ80K Games Packs 1–5 £5.00 + VAT each

EPSON PRINTERS

Epson MX80 Type III .... £349 + VAT
Epson MX80 FT Type III £389 + VAT
Epson MX100 Type III .. £499 + VAT



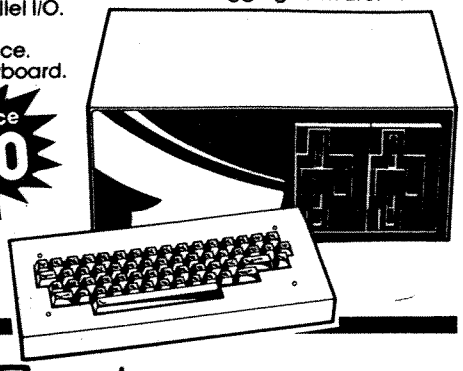
Gemini GALAXY 1 CP/M COMPUTER SYSTEM

A Multiboard based 80-BUS computer

HARDWARE
- ★ Twin Z80A CP/M System.
- ★ 64K Dynamic RAM.
- ★ 800K Disk Storage (Formatted).
- ★ 80 × 25 Screen Format.
- ★ Inverse Video.
- ★ Prog Character Generator.
- ★ 160 × 75 Pixel Graphics.
- ★ Centronics Parallel I/O.
- ★ RS232 I/O.
- ★ Light pen interface.
- ★ 59-Key ASCII Keyboard.

SOFTWARE INCLUDED
- ★ Full 64K CP/M 2.2 with screen edit facility.
- ★ COMAL-80 structured BASIC.
- ★ GEM ZAP Assembler/ Editor.
- ★ GEM PEN Text Editor.
- ★ GEM DEBUG debugging software.

MicroValue price **£1,450 + VAT**



Suggested monitor for use with the Galaxy. **£150 + VAT**

QUIBS-Quantum INTEGRATED BUSINESS SYSTEM

A business accounts package developed for the Galaxy, menu driven.

1. **SALES LEDGER** full VAT reports, statement, credit note + invoice facilities.
2. **PURCHASE LEDGER** full VAT reports, statements, remittances.
3. **NOMINAL LEDGER** 250 analysis heads, trial balances, accruals & repayments.
4. **STOCK CONTROL** costing reports, price lists, etc.,

The system is fully integrated. Comprehensive audit trails are printed. Specially developed for Multiboard based systems.

£400 + VAT Specify disk format when ordering

Quantum DATAFLOW

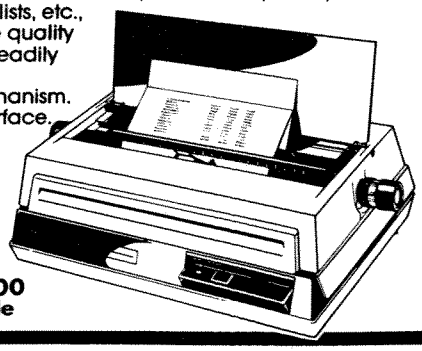
A Data base management and information retrieval package. Allows searching, sorting, report printing, file printing and label printing. Anything which is filed manually can be filed more efficiently with DATAFLOW. For all multiboard CP/M systems.

£120 + VAT Specify disk format when ordering

DAISY WHEELS ARE DOWN — ONLY £485 + VAT

For less than the price of some dot matrix printers, the Smith-Corona TP-1 brings the benefits of daisywheel printers within the reach of most micro users. Now letters, documents, forms, invoices, reports, price lists, etc., can be printed with the quality that until now was not readily affordable.

- ★ Simple reliable mechanism.
- ★ Serial or Parallel interface.
- ★ IEEE option.
- ★ Single sheet and fanfold paper.



FANFOLD PAPER
2,000 SHEETS £16.00
Music Paper available

YOUR LOCAL MicroValue DEALER

All the products on these two pages are available while stocks last from the MicroValue dealers listed below (Mail order enquirers should telephone for delivery dates and post and packing costs.) Access and Barclaycard welcome

AMERSHAM, BUCKS.
Amersham Computer Centre Ltd.,
(formerly Interface Components Ltd.)
Oakfield Corner, Sycamore Road.
Tel: (02403) 22307. Tx: 837788.

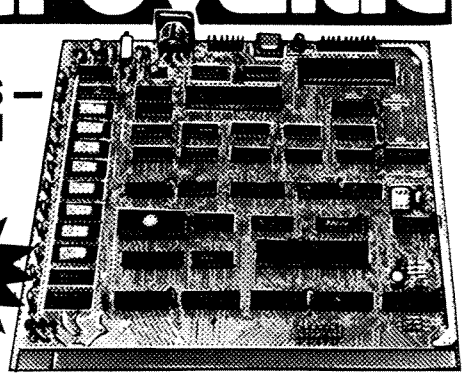
BRISTOL
Target Electronics,
16 Cherry Lane.
Tel: (0272) 421196

EDINBURGH Computer
Interfacing & Equipment Ltd.,
19 Roseburn Terrace.
Tel: (031) 337 5611

EGHAM, SURREY
Electrovalue Ltd.,
28 St Judes, Englefield Green.
Tel: (0784) 33603. Tx: 264475

MicroValue

80-BUS PRODUCTS — NEW FROM Gemini



ONLY £225 + VAT

GM813 CPU/64K RAM Card

The Gemini GM813 is a new 80-BUS compatible CPU card incorporating 64K dynamic RAM and utilising the powerful Z80A microprocessor running at 4MHz. Extended addressing and page mode facilities allow for future memory expansion up to 2 megabytes. Input and output capabilities include both programmable serial and parallel interfaces — RS232, 1200 baud CUTS cassette interface and the Z80A PIO. When used with the GM812 video card, the GM813's unique RP/M monitor allows the creation of cassette or EPROM based programs or files which are upwards compatible with a disk based CP/M system.

OTHER 80-BUS PRODUCTS FOR nascom & Gemini SYSTEMS

GM816 GEMINI I/O BOARD

The new GM816 Gemini I/O board takes a unique approach to the problems of interfacing your Nascom or Gemini Multi-board to external devices. This 80 Bus and Nas-Bus compatible card is supplied fully built, populated and tested and includes three Z80 PIOs, a CTC and a Real Time CLock with battery back-up. In addition, a range of 'daughter' boards that attach straight to the I/O board are under development, catering for a wide variety of interfacing requirements.

GM816 Gemini I/O board MicroValue price — £125 + VAT

Prototyping daughter board MicroValue price — £18 + VAT

EV814 EV COMPUTERS IEEE-488 BOARD

The EV Computers' IEEE-488 card is an 80 Bus and Nas-Bus compatible card designed to fully implement all IEEE-488 interface functions. This built and tested card gives the user a very cost effective and versatile method of controlling any equipment fitted with a standard IEEE-488 or GP1B interface.

MicroValue introductory price £140 + VAT

MP826 MICRODE 32K BATTERY BACKED STATIC RAM CARD

Provides 32K bytes of battery backed RAM. Page Mode is fully supported offering 1 × 32K or 2 independent 16K pages of memory retained for over 40 days without external power.

MicroValue price £170 + VAT

SOFTWARE FOR THE Gemini MULTIBOARD SYSTEM

COMAL 80* — The extended BASIC with powerful PASCAL structures at **£100 + VAT**
GEM PEN* — A comprehensive text editor and text formatting package at **£45 + VAT**
GEM ZAP* — A very fast Z80 assembler with comprehensive screen editing at **£45 + VAT**
GEM DEBUG — A debugging utility program including trace and disassembly features. **£30 + VAT**
COM-PAS — An interactive PASCAL system with on-screen editor. Generates Z80 machine code. **£150 + VAT**
COPY SB — Allows transfer of programs and files between Gemini DDDS and Superbrain DD & QD formats. **£30 + VAT**
LIST/REPAIR — LIST replaces the CP/M TYPE command and provides paging, headings, line numbering, etc. REPAIR is for G809/G815 systems and allows reading and writing of individual disk sectors to assist recovery of lost data. **£25 + VAT**

*Available on cassette or disk

COMING SOON — AP/L

SOFTWARE FOR THE GEMINI DISK SYSTEM FOR NASCOM 1 OR 2

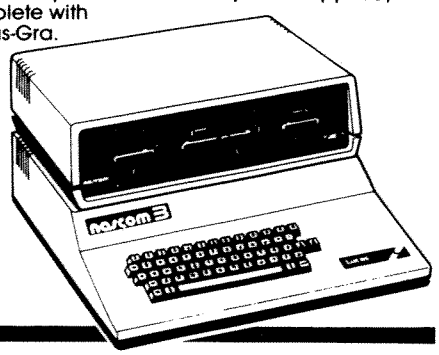
Choose from either the industry standard CP/M 2.2 D.O.S. or POLYDOS — a unique, versatile and well presented DOS that includes an editor, assembler and adds disk commands to the Nascom BASIC.

CP/M 2.2 — for use with GM805 ..... **£100 + VAT**
POLYDOS 1 — for use with GM805 ..... **£90 + VAT**
CP/M 2.2 — for use with GM815 ..... **£100 + VAT**
POLYDOS 2 — for use with GM815 & GM809 . **£90 + VAT**

nascom PRODUCTS & PERIPHERALS

NASCOM 3 AVAILABLE FROM MICROVALUE

Based around the successful Nascom 2 computer, this new system can be built up into a complete disk based system. Supplied, built and tested complete with PSU, Nas-Sys 3 and Nas-Gra.



48K system MicroValue price — £499 + VAT

Dual floppy disk unit (0.7 MB storage) MicroValue price — £685 + VAT

CP/M 2.2 MicroValue price — £100 + VAT

MICROVALUE'S 'NASCOM SPECIAL'

MV-N2 SPECIAL — comprises of a Nascom 2 kit, Nas-Sys 3, Nas-Gra Graphics ROM, Bits & PCs programmers aid, Gemini GM807K 3AMP PSU kit, Gemini GM802K 16K RAM kit (expandable to 64K) and a Micro mother board. **Normal RRP over £405 MicroValue price £340 + VAT save £65**

NASCOM 2 KIT £225 + VAT **Built & Tested £285 + VAT**

80×25 VIDEO FOR nascom

Nascom owners can now have a professional 80 × 25 Video display by using the Gemini G812 Intelligent Video Card with onboard Z80A. This card does not occupy system memory space and provides over 50 user controllable functions including prog character set, fully compatible with Gemini G805 and G815/809 Disk Systems. Built and tested. **£140 + VAT**

QM180 NASCOM GRAPHICS KIT — gives Nascom 2 graphics capability to your Nascom 1. **only £20 + VAT**

NASCOM 1 PRINTED CIRCUIT (inc. parts list) **£25 + VAT**

SOFTWARE FOR nascom

POLYTEXT — a text editor/formatting package for use with POLYDOS **MicroValue price £35 + VAT**
MATHSPAK — Double precision maths package on tape. **MicroValue price £13 + VAT**
MATHSPAK HANDLER — Used in conjunction with MATHSPAK **MicroValue price £9.95 + VAT**
COMMAND EXTENDER — For use with MATHSPAK it extends BASIC's reserve word list **MicroValue price £9.95 + VAT**

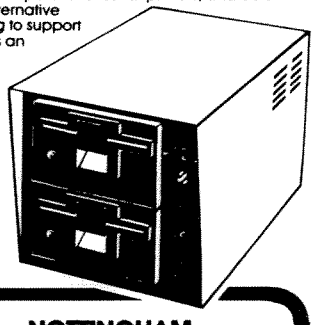
LOGIC SOFT RELOCATER — A software relocating package which allows disassembly and reassembly from anywhere on the memory map. **MicroValue price £13 + VAT**

NASPEN	RRP £30 + VAT . MicroValue price £20 + VAT
Nas-Sys 3	RRP £25 + VAT . MicroValue price £20 + VAT
NasDis D-Bug (EPROM)	RRP £50 + VAT . MicroValue price £30 + VAT
NasDis D-Bug (TAPE) ..	RRP £40 + VAT . MicroValue price £20 + VAT
Imprint	RRP £30 + VAT . MicroValue price £20 + VAT
Bits & PCs Prog Aid ...	RRP £28 + VAT . MicroValue price £20 + VAT

Gemini DISK SYSTEM FOR nascom

GM809 — full Nas-Bus floppy disk controller card — drives up to 4 drives — optional 8" expansion — £125 + VAT.
GM815 — Double density disk system.

With a thousand in daily use, the Gemini Disk system is now the standard for Nascom and Gemini Multiboard systems. Single or twin drive configurations are available, giving 350K storage per drive. The CP/M 2.2 package available supports on-screen editing with either the normal Nascom or Gemini IVC screens. parallel or serial printers, and auto single-double density selection. An optional alternative to CP/M is available for Nascom owners wishing to support existing software. Called POLYDOS 2, it includes an editor and assembler and extends the Nascom BASIC to include disk commands.



Single drive system (G809, G815/1) £450 + VAT

CP/M 2.2 package (G513) £100 + VAT

Double drive system (G809, G815/2) £675 + VAT

POLYDOS 2 £90 + VAT

LONDON W2 Henry's Radio, 404 Edgware Road. Tel: (01) 402 6822 Tx: 262284 (quote ref: 1400)

NOTTINGHAM Skytronics, 2 North Road, The Park. Tel: (0602) 45053/45215

MANCHESTER E.V. Computing, 700 Burnage Lane, Burnage. Tel: (061) 431 4866

WETHERBY, W.YORKS Bits & PC's 4 Westgate. Tel: (0937) 63774

FAREHAM, HANTS. Allegro Electronics Ltd., Newgate Lane Industrial Estate, Newgate Lane. Tel: (0329) 289123

LEEDS Leeds Computer Centre, 62 The Balcony, Merrion Centre. Tel: (0532) 458877

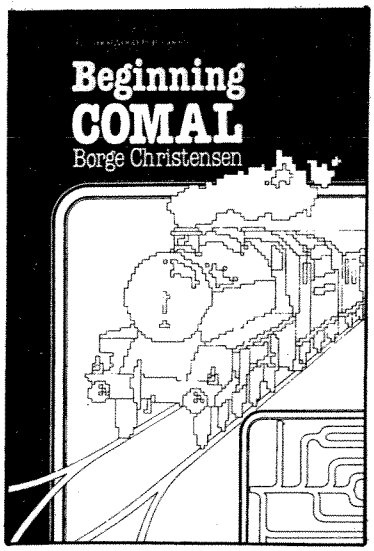
STRUCTURED PROGRAMMING WITH COMAL

ROY ATHERTON
Director of the Computer Education Centre, Bulmershe College of Higher Education, Reading, Berkshire
Adviser on Computer Education to Berkshire County Council, and Senior Editor of *Comal Bulletin*.

Breaks new ground with an exposition of this innovative language, drawing together the best qualities of COMAL and BASIC under one roof. Organised into a lucid learning order, comparable to a spiral, the topics are treated piece by piece. New material is introduced at the appropriate moment, taking care not to exhaust the reader with unnecessarily rigorous explanations and syntactic detail. Roy Atherton's approach is creative and imaginative, teaching the subject in a free-wheeling, yet professional, way. His affinity with COMAL is apparent throughout.

ISBN 0-85312-416-7 Library Edition 266 pages **£18.50**
ISBN 0-85312-423-X Student Edition **£6.90**

Published by ELLIS HORWOOD LIMITED, Chichester
Distributed by John Wiley & Sons Limited, Chichester



BEGINNING COMAL

BORGE CHRISTENSEN
Principal Lecturer, Tonder College of Higher Education, Tonder, Denmark

A book which offers newcomers to programming a complete COMAL program library, so that the student does not have to sit and type four-line programs with no bearing on real-life problems. It enables the beginner to read and analyse programs, modify and extend them, and teaches him the fundamentals of programming in a straightforward manner. No previous knowledge of computers is necessary.

The programmes are based on applications of computers to games, general administration, programmed learning, and even small business. Some are highlighted by structured diagrams, introducing a powerful way of picturing programmes — a feature quite new to books on programming at this level.

ISBN 0-85312-435-3 320 pages **£10.00**

Published and distributed by
ELLIS HORWOOD LIMITED, Chichester

For further information please write to

Department C
Ellis Horwood Limited, Publishers
Market Cross House,
Cooper Street,
Chichester, West Sussex
PO19 1EB