

June/July/August 1980

INMC NEWS

80

Issue: 1

BUMPER "We're still around" ISSUE

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

CONTENTS

Page 1	This is it.
Page 2	Chairman's Bit.
Page 3	Long Live NASCOM.
Page 4	Letters to the Editor.
Page 10	NASCOM Rescue Bid.
Page 11	Print HL in Decimal.
Page 12	Checksum Routines.
Page 14	ZEAP Output to Printer.
Page 15	Conversion of T4 to Nas-Sys.
Page 16	Make Your Own Characters.
Page 18	Hex Codes/Graphics/Reserved Words.
Page 19	Teach Yourself Z80.
Page 22	S100-Readers' Replies.
Page 24	Using the PIO.
Page 26	Hardware Reviews.
	Nascom I/O Board.
Page 28	Port Probe & Key Pads.
	3M Cassettes.
Page 29	CUTS Cassette Interface.
Page 30	BASIC at 4MHz.
Page 31	Notes on PASCAL.
Page 32	Book Reviews. advised by D. R. Hunt
	Z80 Books.
Page 33	BASIC Books. advised by D. R. Hunt
	Classified Ads.
Page 34	Doctor Dark's Diary. - 1621 and others
Page 36	WHAT ?
Page 37	Index to INMC Newsletters - 1 to 7.
Page 39	Computers ?
Page 40	Assemblers and Nas-Sys. by D. R. Hunt
Page 41	HALT and the Nascom.
Page 42	Generated BASIC Programs. Hunt
Page 44	Cassette Hints.
	Cassette Reliability. by D. R. Hunt
	Cut That Noise.
Page 45	2400 Baud on Nascom 2. D. R. Hunt
Page 46	ZEAP 1.1 Files to 2.0 Files.
	8 Amp PSU - A Warning. D. R. Hunt
Page 47	Nas-Sys 3.
Page 48	Laurence Fights Back. D. R. Hunt
Page 49	Our very own "IMPERSONAL" column.
Pages 43/50/51	Advertisements.

PLEASE NOTE. Our address is now:
INMC80, c/o Oakfield Corner,
Sycamore Road, Amersham, Bucks. HP6 6SU.

Non-members

90p

PLEASE ALSO NOTE. This address is used by INMC80 purely as a postbox. We cannot arrange to send you leaflets or answer all your sales and technical queries. Contact NASCOM or your NASCOM Distributor for these.

'HE' speaks

CHAIRMAN'S BIT

=====

Not so long ago you will have received a letter from us stating that Nascom had called for a Receiver to handle its affairs. At the same time the letter said we'd all resigned. Implication: we were all off to the Bahamas with the INMC funds. Well sorry, but it's not true, we are back again, and to prove it, with a bumper issue containing about 10 pages more than usual.

In the past we have relied on Nascom for secretarial support, which has been withdrawn on the arrival of the new 'Powers That Be' over at Chesham. However, a friendly Nascom distributor has agreed (at least temporarily) to give us a home, help us keep the paperwork straight and deal with programs, subscriptions and distribution. We won't advertise here (it won't take you long to find out who he is), but Ta folks.

Now to the INMC itself, we've got to change the name, to get round any legal nasties and break entirely with Nascom, so if you haven't noticed already, this newsletter is called INMC80 issue 1. You know and I know its INMC 8, but that's the way it goes. The '80' stands for 1980, or 280, or the square root of the number of members, or something. Cheques payable to INMC80 in future please. Secondly, we must register ourselves as a charity or a 'Ltd' company pretty shortly, so we can get the benefit of VAT refunds etc., we haven't decided which yet, but if anyone has any vigorous objections to one or the other, or can offer advice, let us know.

As we mentioned last newsletter, funds aren't quite what they could be. The cost of newsletter printing and distribution costs rather more than the annual subscriptions. Hint !!! We had hoped to make up this deficit with advertising. Where are you ??? Only two this time, and we need five pages worth to keep the books in balance. If you are passing your friendly distributor, wave your copy of this newsletter under his nose and tell him we need his custom (and you'd naturally buy it if you saw in OUR pages). In the meantime, our friendly distributor has agreed to stand any liability we may incur for 12 months, or until the funds allow us to be totally self supporting (which ever is the sooner).

There has been some re-organisation of the committee; there will be an honorary executive committee consisting of Chairman, Secretary/Editor and Treasurer; and an editorial committee. All the existing loonies have agreed to continue to stand in their original capacities, we will acquire a Nascom representative (for the inside information), and we need a Dodo. If you are wondering what a Dodo is, well if you don't follow every piece of golden prose to be found on these hallowed pages, then it could be you. No offence intended, but the committee needs someone who has little practical understanding of Nascoms and who can argue with the rest of the editorial committee. His job will be to stop us all making this newsletter only readable by the elite.

Sits. Vac.

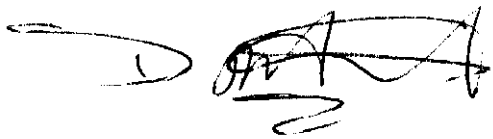
One Dodo required. Nascom owner essential (fitted with CUTS interface),

should have Naspen. Experience required - None (preferable). Must live in the South East, but committee meetings usually held (evenings) in North London and no travel expenses offered.

Articles wanted

To keep this newsletter going we need articles, preferably readable and understandable, on any Nascom related subject. You'll notice from this issue that we have had another couple of volunteers, but we need more. The more you put into this thing, the more you will get out of it.

So to sum up, the news is; the INMC still exists, and we intend (with your help) to make the Newsletters bigger, better and more informative. But we need more help from distributors (advertising), committee (one Dodo), and everyone else (articles/reviews/comments).



Dave Hunt

NASCOM IS DEAD - LONG LIVE NASCOM

=====

The story so far. Nascom started from humble beginnings in November 1977, as an offshoot of a semiconductor distributor, Nasco Ltd. They did well, too well. Expansion requires money for development and marketing, and money comes from banks and places like that. You try walking into a bank (any bank) and try to get a loan without either previous credit history, or on the promise that what you intend to do with the money will make the bank's fortune. When you pick yourself up off the door step, consider Nascom's problem.

Nascom 2 was an expensive exercise, financed by Nascom, but money to get it into production was provided by the 'city'. Nascom's return on Nascom 2 was immediately frustrated by component supply difficulties and instead of making a healthy profit the 'give away 16K RAM' exercise was born. Their expansion came to an abrupt stop when the 'city financial institution' decided it was not prepared to continue to allow Nascom sufficient working capital to continue. Nascom were presented with no choice but to call in a Receiver. See Guy Kewney's article on page 42 of July 80 edition of PCW.

What now ? The Receiver is keeping Nascom going until such time as a buyer is found. He is even getting the Floppy Disk Controller, the Programmable Character Generator and the Colour Board into production. We understand that there are a number of potential purchasers in the offing. The future of Nascom is very much at the mercy of whoever ultimately buys it, but some predictions can be made. Any purchaser is unlikely to 'asset strip' as Nascom has few tangible assets to strip, therefore anyone buying it can only do so with the intention of keeping it going. However, the purchaser is not committed to produce those Nascom items still waiting to see the light of day. For instance, the 'System 80' box may be restyled and further delayed.

In any event, it seems that Nascom will continue in some form or other, and whilst the product exists we will do our best to support it.

Letters

Dear Editor,

One night, whilst on a recent visit to California, I happened to be looking for somewhere to sleep. Being particularly hard up, and it being a fine night, I had a nice comfy packing case in mind. I was attracted to a pile of boxes full of shredded paper behind a factory displaying a trade name in the form of a stylized 'Z' picked out in blue. I must congratulate the owners of the factory on the quality of their shredded paper, as I spent a very comfortable night, and left the area the following morning thoroughly refreshed. It wasn't until some time later that I discovered 'The Paper'. It was a single sheet, that the shredding machine had somehow missed, which must have worked its way into my pocket as I slept. Being a thrifty sort, and paper always coming in useful, I kept it.

On my return to the UK, I (naturally) set to, to catch up on my back issues of INMC news, and was intrigued by Dr. Dark's 'Search for the Missing Op-code'. 'The paper' came to mind, as there was something basically familiar about it. Of course !!!! It was one half of a page from a programming manual, the mnemonics were there, but the shredder had had the op-codes. I offer the following to Dr. Dark, in the hope that he can discover the op-codes which go with these very useful instructions:

ARN	Add and reset to non-zero	LDPB	Load, print & blush
BTI	Blow trumpet immediate	LDTR	Load tape & run away
CCS	Chinese character set	LDS	Load tape & scramble
CRN	Convert to roman numerals	MTI	Make tape invalid
DAC	Divide and conquer	NPN	No program necessary
DAR	Develop address out of range	PI	Punch invalid
DM	Destroy memory	PO	Punch operator
DWS	Destroy write protect switch	PS	Print and smear
EIO	Execute invalid op-code	RBT	Read blank tape
EO	Erase operator	R n	Reset to n
FBJP	Float bus and jump	ROE	Randomize op-code
FSG	Fill screen with garbage	ROM	Read operator's mind
GO	Generate own operand	RPM	Read programmer's mind
HCF	Halt and catch fire	RT	Reduce thruput (Ugh!)
HLP	Dial 999	SCT	Select and chew tape
IA	Illogical AND	SD	Scramble data
IOR	Illogical OR	SIO	Scramble I/O
JPLO	Jump on lost operand	SML	Shift memory left
JPPF	Jump on power failure	SMR	Shift memory right
JSO	Jump on sleepy operator	T00	Execute operand only
LDBT	Load blank tape	TPD	Triple packed decimal
LDCM	Load then clear memory	TPN	Turn power on
LDDRC	Move about, continuous	TPO	Turn power off
LDID	Load invalid data	WM	Write to monitor
LDNT	Load noise tape	XOM	Execute NOP & hangup

Regards, Divad Tnuh

With apologies to BYTE.

PARDON ?

=====

Dear Sir,

For some time, rumours have been circulating about the appalling conditions in which some Nascoms are expected to work. There have even been some stories of them being EX-PORTED. How would you like your ears cut off, I ask you ?

Yours faithfully,

J.R. Keneally.
Dorset.

BASIC PROGRAM

=====

Dear Sirs,

Please find enclosed a bit of a program that I find fairly useful for games and graph joining etc. It is for Nascom graphics and joins together any two points on the screen. It does NOT test for off screen positions.

I have found that it can speed up quite a lot of programs which draw graphs or circles etc using scientific functions with larger spacings than for continuous graphics, because the number of points available and the definition of Nas graphics is fairly low anyway.

The second point is a bit of a moan about the Microsoft Basic, which, as far as I can tell, is only valid on the Nascom implementation. This is when listing reserved words (graphic characters) within quotes on a print line and those after a REM statement are printed as their command equivalents, making it impossible to edit them. I have had to write a machine language routine to let me enter them and edit them then dumping to tape to be read in as lines in a program. Not to worry version 123.4 will no doubt cure it.

All the same a magic beastie, now after a floppy from California.

Cheers,

Martin Taylor.
Leconfield.

```
1 .      PRINTER RESTRICTIONS (CREED 7E)
2 .      -----
3 .
4 .      CAN ONLY PRINT COLONS NOT SEMICOLONS
5 .      MULTIPLY PRINTS AS  .X.
6 .      EXPONENT PRINTS AS  .XX.
7 .      LESS THAN           .LE.
8 .      MORE THAN           .GR.
```

LIST

```
65504 REM ++++++
65505 REM +   VECTORS   M. TAYLOR   +
65506 REM + X1,Y1 AND X2,Y2 ARE JOINED +
65507 REM + AS TWO GRAPHIC POINTS +
65508 REM + FOR USE WITH NASCOM GRAPHICS +
65509 REM ++++++
65510 IF X1=X2 AND Y1=Y2 THEN RETURN
65511 SET(X1,Y1): SET(X2,Y2)
65512 DX = ABS(X1-X2): DY = ABS(Y1-Y2)
65513 IF DX .LE. DY GOTO 65517
65514 FOR X=0 TO X2-X1 STEP SGN(X2-X1)
65515 Y=Y1+((Y2-Y1)/DX).X.ABS(X)
65516 SET(X+X1,Y):NEXT:RETURN
65517 FOR Y=0 TO Y2-Y1 STEP SGN(Y2-Y1)
65518 X=X1+((X2-X1)/DY).X.ABS(Y)
65519 SET(X,Y+Y1):NEXT:RETURN
OK
```

Ed: Thanks for the Basic subroutine which is very efficient. You don't need the outer brackets in Lines 65515 and 65518.

We have come across several minor bugs in the Nascom Basic, but luckily none of them have any effect on running programs. The bug you mention is quite annoying, we agree. To be fair to Nascom, this particular bug is, we think, in the 8080/Z80 Microsoft 8K Basic, and was not added by Nascom.

MORE ON BASIC
=====

Dear Sir,

I have enjoyed reading the INMC newsletters but would naturally like to see more items for the Nascom 2. Obviously, you can only do this with readers help and I profer my contributions below:-

1. The Nascom suggestion for writing to line 16 is to first write on line 15 and then transfer this to line 16. This does work but gives the appearance that the program has gone wrong - and we all know our programs never do that! The following routine writes a string C\$ directly to the top line:-

```
10 C$ ="THIS IS THE LINE TO WHICH YOU CAN'T WRITE"
20 FOR A = 1 TO LEN (C$)
30 POKE 3017 + A, ASC(MID$(C$,A,1))
40 NEXT
```

Working on the assumption that 'anything POKE can do DOKE can do quicker' I tried to improve the above, but without success. I found it necessary first to make LEN(C\$) exactly divisible by 2 and then to format the ASCII codes of the two characters currently being dealt with by first reversing them and then converting to a single decimal number. Having provided the approach to steer well clear of I await suggestions from my peers.

```
45 C$ = "DO IT THIS WAY IF YOU'RE DAFT"
50 IF LEN(C$)/2 < > INT(LEN(C$)/2) THEN C$ = C$ + " "
60 FOR A = 1 TO LEN(C$) STEP 2
70 B = ASC(MID$(C$,A+1,1)) * 256 + ASC(MID$(C$,A1))
80 DOKE 3017 + A,B
90 NEXT
```

Some "little known facts":

If you have to write a long DATA table or whatever consisting of very similar lines, you can type in the first line and press ENTER, then, using the cursor control keys, overwrite the line number with the number of the next line required and press ENTER, etc, etc. A LIST command will then show your series of identical lines for editing as necessary.

When entering a line of the full 48 characters the act of typing the final character places the cursor on the following line and pressing ENTER has no effect. It is only necessary, however, to place the cursor back up on the line - at any point - for a ENTER to enter the whole line.

I have added a Bits & P.C.'s hex pad to my Nascom 2, which works well for hex and integers but lacks a decimal point. The eventual solution must be a hardware fix, but my interim solution is a short series of Nas-Sys instructions which copy out the keyboard table from ROM into that portion of RAM not accessed by BASIC, switch the values for "." and "F" and gives Nas-Sys the address of the new table. Full credit goes to the writer of Nas-Sys for leaving the door open!

```
At switch-on:
C 059E 0C80 0060      (Copy out table)
M 0CAE
43 / 0CC6
21 / 0C6F              (Switch values)
80 0C.
```

```
After a Reset:
M 0C6F
80 0C
```

Yours faithfully,

D. Walker.
Windsor.

PIO HELP NEEDED
=====

Dear INMC,

Could you include some programming hints on the use of the PIO on Nascom 1 in some future issue? One useful example would be a traffic lights program driving a LED display via the PIO. There is a lot of

interest in Nascoms in Bradford College where I teach. Hopefully we will equip an MPU workshop with Nascoms (if cuts permit) for teaching MPU applications.

Can you recommend a book on interfacing MPU's to external equipment ? Something on the lines of a tutor text would be ideal.

Yours faithfully,

P. Nurse.
Bradford.

Ed: We put several articles on the PIO in issue 6, but a short article describing the software and hardware needed to control a row of LED's should be easy to find. Please will an INMC member send one in, and can anyone help on suggestions for books describing interfacing that are simple to understand ?

MASTERMIND MODS
=====

Dear INMC,

I'm sure a lot of you and other Nascom owners have tried the Mastermind game by D. Ritchie, which you have featured in the mag. This is one of the best games around, the Nascom usually beats me, but the only let down in the program is the poor random number selection for the 'computers guess', which after a few goes is predictable, 0123, 5567, 3455, typical of its guess's. I feel this is due to the simple use of 'ED 5F', location 0C9F. I have worked out a simple random number generator which works a lot better, using the rnd. no. generator of 'B-Bug'. (What's B-Bug ? -Ed.)

The modifications are:

0C9D	06 04	LD B, 04
0C9F	E5	PUSH HL
0CA0	21 B0 0C	LD HL, 0CB0 (workspace)
0CA3	3E 07	LD A, 07 (max. no.)
0CA5	CD 7A 04	Call B-Bug Rnd. No.
0CA8	E1	POP HL
0CA9	77	LD (HL),A
0CAA	23	INC HL
0CAB	10 F2	DJNZ 0C9F (repeat for 4 nos.)
0CAD	C9	RET

Have a go, and I think you'll find it's better.

Yours faithfully,

G. Benson
Lichfield.

DOCUTEMNATION ERUR
=====

Dear Sir,

I am a recent member of INMC having just built myself a NASCOM 2. The following may possibly be of interest to your readers.

There is a misprint on page 9 of the 8K BASIC Software manual in the NASCOM 2 documentation. In fact, this version of BASIC does NOT support user defined functions with more than one argument. Thus, DEF FNAVE(V,W) = (V+W)/2 is not valid, although the other example given is of course single and OK.

I hope that this information may be of use to others !

Yours faithfully,

H.J.J. Berridge.
Crowthorpe.

USR SUBROUTINES
=====

Dear Sirs,

Many thanks for your excellent newsletter. I used the machine code subroutine for 8K Basic as in your Issue 4, and lo & behold, ?FC error. The fault was mine, not yours; in case it is useful to others I will explain what I should have read in the manual.

Before a USR is used, the address of the subroutine to be called must be poked into a 2-byte location known as USRLOC. For those of us with the tape Basic, assembled at 1000H to 2FFFH, that location is 3004 & 5H (decimal 12292 & 3).

Similarly the address of the routine which sets the value to be returned as USR() as the current contents of the A & B registers in 100A & BH for Tape Basic users.

A useful feature which could easily be incorporated in Basic, but is not available in any version I know of, would be to allow variables to be in Hexadecimal. This would make POKE-ing & DOKE-ing much simpler. I enclose a pair of subroutines in Basic for the library, to simplify it, but machine code subroutines accessing the Data statements would be much more efficient in time and space. If only the folk who write interpreters would give us the source code.

I look forward to seeing some programs in your organ of more use or interest than games.

Yours faithfully,

Robert Edlin.
Nottingham.

Rescue Bid

This letter was received recently by the INMC. We understand that similar letters have been received by dealers and individuals. We reprint the letter here, but think that the INMC must remain strictly neutral.

NASCOM RESCUE BID
=====

3, Bishopstone Close,
Golden Valley,
Cheltenham,
Glos.
GL51 0UD.

14th June 1980

Dear INMC Member,

You are probably aware that Nascom Microcomputers is in the hands of a receiver who is preparing to sell the company in order to satisfy a financial institution who has recalled a loan.

If the company is sold into the hands of competitors it will certainly closed down by them and the assets stripped.

YOU CAN HELP to save Nascom from extinction.

This is a viable British company with excellent products and more on the way. They have a temporary cashflow problem and just need help to get over the hump and on their way. They are not bankrupt.

All that is required is the formation of a 'rescue company' to buy Nascom from the receiver. This company would own Nascom and provide the necessary finance to allow them to function successfully.

The 'rescue company' would obtain its capital by the issue of shares of 50 pence with a minimum shareholding of 10.00. Larger shareholdings would be permitted, and are of course desirable. However, the minimum holding of 20 shares would permit those with limited funds to participate as well as the more affluent. There are about 20,000 Nascom owners and 32 distributors. It is in the interests of all of us to ensure that Nascom survives.

The Receiver thinks my proposal viable and has given me a copy of the sale proposals and relevant accounts.

In order that I may assess the amount of support would you please complete the attached slip and send it to me with a stamped self addressed envelope. This will enable me to send you further news of the progress of the rescue operation.

Yours sincerely,

J. G. Margetts INMC Member.

=====

Nascom Rescue Bid

I would/would not support the rescue bid.

I would probably spare to buy shares.

Signed

Date

NOTE: Completing this slip does not commit you in any way.

Do not forget to enclose a self addressed envelope with this slip.

Machine Code

PRINT HL IN DECIMAL
=====

J. Addey/R. Beal

Mr. J. Addey has sent us a useful subroutine which prints HL in decimal. We have modified it for Nascom/Nas-Sys, and have also saved quite a few bytes by tidying it up and converting to full use of the Z80. Here it is complete with a little program which demonstrates the use of several Nas-Sys routines. Run the program at 0D00H and type in a value in HEXadecimal. Errors are trapped. This program is a good starting point for beginners to analyse.

ZEAP Z80 Assembler - Source Listing

```

0010 ; DEMONSTRATION PROGRAM
0D00      0020 DEMO   ORG   0D00H
0D00 0063  0030 ZINLIN EQU  63H
0D00 0079  0040 ZRLIN EQU  79H
0D00 006B  0050 ZERRM EQU  6BH
0D00 0060  0060 ZARGS EQU  60H
0D00 006A  0070 ZCRLF EQU  6AH
0D00 0C0B  0080 ARGN  EQU  0C0BH
0D00 DF63  0090      SCAL ZINLIN ; Get the input line
0D02 DF79  0100      SCAL ZRLIN ; Get the values from the line
0D04 3004  0110      JR   NC, CHK ; Check the values are 0k
0D06 DF6B  0120 ERR   SCAL ZERRM ; Error message
0D08 18F6  0130      JR   DEMO
0D0A 3A0B0C 0140 CHK   LD   A, (ARGN) ; Check one value only entered
0D0D FE01  0150      CP   1
0D0F 20F5  0160      JR   NZ, ERR
0D11 DF60  0170 PRT   SCAL ZARGS ; Get the value in HL
0D13 CD1A0D 0180      CALL DEC5HL ; Call print routine
0D16 DF6A  0190      SCAL ZCRLF ; Move to next line
0D18 18E6  0200      JR   DEMO ; Start again

0220 ; Routine to print HL in decimal
0230 ; Set HL to the value and call the routine
0240 ; AF, DE, and C registers are modified
0D1A 0030  0250 ROUT  EQU  30H
0D1A 111027 0260 DEC5HL LD   DE, 2710H ; 10,000 decimal
0D1D CD350D 0270      CALL SUBR
0D20 11E803 0280 DEC4HL LD   DE, 03E8H ; 1,000 decimal
0D23 CD350D 0290      CALL SUBR
0D26 116400 0300 DEC3HL LD   DE, 0064H ; 100 decimal
0D29 CD350D 0310      CALL SUBR
0D2C 110A00 0320 DEC2HL LD   DE, 000AH ; 10 decimal
0D2F CD350D 0330      CALL SUBR
0D32 110100 0340 DEC1HL LD   DE, 0001H ; 1 decimal
0D35 0E00  0350 SUBR   LD   C, 0 ; Count subtractions
0D37 0C  0360 SUB2    INC   C
0D38 B7  0370      OR    A
0D39 ED52 0380      SBC   HL, DE
0D3B 30FA 0390      JR   NC, SUB2
0D3D 0D  0400      DEC   C
0D3E 19  0410      ADD   HL, DE
0D3F 3E30 0420      LD   A, 30H ; Convert to ASCII
0D41 81  0430      ADD   A, C
0D42 F7  0440      RST   ROUT ; Output answer
0D43 C9  0450      RET

```

CHECKSUM ROUTINES

The following assembler programs submitted by David Wadham are to enable 16 bit checksums to be accumulated for any length program. This is a useful feature and we may well use it in future for published programs, allowing members to verify that programs have been typed in correctly. Also note the economy of the listing when written for Nas-Sys using Nas-sys internal routines.

ZEAP Z80 Assembler - Source Listing

```

0010 ; MEMORY BLOCK 16 BIT CHECKSUM : V2.1
0020 ; for NASBUG T2 or T4

0050 ; Execute: (BUGSUM) XXXX YYYY
0060 ; where: (BUGSUM) = location of program
0070 ;           XXXX = start of memory block,
0080 ;           YYYY = length of memory block
0090 ; (all values are hexadecimal).

0110 ; The result is:
0120 ; AAAA SSSS;
0130 ; where: AAAA = address of last byte,
0140 ;           SSSS=16 bit checksum (any
0150 ;           carries beyond this are discarded)
0160 ; Program then returns to the monitor.

0180 ; Program is fully relocatable (and
0190 ; may run in ROM).

```

```

0D00          0210      ORG  0D00H
0D00 0C0E      0220 ARG2  EQU  0C0EH
0D00 0C10      0230 ARG3  EQU  0C10H
0D00 0232      0240 TBCD3  EQU  0232H
0D00 0240      0250 CRLF   EQU  0240H
0D00 0359      0260 STRT0  EQU  0359H
0D00 2A0E0C    0270 BUGSUM LD   HL,(ARG2); Get the
0D03 ED4B100C  0280      LD   BC,(ARG3); arguments.
0D07 110000    0290      LD   DE, 0; Clear D & pushed sum store.
0D0A D5        0300      PUSH DE
0D0B 5E        0310 LOOP  LD   E,(HL); Get current value.
0D0C E3        0320      EX   (SP),HL
0D0D 19        0330      ADD  HL,DE; Add value to
0D0E E3        0340      EX   (SP),HL; pushed sum store.
0D0F 23        0350      INC  HL;   To next location.
0D10 0B        0360      DEC  BC;   Decrement counter.
0D11 78        0370      LD   A,B
0D12 B1        0380      OR   C;    set /reset flag
0D13 20F6      0390      JR   NZ LOOP
0D15 2B        0400      DEC  HL;   Back to last byte.
0D16 CD3202    0410      CALL TBCD3; Display last address.
0D19 E1        0420      POP  HL;   Get checksum.
0D1A CD3202    0430      CALL TBCD3; Display it.
0D1D CD4002    0440      CALL CRLF; New Line.
0D20 31330C    0450      LD   SP, 0C33H; Set Nasbus stack pointer.
0D23 C35903    0460      JP   STRT0; Back into Nasbus.
0470 ; END

```

ZEAP Z80 Assembler - Source Listing

```

0010 ; MEMORY BLOCK 16 BIT CHECKSUM : V1.1
0020 ; for Nas-Sys

```

```

0050 ; Execute: (CHKSUM) XXXX YYYY
0060 ; where: (CHKSUM) = location of program
0070 ;         XXXX = start of memory block,
0080 ;         YYYY = length of memory block
0090 ; (all values are hexadecimal).

```

```

0110 ; The result is:
0120 ; AAAA SSSS NAS-SYS 1;
0130 ; where: AAAA = address of last byte,
0140 ;         SSSS=16 bit checksum (any
0150 ;         carries beyond this are discarded)
0160 ;         and: NAS-SYS 1 indicates that the
0170 ; program has returned to the monitor.

```

```

0190 ; Program is fully relocatable (and
0200 ; may run in ROM).

```

```

0D00          0220      ORG  0D00H
0D00 0060      0230 ZARGS EQU  60H
0D00 006C      0240 ZTX1  EQU  6CH
0D00 005B      0250 ZMRET EQU  5BH
0D00 DF60      0260 CHKSUM SCAL ZARGS ; Get the arguments.
0D02 EB        0270      EX   DE,HL ; Mem pointer into HL
0D03 110000    0280      LD   DE, 0; Clear D & pushed sum store.
0D06 D5        0290      PUSH DE
0D07 5E        0300 LOOP  LD   E,(HL); Get current value.
0D08 E3        0310      EX   (SP),HL
0D09 19        0320      ADD  HL,DE; Add value to
0D0A E3        0330      EX   (SP),HL; pushed sum store.
0D0B 23        0340      INC  HL;   To next location.
0D0C 0B        0350      DEC  BC;   Decrement counter.
0D0D 78        0360      LD   A,B
0D0E B1        0370      OR   C;    Set /reset Z flag
0D0F 20F6      0380      JR   NZ LOOP
0D11 2B        0390      DEC  HL;   Back to last byte.
0D12 D1        0400      POP  DE;   Get checksum.
0D13 DF6C      0410      SCAL ZTX1 ; Display HL & DE
0D15 DF5B      0420      SCAL ZMRET; Return to NAS-SYS
          0430 ; END

```

Object code:

NASBUG version

```

>T 0D00 0D27
  0D00 2A 0E 0C ED 4B 10 0C 11
  0D08 00 00 D5 5E E3 19 E3 23
  0D10 0B 78 B1 20 F6 2B CD 32
  0D18 02 E1 CD 32 02 CD 40 02
  0D20 31 33 0C C3 59 03 00 00

```

Nas-Sys version

```

T 0D00 0D17
  0D00 DF 60 EB 11 00 00 D5 5E
  0D08 E3 19 E3 23 0B 78 B1 20
  0D10 F6 2B D1 DF 6C DF 5B 00

```

ZEAP O/P

ZEAP OUTPUT TO PRINTER/'TELETYPE'

by R. Beal

=====

Mr. J. Addey has written to the INMC saying:-

"We bought ZEAP some time ago hoping to use it on our NASCOM 1 which is exclusively used in conjunction with an ASR 'teletype'. End of story. No-one from NASCOM can offer any advice as to how ZEAP may be used except by using the original keyboard + VDU O/P."

Here are the answers for both ZEAP 1 and ZEAP 2.

ZEAP 1

=====

The address of the output routine is located at 0F22-0F23, so cold start ZEAP, then return to the monitor, then alter this address to your own routine and then warm start ZEAP. When you use ZEAP, only output from the U command (=list to UART) and assembler output with option 04 on (TTY option) will be output to the TTY. This is extremely useful as updates to text are carried out using the Nascom keyboard and VDU, command 04 is entered and then the A command, for hard copy listing.

If you are simply using a serial ASCII printer, there is no need to change the address at 0F22-0F23 because the default is to send the data to the UART in any case. Read pages 43 and 49 of the ZEAP 1 manual where it is explained very clearly. If you are using a parallel printer (or serial device) which require special print handling routines, then this pointer must be changed to direct data to the output handling routine.

ZEAP 2

=====

The comments above for ZEAP 1 apply also to ZEAP 2, except that the address of the output routine is at 0F05-0F06. See appendix E, and appendix G, as well as the description of the U command, all clearly set out in the ZEAP 2 manual.

INPUT

=====

You may wish to use an external keyboard for input. ZEAP does nothing special when it gets an input character, so you can activate a teletype keyboard as described in the Nas-Sys manual under the X command. (Enter X0.) This also works with the T4 monitor, and applies to ZEAP 1 and ZEAP 2. If you are still back in the dark ages with T2 then bad luck, you can't do it without writing a special program. Using the X command will not in this case affect output, as ZEAP itself takes control of where the output is to go to.

All Change

CONVERSION TO NAS-SYS

BY RICHARD BEAL

=====

Many readers have requested a table showing equivalent codes for old monitors and for Nas-Sys. Many new INMC members have only ever used Nas-Sys, and have come across old programs they wish to convert. Derek Brough has made a tremendous effort in converting the T4 library to Nas-Sys, so you will find it easier to get some of these programs from the new Nas-Sys machine code program library.

To keep the table simple we have compared T4 to Nas-Sys. The older T2 and B-Bug monitors were subsets of T4, so the conversion works in the same way for them. It is not possible to show every detail because there are so many small differences, in particular the method of input and output.

Name	Function performed	T4	Nas-Sys
====	=====	==	=====
RIN	Get input character	CD 3E 00	CF
ROUT	Output a character	CD 4A 0C	F7
		or F7	
		or CD 3B 01	
PRS	Print a string	EF ...00	EF...00
RCAL	Relative call	D7 ..	D7 ..
SCAL	Subroutine call	NONE	DF ..
RDEL	Delay	FF	FF
		or CD 35 00	
BRKPT	Breakpoint	E7	E7
START	Reset computer	C7	C7
MRET	Return to montior	C3 86 02	DF 5B
		or CF	
TDEL	Long delay	NONE	DF 5D
FFLP	Flip bits in port 0	CA 4A 00	DF 5E
MFLP	Flip tape drive LED	CD 51 00	DF 5F
ARGS	Load arguments	CD 97 06	DF 60
IN	Scan for input	CD 4D 0C	DF 62
	character	or CD 69 00	
INLIN	Obtain input line	NONE	DF 63
NUM	Convert from ASCII to	CD 5A 02	DF 64
	binary		
TBCD3	Output HL in ASCII	CD 32 02	DF 66
TBCD2	Output A in ASCII	CD 2B 02	DF 67
B2HEX	Output A in ASCII	CD 44 02	DF 68
SPACE	Output space	CD 3C 02	DF 69
CRLF	Output carriage return	CD 40 02	DF 6A
ERRM	Output error message	NONE	DF 6B
TX1	Output HL,DE in ASCII	CD 5B 04	DF 6C
SOUT	String of characters	CD CC 06	DF 6D
	to serial output		
SRLX	Character to serial	CD 5D 00	DF 6F
	output	or CD 5E 00	
RLIN	Get arguments from	NONE	DF 79
	input line		
B1HEX	Output half of A in	CD 4D 02	DF 7A
	ASCII		
BLINK	Blink cursor, get input	NONE	DF 7B
CPOS	Find start of line	NONE	DF 7C

DOTS

DOTS

====

The advent of the Nascom 2, with its graphics ability, the Econographics add-on kit giving similar facilities to the Nascom 1, and the much awaited and (as far as we know) imminent Programmable Character Generator, has made investigation into how the characters are formed on your monitor screen appropriate. The fact that the Nascom 2 and the Econographics both use a 2716 compatible ROM as the character generator, coupled with the new breed of cheap 2716 EPROM programmers means that character sets of your own are very easily prepared.

We don't intend to discuss in detail exactly how the characters are sent to your screen, but to concentrate on the organisation of the character generator ROM (or in the case of the PCG, RAM). First, the ROM is a 2K device, hence has 11 address lines. Essentially what happens is that the hardware scans the video RAM which in turn produces 48 sequential addresses representing the 48 characters to be displayed on that row. 7 address lines are used for this, known as the Character Selects (CS0 - 6). Each address is applied to the character ROM causing a data byte to be output from the character ROM which is latched into a 'parallel in/serial out' shift register, and a clock shifts the data byte out bit by bit producing a pattern of 1s and 0s which represent white and black dots (respectively) on the screen. A row of characters is actually composed of 16 TV lines (14 on a Nascom 2, but it only skips that last two), and the remaining 4 address lines, known as the Row Selects (RS0 - 3), tell the character ROM which TV line it is on. Enough of this, it takes a fair amount of study of the circuits to understand it properly, and it's not really important to the matter of programming a character set.

The important things are that a character is represented by 16 rows of 8 dots each. Each dot is one bit of a byte, so that the character can be represented by a total of 16 bytes. As there are 128 characters in a character ROM, each of 16 bytes, then the ROM must be a 2K device. It's not difficult to program a 2716 to make a special graphics EPROM, so all that remains is to find out how to program a character.

So to designing a special character. Take a piece of 0.1" or 2mm graph paper, and mark off a rectangle 8 squares wide by 16 squares deep. Outline your character within the rectangle in pencil, then shade in the squares that your outline passes through. Stand back and examine the character from a distance to see if it looks right. If so then code the 16 bytes used, remember that a binary 0 is black, a binary 1 is white. (Don't forget your monitor displays white on black.) See the attached diagram of a letter 'A'. Note that in this instance the right hand column is left blank to allow for a space between characters, if characters were to be joined together, then no blank is required. The code for an 'A' is:

```
38 44 82 82 FE 82 82 82
```

```
82 00 00 00 00 00 00 00
```

Notice the character is slightly broader than it will appear on the screen, due to the non-symmetrical aspect ratio of the monitor screen. Any character may be built up in a similar fashion. Hence the evil looking 'Space Invader' on the right is built up of three characters as follows:

```
00 00 00 03 07 0E 0F 1F
```

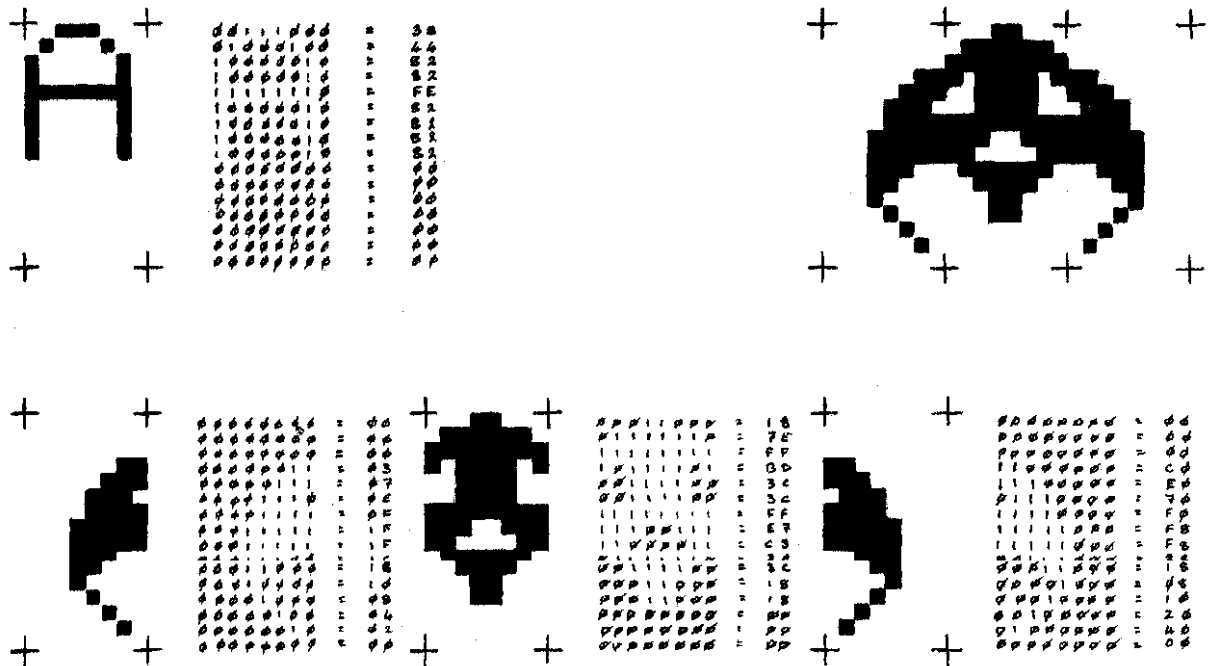
```
1F 1C 18 10 08 04 02 00
```



```
18 7E FF BD 3C 3C FF E7
C3 7E BC 18 18 0F 00 00
```

```
00 00 00 C0 E0 70 F0 F8
F8 38 18 08 10 20 40 00
```

Each is a discrete character, which when displayed together on the screen form a complete 'Invader'.



So having discovered how a character is built up, how is it organised in the ROM? The first thing to realise that a ROM is an independant device, the first byte in the ROM is address 000H, regardless of where the ROM may sit in a memory map. The first byte of the first character starts at address 000H and the character takes up 16 (10H) bytes. The second character therefore starts at address 010H, the third at 020H, etc., the last character starts at address 7F0H. Simple really. Now another sneaky trick. Consider the alpha-numeric character generator, notice that the character numbers are 00, 01, 02, etc. Well relate that to the addresses in the ROM. Character 00 starts at address 000H, character 01 starts at address 010H, 02 at 020H etc. See the pattern? The character number is the same as the address with the least significant '0' missing. Not difficult. Now the graphics characters start at character number 80. So all you have to do to find your new character in your new graphics ROM is to start counting at 80 instead of 00.

The characters must first be programmed into RAM, and as the graphics start at 80, a convenient place to start is an X800H memory boundary, that way you can keep track of the character addresses without mental arithmetic to add 80 to each character number. You don't have to program all 128 characters at once, but leave all unprogrammed bytes at FFH, that way you can 'over program' later. With all the characters in RAM, make a tape of them, and then blow the EPROM. At long last it can be said that Nascom owners are going dotty !!!!

HEX~?

Reserved Words, and Graphics =====

The Hex codes for the reserved words and graphics are tabulated below. It is rather unfortunate that the results LISTed for the last 48 graphics, where there are no reserved words, are in many cases beyond the scope of the printer. Where an unprintable character is encountered, a '.' is substituted. We suggest that users complete this table for themselves, typing in a Basic line using the combination of keys given, then LISTing the results. This list of graphics, reserved words and keys required should be used in conjunction with the graphics list on page 10 of INMC 7.

The key to the keys used is:

Gr = GRAPH, Ct = CTRL, Sh = SHIFT

The keys should be depressed in the order listed.

HEX	CHR\$	Keys	Reserved word
80	128	Gr/Ct/Sh/@	= END
81	129	Gr/Ct/A	= FOR
82	130	Gr/Ct/B	= NEXT
83	131	Gr/Ct/C	= DATA
84	132	Gr/Ct/D	= INPUT
85	133	Gr/Ct/E	= DIM
86	134	Gr/Ct/F	= READ
87	135	Gr/Ct/G	= LET
88	136	Gr/Ct/H	= GOTO
89	137	Gr/Ct/I	= RUN
8A	138	Gr/Ct/J	= IF
8B	139	Gr/Ct/K	= RESTORE
8C	140	Gr/Ct/L	= GOSUB
8D	141	Gr/Ct/M	= RETURN
8E	142	Gr/Ct/N	= REM
8F	143	Gr/Ct/O	= STOP
90	144	Gr/Ct/P	= OUT
91	145	Gr/Ct/Q	= ON
92	146	Gr/Ct/R	= NULL
93	147	Gr/Ct/S	= WAIT
94	148	Gr/Ct/T	= DEF
95	149	Gr/Ct/U	= POKE
96	150	Gr/Ct/V	= DOKE
97	151	Gr/Ct/W	= SCREEN
98	152	Gr/Ct/X	= LINES
99	153	Gr/Ct/Y	= CLS
9A	154	Gr/Ct/Z	= WIDTH
9B	155	Gr/Ct/[= MONITOR
9C	156	Gr/Ct/Sh/[= SET
9D	157	Gr/Ct/]	= RESET
9E	158	Gr/Ct/Sh/0	= PRINT
9F	159	Gr/Ct/Sh/]	= CONT
A0	160	Gr/space	= LIST
A1	161	Gr/Sh/1	= CLEAR
A2	162	Gr/Sh/2	= CLOAD
A3	163	Gr/Sh/3	= CSAVE
A4	164	Gr/Sh/4	= NEW
A5	165	Gr/Sh/5	= TAB(
A6	166	Gr/Sh/6	= TO
A7	167	Gr/Sh/7	= FN
A8	168	Gr/Sh/8	= SPC(
A9	169	Gr/Sh/9	= THEN
AA	170	Gr/Sh/:	= NOT
AB	171	Gr/Sh/;	= STOP
AC	172	Gr/Sh/,	= +
AD	173	Gr/-	= -
AE	174	Gr/.	= *
AF	175	Gr//	= /
B0	176	Gr/0	= !
B1	177	Gr/1	= AND
B2	178	Gr/2	= OR
B3	179	Gr/3	= >
B4	180	Gr/4	= =

B5	181	Gr/5	= <
B6	182	Gr/6	= SGN
B7	183	Gr/7	= INT
B8	184	Gr/8	= ABS
B9	185	Gr/9	= USR
BA	186	Gr/:	= FRE
BB	187	Gr/;	= INP
BC	188	Gr/Sh/,	= POS
BD	189	Gr/Sh/-	= SQR
BE	190	Gr/Sh/.	= RND
BF	191	Gr/Sh//	= LOG
C0	192	Gr/Sh/@	= EXP
C1	193	Gr/A	= COS
C2	194	Gr/B	= SIN
C3	195	Gr/C	= TAN
C4	196	Gr/D	= ATN
C5	197	Gr/E	= PEEK
C6	198	Gr/F	= DEEK
C7	199	Gr/G	= POINT
C8	200	Gr/H	= LEN
C9	201	Gr/I	= STR\$
CA	202	Gr/J	= VAL
CB	203	Gr/K	= ASC
CC	204	Gr/L	= CHR\$
CD	205	Gr/M	= LEFT\$
CE	206	Gr/N	= RIGHT\$
CF	207	Gr/O	= MID\$
End of list of reserved words			
D0	208	Gr/P	= r
D1	209	Gr/Q	= hy
D2	210	Gr/R	= g
D3	211	Gr/S	= v
D4	212	Gr/T	= lp
D5	213	Gr/U	= j
D6	214	Gr/V	= .
D7	215	Gr/W	= k(
D8	216	Gr/X	= o,
D9	217	Gr/Y	= 1
DA	218	Gr/Z	= .
DB	219	Gr/[= j-
DC	220	Gr/Sh/[= j.
DD	221	Gr/]	= j
DE	222	Gr/Sh/0	= .
DF	223	Gr/Sh/]	= jF
E0	224	Gr/Ct/Sh/space	= h.
E1	225	Gr/Sh/A	= jK
E2	226	Gr/Sh/B	= jr
E3	227	Gr/Sh/C	= jp
E4	228	Gr/Sh/D	= hM
E5	229	Gr/Sh/E	= t
E6	230	Gr/Sh/F	= a
E7	231	Gr/Sh/G	= j
E8	232	Gr/Sh/H	= 1
E9	233	Gr/Sh/I	= hs
EA	234	Gr/Sh/J	= t.
EB	235	Gr/Sh/K	= q
EC	236	Gr/Sh/L	= *
ED	237	Gr/Sh/M	= u
EE	238	Gr/Sh/N	= G
EF	239	Gr/Sh/O	= .
F0	240	Gr/Sh/P	= f
F1	241	Gr/Sh/Q	= .
F2	242	Gr/Sh/R	= -
F3	243	Gr/Sh/S	= .
F4	244	Gr/Sh/T	= .
F5	245	Gr/Sh/U	= .
F6	246	Gr/Sh/V	= %
F7	247	Gr/Sh/W	= .
F8	248	Gr/Sh/X	= .
F9	249	Gr/Sh/Y	= .T.W.#
FA	250	Gr/Sh/Z	= k
FB	251	Gr/Ct/;	= .
FC	252	Gr/Ct/Sh/,	= h
FD	253	Gr/Ct/Sh/-	= .
FE	254	Gr/Ct/Sh/.	= f
FF	255	Gr/Ct/Sh//	= J

Z80 made simple

THE KIDDIES' GUIDE TO Z80 ASSEMBLER PROGRAMMING.

D. R. Hunt

=====
Part: The First (and if doesn't go down too well, Part: The Last).

Funny numbers, counting with sixteen fingers, and all that.
=====

We have had many appeals for a beginner's guide to Z80 assembler programming, and as no-one else volunteered, I thought I might have a go. After all, my qualifications for this task are impressive.

- 1) Three years ago I knew nothing about it.
- 2) It is arguable if I have learned much in the meantime.
- 3) I don't mind making myself look an idiot in print (in the eyes of the enlightened).

All this means is that I'm not too clever at it to baffle the reader, and that I'm new enough at it remember the difficulties I had at first.

So here goes: The first thing to learn is HEXadecimal, the numbering system used when writing machine code. HEX numbers are usually indicated by numbers either being prefixed '#', or suffixed 'H'. Now to use HEX effectively, really means that you should grow three more fingers on each hand, as this is a little difficult for most normal people, an explanation will have to suffice.

Binary and HEXadecimal

=====
Most of us count in units of 1, 10, 100, etc. For historical reasons we need not discuss, this counting in 10s business (known as 'base 10' counting or Decimal) is so much second nature that counting in any other form may well seem ludicrous. However, there are other things on this earth which do use different systems, and computers figure largely among them. Right at the heart of it, the computer uses the Binary system, and all it knows is that a number represented as 'no volts' will be interpreted as a '0', whilst a number represented by 'some volts' will be interpreted as '1'. From this it can be seen that the computer counts in twos (known as 'base 2' counting or Binary). In the same way that we count in 1s, 10s, 100s, etc, the computer counts in 1s, 10s, 100s etc. Unfortunately, although the numbers look the same, they are in fact different. As each unit is the base number raised to next power (mathematically speaking) they actually mean:

We think	Computers think
10 to the power 0 = 1	2 to the power 0 = 1 (= 1 Decimal)
10 to the power 1 = 10	2 to the power 1 = 10 (= 2 Decimal)
10 to the power 2 = 100	2 to the power 2 = 100 (= 4 Decimal)
10 to the power 3 = 1000	2 to the power 3 = 1000 (= 8 Decimal)
	etc.

So that the number fifteen (Decimal) is expressed as:

1000	(8 in Decimal)
100	(4 in Decimal)
10	(2 in Decimal)
1	(1 in Decimal)
====	
1111	(= 15 in Decimal)

Don't forget when adding Binary numbers that $1 + 1 = 0$, carry 1, and not 2, as it would in Decimal.

Now what has all this got to do with HEX ? Well, that answer is managability. If we take a Decimal number, say ten, then we write '10'. If want to express the same amount in Binary, then it becomes the cumbersome '1010', worse, if we write one thousand three hundred and twelve, we write 1312 in Decimal, in Binary this becomes the forbidding figure '1010010000'. You can see that even with fairly modest Decimal figures, the Binary equivalents are getting unmanagable. The Z80 processor has an addressing capacity of 65535 bytes, which in Binary becomes the incredible '1111111111111111', which is difficult to read yet alone assimilate the actual number. To answer the question at the beginning of the paragraph, we've got to find some more convenient way of expressing the Binary digits, or the whole thing becomes impossible from the start. That's where HEX comes in, but to understand that, we've got to go through the knotty business of Binary to HEX and Decimal to HEX conversions.

Having established that other 'bases' may be used apart from 'base 10', 16 could be used as a base, in fact that is what HEXadecimal means, counting in sixteens. A second thing to realise is that when counting to 'base 2' we only need two characters. 0 and 1; counting in 'base 10' requires ten characters, 0 1 2 3 4 5 6 7 8 9; it therefore follows if we are to count in sixteens, sixteen characters are required. These are:

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

The third thing to notice is that 16 is the fourth power of 2, and we can make use of that to make simple conversions. We do it by mentally converting a Binary number to Decimal then converting the Decimal to HEX. A messy process, and one that is soon forgotten when you get familiar with HEX. So think of a Binary number, say 11101101110, now split it into groups of four, starting with the right hand end, thus, 111 0110 1110, add a '0' to the left hand end to complete the last group of four, and we get 0111 0110 1110. Now notice that the maximum Decimal number in any group is 15, and with a little practice, the conversion becomes dead easy, like this:

0111	0110	1110
7	6	14

Now cheat and look at the first (right hand) column in the next paragraph. Look up the Decimal number, and write down the equivalents. So:

0111	0110	1110
7	6	14
7	6	E

our Binary number 11101101110 becomes a nice manageable 76EH. As there are only six letters to learn to replace the Decimal numbers 10 - 15, a little practice will soon dispense with the intermediate Decimal stage, and you can do Binary to HEX conversions direct. (To tell the truth I still have difficulty doing it backwards, and end up jotting down a Binary table from 10 - 16.)

Decimal to HEX is easiest achieved using a conversions table written in powers of 16, expressed in 'base 10'. The table looks like this:

HEXADECIMAL COLUMNS			
4	3	2	1
0 0	0 0	0 0	0 0
1 4,096	1 256	1 16	1 1
2 8,192	2 512	2 32	2 2
3 12,288	3 768	3 48	3 3
4 16,384	4 1,024	4 64	4 4
5 20,480	5 1,280	5 80	5 5
6 24,576	6 1,536	6 96	6 6
7 28,672	7 1,792	7 112	7 7
8 32,768	8 2,048	8 128	8 8
9 35,864	9 2,304	9 144	9 9
A 40,960	A 2,560	A 160	A 10
B 45,056	B 2,816	B 176	B 11
C 49,152	C 3,072	C 192	C 12
D 53,248	D 3,328	D 208	D 13
E 57,344	E 3,584	E 224	E 14
F 61,440	F 3,840	F 240	F 15

Right, now what to do with it ? Think of a number, an easy one, say, 49. Look in column 4, is there a number less than 49 ? Yes that number is 0, so the first digit is 0. Try in column 3, 0 again. Try column 2, Ah hah, 48 is less than 49, so the HEX digit is 3. Now subtract 48 from 49 and look in column 1 for the HEX equivalent to the answer. Yes, you got it, 1 !!! So the HEX equivalent of 49 is 0031. We don't write the Decimal number 0049, so it follows that the HEX number is 31. Also as we can't assume it's to 'base 10' (because it isn't), we'd better tell the customers that its HEX, so:

Decimal 49 = 31H

Now that is a lot easier than writing 49 in Binary (which would be 110001), and even vaguely understandable to the uninitiate. Now try 19,514, and see if you understand how I got 4C3AH out of that. Go on try a few numbers of your own.

The whole object of this exercise has been to point out that different numbering systems exist and that HEX numbers are just as real and meaningful as counting from 1 to 10 (in Decimal of course). How computers make the conversion from HEX to Binary for internal use will be revealed in the next chapter. Meantime, try getting to grips with HEX, and relating it Decimal and Binary counting systems.

S100

READERS' REPLIES

S100 EXPANSION

=====

Below are extracts from several letters that we received in response to our article in INMC 6 on Comp S100 expansion.

S100-1

=====

Dear Sir,

Firstly thanks for a newsy and (usually) accurate comic which is always worth reading.

With regard to the article in Issue 6, caution should be exercised when buying the COMP S100 expansion kit. We looked into using this product to drive a (U.S.) Heuristics Speechlab Board and found that instead of being a true S100 decoder, ONLY these signals required for use with the range of boards COMP sell are actually decoded.

Hope the comments are of some use.

Yours faithfully,

J. Addey.
North Humberside.

S100-2

=====

Dear Sir,

In INMC News no. 6, I was pleased to see that someone else was experiencing problems with the Comp S100 expansion. Mine has never worked, I started off with symptoms similar to those described by Mr. Curtis, but all this was cured by rearranging the wiring. My Nascom works perfectly, on external, with the buffer S100 board attached, I can write into the 8K expansion, but when I try to execute any program in the expansion, the program is corrupted. Any program in the on board Nascom memory runs without problems. Anyway I'm now going to look at the R/W lines. Comp assured me that they've had no problems with this kit at all. At the moment I am working on connecting up an MM57109 Number Cruncher. I will let you know how I get on.

Yours faithfully,

A.C. Cox
Abadan, Iran.

S100-3
=====

Dear INMC,

After receiving INMC News 6, I am prompted by your S100 expansion article to write to you about my own efforts in that direction.

When I started expanding my Nascom 1, my first step was to buy and construct the Nascom buffer board. After this I decided to go my own way and use S100 rather than Nasbus, primarily because of interest in S100, and also because at that time (mid 79) there was no way I could get hold of Nascom 8k memory.

I was lucky enough to be enquiring one day about S100 backplanes in Comp Shop of Barnet and they kindly sold me a blank PCB as used in their own S100 expansion kit. On closer examination of this PCB I noticed that they had, in their design, provided a set of holes at one end of the board to take the standard Nasbus 77 way connector. This PCB was then an ideal basis for Nasbus to S100 conversion.

The PCB was cut accordingly just leaving the three S100 sockets and the Nasbus connector. All address and data bus lines were still intact between the sockets, with only the control bus signals requiring attention. A simple circuit using three IC's was employed to generate the S100 MREAD and MWRITE signals, the MEMEXT signal for Nascom 1, and the DBDR signal for the Nascom buffer board.

The standard power supply was upgraded as suggested by Mr. Curtis in INMC 6, but with the addition of an extra smoothing capacitor and current limiting resistor to provide a true 8V supply for the S100 on board regulators. (The output at my bridge rectifier is approx. 10V and would, without a limiting resistor, result in the regulators getting unnecessarily hot trying to dissipate a further 2V).

No problems were encountered running this set up, with 8k of static memory; and with the recent addition of the excellent Nascom Rom Basic chip (fitted to a surplus S100 board using a single TTL IC to decode the chip enable). The system performs beautifully, with no hardware crashes and a very cool running PSU.

All I now need is a copy of Nas-Sys 1 and I have effectively upgraded to Nascom 2 (the on screen editing facilities of Nas-Sys are an absolute must when editing 8k Basic programs). Perhaps the software experts amongst you will devise an addition to 8k basic, held in RAM to provide these facilities for us poor T4 owners.

I would of course only be too willing to give more information about my conversion to anybody contemplating a similar project, although I suspect that nowadays with the plentiful supply of Nascom memory, users would sensibly opt for a 100% Nascom system.

Yours truly,

Malcolm Bay,
Bedfordshire.

PIO pages

USING THE PIO OR 'SPACE INVADERS MEET THE NASCOM PIO'

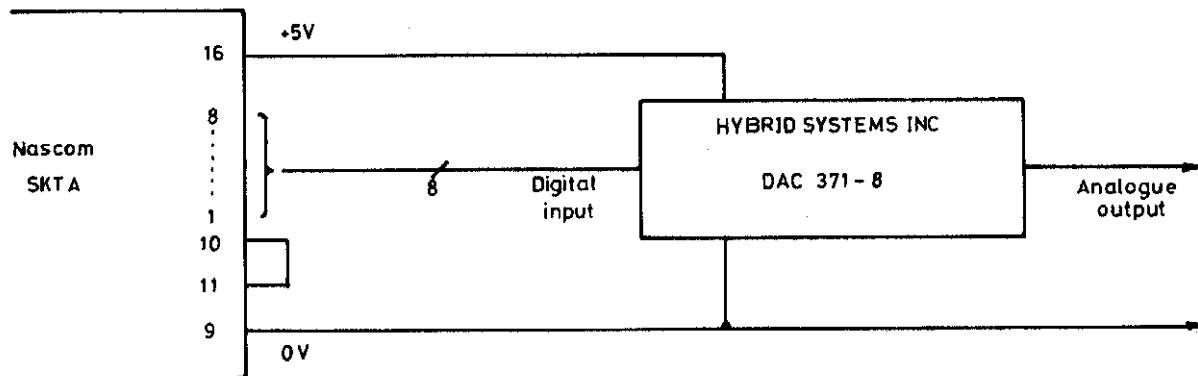
by Martin Dyer

=====

I must confess that for a long time I regarded the PIO as 'just the third 40 pin chip on the Nascom board', but one day I would get round to using it for some application.

Then INMC News, issue 2 arrived, containing a very useful article on the PIO. This gave practical tips & facts not covered in the PIO Handbook. The article concluded with a request, nay more of a plea, for members to describe their applications of the PIO.

Armed with all this information I set about making a Digital to Analogue Interface using a Digital - Analogue Convertor (DAC) I bought at the Longleat (Amateur Radio) Mobile Rally.

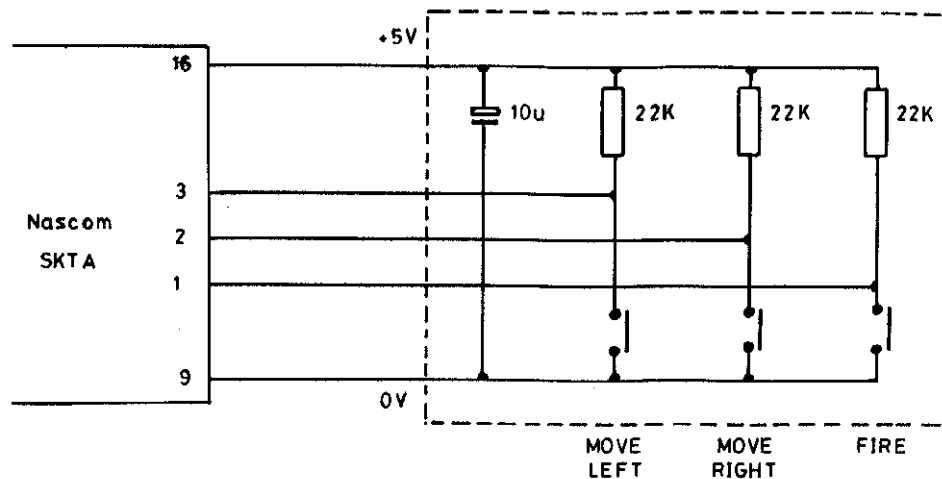


No program was written, because using the T4 Output command port A could be set up for mode 0 and no Interrupt, and data written to the DAC. The long term aim is to attempt to write a Music Program, directly synthesizing music waveforms or using the analogue signal to control a voltage controlled oscillator.

For reference, the address of the supplier of the writer's DAC is LB Electronics., 43 Westacott, Hayes, Middlesex. The writer has no information regarding current price and availability.

Some time after, Graham Clarke's Space Invasion game was obtained from the Software Library. I will certainly recommend this game to every one, it is most entertaining and very addictive. I won't describe the game, so play it to find out! The software uses the A, X and Z keys to input information to the Nascom from the operator. The KBD subroutine detects the action of a key-depression but provides no information if the key is kept depressed. In the Space Invasion game the effect was making the software slow to respond to operator commands. The Hardware and Software described below overcomes many of the problems.

Three push button switches were mounted in a 4" x 4" x 1.5" plastic box and was connected to the Nascom by a multicore cable. This made a hand-held control box, which was easier to use than finding the correct keys on the keyboard.



The program to drive the game replaces the code using the keyboard exactly, apart from the delay routine which follows the end of the original coding.

The program executes from D00H, starting with P10 initialisation and then jumping to the original program.

LOCATIONS	ROUTINE	DESCRIPTION
CA9-CC0	MLOOP	Scans the switches, will allow firing of a missile whilst moving left or right.
CC2-CD2	FIRW	Fires a missile as per original program.
CD5-CE4	T12	Prevents more than one missile being fired for each key depression.
CE6-CFC)	LEW	Move left, as per original program.
D0D-D20)		
D00-D0A	START	Initialises P10; Starts program.
D22-D4A	REW	Move right, as per original program.
FD2-FDE	SLOW	Controls speed of movement, by value at FD4

```

0CA9 DB 04 CB 47 28 13 CB 4F 28 6F CB 57 28 2F CD 76
0CB9 0D CD F0 0E CD 7A 0F 18 E7 DD 7E 01 FE 19 20 0C
0CC9 DD 77 C1 DD 36 01 20 3E 04 32 52 0C DB 04 CB 57
0CD9 20 CE CD 76 0D CD F0 0E CD 7A 0F 18 EF CD D2 0F
0CE9 3A 50 0C FE 00 28 B9 DD 7E 00 DD 77 FF DD 7E 01
0CF9 DD 77 00 18 0F
0D00 3E FF D3 06 3E 07 D3 06 D3 06 C3 24 0F DD 7E 02
0D10 DD 77 01 DD 36 02 20 DD 2B 3A 50 0C 3D 32 50 0C
0D20 18 87 CD D2 0F 3A 50 0C FE 2C 28 F4 DD 7E 02 DD
0D30 77 03 DD 7E 01 DD 77 02 DD 7E 00 DD 77 01 DD 36
0D40 00 20 DD 23 3A 50 0C 3C 18 D3 00 00 3A C6 0D FE
0D50 09 F0 F1 C3 97 0F
0FD2 C5 06 09 CD 35 00 10 FB C1 CD 76 0D C9

```

In principle, the ideas given above can be adapted to suit many games. The unit makes Space Invasion much easier to play and saves wear and tear on the keyboard when hitting the keys in the excitement of the game.

Reviews

NASCOM I/O BOARD REVIEW

D. R. Hunt.

=====

The Nascom I/O board was just too late for the last issue, so this review was held over. The I/O board accomodates 3 PIOs, 1 UART and 1 CTC. The PIOs are the MK3881 type as fitted to the Nascom main board, the UART is likewise similar, but has a crystal controlled BAUD rate generator giving speeds from 110 to 9600 BAUD, transmit and receive can only take place at the same speed, unlike the separate transmit/receive arrangement on the Nascom 2. The final chip is the MK3882 "Counter Timer Circuit", an ingeniously simple chip, more details later. The board is supplied with sockets for all the chips, but only the necessary chips to put the board into action come with the kit. The major I/O devices, their attendant decoders, plugs, leads (and in the case of the UART, the BAUD rate generator, crystal and sundry Rs and Cs) are supplied as 'add-on' packs, to be purchased as required. The kit is supplied with documentation and the PIO, UART and CTC technical manuals. The three PIOs talk to the outside world through three 26 way connectors (a la Nascom 2) on the front edge of the pcb. The UART has RS232 and 20mA input/output via a 16 pin dil socket. The CTC input/output and the BAUD rate select are also via 16 pin dil sockets. If I remember correctly, the original published spec. of the I/O board included a modem as well. This seems to have fallen by the wayside.

The issue 2 boards supplied have a track error which we understand was not spotted until a quantity of pcs had been made, and an errata is included giving the small amount of 'board surgery' required. This 'surgery' is only required when the UART is brought into play. The errata also covers other small documentation errors, but has not spotted that PIO 1 and PIO 3 are reversed throughout. The documentation rates the 'INMC FOUR STAR AWARD' for incomprehensibility, understanding was not helped by the fact that a number of kits escaped with the last three pages of the documentation missing. The circuit diagram however was clear and detailed, and once one realises that the 8131 decoders used are only glorified XOR gates (or comparators), then, what the documentation lacks, the circuit diagram makes up for.

Construction was straight forward and simple, the pcb being of the usual high standard, however, the style of pcb layout was entirely different to the usual style of Nascom pcbs, which made following things through a little more difficult than usual. Nothing wrong here, just different. We understand the pcb was laid out by hand rather than Nascoms' usual practice of using a drafting computer. The manual gave details of recommended I/O mapping, although perhaps this was uninspired, and not helped by the missing appendix which we presume gave an expanded I/O map. Connecting the decode links was carried out from the drawings in the manual, no explanation was offered as to why things should be linked in the fashion shown. Watch out for the fact that the Nascom internal port decode (IOEXT line) can be set to either ports 0 - 4 or 0 - 8. When used with Nascom 1 the onboard PIO has to be removed, meaning that the lowest decode address may be 4, whereas the PIO in the Nascom 2 can remain in situ, if so, then the lowest decode address should be set to 2. After a visual check, the board was tried.

Then a hidden bug appeared. We've mentioned before that the IEI and IEO lines get connected together on a Nascom 2 if you use the little mini-bus that was supplied with the 16K versions or a 'Vero bus', and this means that the 'daisy chain' interrupt priority function is 'all screwed up'. Not a serious problem, but requires thought to

unscramble in a satisfactory manner. The best way to get IEI and IE0 straight is to, first, read and understand the bit in the PIO manual about the 'daisy chain'. There is a good applications note titled "The Z80 Family Program Interrupt Structure" available from Zilog, which explains everything. Then cut the tracks between 19 - 19, 20 - 20 between each card on the bus. Decide which end of the bus is going to have the highest priority and connect a 10K resistor to bus line 19 at that end. Connect line 20 of the highest to 19 of the next highest, connect 20 of this to 19 of the next, and so on. Whilst you are at it, repeat the process for BAI/BA0, as these are a form of 'daisy chain' also. The cards may then be inserted from the end which has the highest priority, the 'chain' linking through the RAM cards etc. Do not leave gaps in inserting the cards, as the 'chain' would then be broken. A simple test program was written, and each device on the card tested and found to function correctly.

The I/O card allows considerable addressing flexibility, with the 'daisy chain' fully implemented even when devices are left off the board, so that two I/O cards could be used, each only containing two PIOs for instance. On the card itself, the priorities for the 'daisy chain' are fixed, with the CTC being the highest, followed by PIOs 1 - 3. The UART is not implemented within the priority structure. Cascading of two I/O boards is possible, but more may put timing limitations on the priority structure. The Zilog application note explains what could happen.

In general, the I/O card is another carefully thought out product, let down (as usual) by poor documentation. Although somewhat of a specialist nature, it is a must for those who want to speak to the outside world. Of course, interrupts may be a problem, as NAS-SYS 1 is not interruptable, (and there are limitations on the NASBUGs as well). There is a new monitor on the stocks, which overcomes this problem, and has other enhancements. But for most people, interrupts can be handled with a little careful programming. If you are clever enough to work out how to use the interrupts in the first place, avoiding trouble with the monitor should only be a minor inconvenience.

Lastly, a brief description of the CTC, as this device will be unfamiliar to most users. This is a simple but very versatile chip consisting of four virtually identical channels. At the heart of each channel is a 256 (8 bit) countdown counter, with a parallel register which contains a preprogrammed constant. At any time the processor can access the countdown counter to determine the count, or preload the constant register. Two inputs are provided for the countdown counter, and may be selected by the appropriate control word. One from the system clock via a divide 16 or divide 256 prescaler, or from an external input. Control words select the sense of the trigger counting edge, or in the timer mode, whether the count will start immediately or after the input of a preselectable input edge. The scheme is as follows:

Preload the constant, allow the countdown to proceed to zero, whereupon the CTC outputs a pulse, and optionally causes an interrupt (telling the CPU which of the 4 counters it was). At zero count, the constant is reloaded into the counter and it all starts again. All four counters work independantly, and may be timers or counters, they may be cascaded allowing very large division ratios to be achieved if required. It really is a clever little beast with dozens of uses.

Nice one Nascom!

Review of Port Probe and Key Pads from Bits & P.c's. by R. O'Farrell
=====

The Port Probe is a small single sided P.C. board 4" x 3", supplied drilled and tinned, which holds 4 i.c.s, an 8 position DIL switch, 9 LEDs and four push button switches. The i.c.s are supplied with sockets, and are configured to make up a clock, whose frequency can be altered by a preset within limits (1 cycle every 0.5 sec to 1 cycle every 3 secs approx). This clock can be started and stopped by two of the push buttons, and is fed to a timing chain. Selected outputs from this chain can be fed to a PIO port (16 pin dil connector supplied) and the workings of this complex and mysterious item explored in full. The other two switches on the probe are to reset the counter and to handshake with the port. It is well documented with two sample programs supplied to get you started. At the price of 11.20 I would regard it as a must for any N1/N2 owner.

The control and HEX keyboards consist of a bank of keys on each side of an island of i.c.s. Depending on whether you order one or the other or both, so you get the centre island, and one or the other bank of keys. The keys are supplied with proper keytops, very like the Nascom keyboard. If you have changed an N1 to Nas Sys, then the control keypad allows all the cursor movement characters to be accessed easily - much easier than remembering how to get say 'control T'. Again, well documented and socketed. The extension board takes the umbilical from the Nascom board, and in turn plugs into the Nascom Main board. For a Nascom 1 running under Nas Sys, the control pad is essential, the HEX pad not so important. One slight problem on mine, a tendency for certain characters to repeat. This I cured by swapping certain of the i.c.s about. I feel it is due to the flipflop connected gates on the Nascom Keyboard re-flipping because of echoes on the line, and perhaps if it occurs often a veroboard mounted line driver near the N1 keyboard might cure it. All in all, two most satisfactory purchases. I strongly recommend the port probe to all N1/N2 owners, and the control section of the extended keyboard to N1's running under Nas Sys. The control keypad costs 23.10, the HEX keypad 30.10, and both together 37.25.

3M PERSONAL COMPUTING CASSETTES.
=====

by D. Brough

3M have recently introduced C10 & C30 cassetts designed for personal computer users. Several people have had very good results with the C30's up to at least 1200 baud both on NASCOMs and SORCERORs. One of the test tapes did have some creases which caused dropouts but this was replaced without question. The tapes at present have transparent leaders but this may be changed soon. Prices are:- (Post Free).

C10 - 64p incl VAT

C30 - 67p incl VAT

Contact 3M at 3M (UK) Ltd., P.O. Box 1, Bracknell, Berkshire. RG12 1JU.

Review of Cottis-Blandford Cassette Interface.

By J.R. Keneally.

=====

This note describes some experiences in using the cassette interface supplied by Newbear Ltd. (Kit form, current price 12.50), and may be of use to anyone wishing to upgrade the speed and reliability of the cassette interface on their Nascom 1.

Technical details.

1. Uses CUTS type encoding (one - 2400Hz, zero - 1200Hz).
2. Capable of running at 300 to 2400 bits per second.
3. Fairly easy to interface to the Nascom, though the method suggested by Newbear can be improved upon (eg: see below).

Value for money ?

The kit seems reasonable value for money, but there were several pitfalls for the unwary. The two main ones were:

1. The printed circuit board lay-out is poor, and must be very carefully checked for bridges across tracks. Also, it is very easy to cause bridges when soldering certain connections.
2. The documentation is wrong in places, and vague in others. For example, two of the board terminal points were labelled incorrectly on the board lay-out diagram, and it was not stated anywhere that all the output terminal points had to be wired to go to the outside world via spare inverter-buffers in IC3.

The moral is to cross-check everything against the various diagrams etc.

Does it work ?

On first switching on the finished kit, the transmitter section worked properly, and the tapes could be read back on the lowest speed of 300 bits/sec, though with low reliability. Clearly, there were no wiring faults. The problem was traced to two causes:

1. The amplitudes of the high & low frequency tones on replay were different by a factor of about three times. This was due partly to the low-pass filtering used on the transmitter output circuitry, and possibly also due to poor tape recorder frequency response. When the transmitter output circuit was set up so that roughly equal amplitudes were obtained on replay, a big improvement in reliability was obtained. This required a high-pass RC filter, just the opposite of that provided. Obviously, adjustments are required to obtain good results with particular recorders.
2. The input circuitry from the replay has no means of adjusting the D.C. bias level, so as to give an even mark/space ratio on pin 10 of IC10 (a). This ratio seems to be very critical for high reliability, and the required bias can easily be obtained using a potentiometer across the 5V supply. The trick seems to be to record a section of tape having no data on it, i.e just a continuous high frequency tone at 2400

Hz, and then adjust the bias level to give even mark/space at the lowest possible input level that will achieve stable square waves at 1010. It is also worth reading the article in PCW December 78, which describes the interface in detail.

After making the above twiddles, the interface now runs reliably at 2400 bits/secs., ie 10 times the standard Nascom 1 speed ! I personally have standardised on a speed of 1200, and so far the error rate has been zero. A word of warning, however: you may find that on the highest speeds, the TV display will not scroll fast enough to avoid missing a cassette character. Consequently, if an error does occur during a LOAD, all subsequent lines can be in error.

Connections to Nascom

The documentation recommends various wire links to be made to the serial I/O socket SK2. It makes it easier to retain the ability to use the normal Nascom interface if the UART clock at LINK 4, and the Nascom generated UART input at LINK 3, are also brought out to spare pins on SK2. The normal Nascom configuration can then be re-established, for instance, by wiring a 'dummy' plug to fit into SK2. It will still, of course, be necessary to change the actual tape recorder connections back to the Nascom board.

4MHZ

4MHz Without Wait States

=====

After our comment about 8K Basic only running with a wait state, we have had a number of letters from people saying that their Basic runs fine without a wait state. There are two problems, first the ROM is (unbelievably) dynamic. It achieves its refresh from the CE signal. The first time it's accessed, it's a bit slow, and sometimes doesn't quite make it in time for the read cycle, hence a 'crash'. However, the read cycle refreshed the ROM, so if you are quick about it, reset and restart the system, then the Basic runs perfectly. It won't forget what it was up to unless you write very long 'USR' routines which leave the Basic ROM alone for some time. The book says 2mS, but we've found that a typical Basic ROM won't forget until about 15 seconds have elapsed.

The second problem is the RAM (A) board not being up to 4MHz because of timing problems with the CAS and RAS signals. If you would like your system to run at 4MHz without wait states the fixes given on pages 22 and 23 of issue 7 of the INMC News may be necessary, in particular item 3 at the top of page 23.

PASCAL Notes

PASCAL
=====

by Rory O'Farrell

Notes on the steps towards implementation of a "Tiny" Pascal for Nascom.

The best known of all Tiny Pascal compilers is that published in BYTE, Sept./Oct./Nov 1978, by Chung and Yuen. This is designed to be implemented on a North Star with floppies, and is written in Basic! (Ugh!). It can however be made to run on a Nascom with 8k Basic. The main changes involve the Functions - North Star Basic allows multiple line functions, Nascom does not, but the change is easy. In the original, FNG (x,y,z,) is defined with a multiple line definition - for Nascom load three variables not used elsewhere, say e.g. G1, G2, G3, then GOSUB the line where you have repeated the body of the FNG (x,y,z,) definition (Using G1, G2, G3, as the variables). Each call to the function is replaced by a line consisting of G1=value 1: G2=value 2 G3=value 3: GOSUB LINENO. This is quite effective in the case of FNG, the code generation function, and can be done on the fly, as you enter the program. The two error functions can be handled similarly. The notes in the back of the Nascom Basic Manual on string format conversions between one system and another are most useful, as North Star uses a different convention for MID\$ commands. The program as published in Byte can be fitted in less than 20K - less than 16K if comments are deleted.

The Basic program as published in Byte produces pseudo (p) codes for an ideal machine. Unfortunately Byte no longer make available the Pcode to 8080 conversion program, so one will have to be written. Then to compile a Pascal program there will be two stages - use the 16/20K basic program to produce a Pcode listing, and then use a Pcode to 8080/Z80 translator to produce machine code. This machine code will have a number of calls to runtime support routines (perhaps in Eprom) for functions such as multiply and divide., but Chung and Yuen state that their support package amounted to about 1K, so it should not prove an insurmountable task to write.

This combination of Basic/Machine language programs will not be the quickest way to compile a Pascal program, but it is possible (I hope) and may allow a Pascal compiler written in Pascal to be compiled. For the Pascal compiler written in Pascal, I would direct attention to a new book "Structured System Programming" by Welsh and McKeag of Queens University of Belfast (Prentice Hall International). This contains within it a very nice Pascal compiler, written in Pascal, to produce what they call MINI Pascal. It looks very nicely written, and is modular in form, so that the code generation section which generates code for a 24 bit representation of an ICL 1900 could be rewritten for a Z80 (I'd be interested to hear from anyone who has done/is prepared to do this). Unfortunately, this Mini Pascal compiler uses some Pascal constructs not implemented in the Tiny Pascal, so a certain amount of extension of that will be necessary, before the Mini Pascal can be compiled. A format control program in Pascal was given in PCW April 1980, and might be of use in the editor section of a Mini Pascal compiler. To get the vast amount of source program through the machine may create a problem in memory management, but a little bit of thought should overcome that difficulty - (auto tape start/stop?)

I would be interested to correspond with anyone else working along these lines.

(Ed: We have seen a Pascal for the Nascom advertised and we hope to get a copy for review shortly.)

Z80 Books

Books and the Z80.

by R. O'Farrell

=====

When faced with a limited budget, and a microcomputer which is constantly demanding new peripherals, one is loath to spend money on non essentials. To many people, books come under this heading, and such people are frequently the ones who are to be heard complaining of the difficulties of learning how to program. It is my belief that the correct choice of book or books can have a value disproportionate to its cost. Nowhere is this more obvious than in computer books - some are very very good, others are a waste of the paper on which they are printed. What I intend to do in the following page or so is to give the name, author, and publisher of some books I have read on some aspects of computing, with a very short and highly subjective series of comments on it, only approximate prices are given in pounds sterling, as the Irish pound fluctuates in value against sterling (and we have 10% VAT on books). The views expressed herein are mine alone, and have not been purchased, or induced in any way (although I am open to good offers - I'm considering a floppy disc system).

The Z80 Microcomputer Handbook, W. Barden: Sams (distributed Prentice Hall costs about 6.50) It is a paperback of some 300 pages, and contains a full description of the opcodes of the Z80, and then proceeds to demonstrate their use in a series of short and useful programs. The book then concludes with a description of 4 Z80 systems. A most readable and interesting book, and if you decide to have only one then make it this one.

The Z80 Software Gourmet Guide And Cookbook, Scelbi (7.95 I think) is another in Scelbi's Gourmet Guide series. A complete description of the Z80 codes and discussion on how they may be used, culminating in the presentation of a floating point package, which will fit in two kilobytes. This floating point package is interesting because it can readily be extended from four bytes to as many as you like. There is one bug in it, AND A,1 rather than ADD A,1. If interested in number crunching, then this is for you.

Assembly Language Programing - the Z80: Lance Leventhal: Osborne books Costs about 7.50 and is probably more complete in its treatment than the Gourmet guide. Whether it is as readable is another matter. It has a very complete treatment of the programing of the PIO and the SIO which make it worth having.

Programming The Z80, R. Zaks: Sybex (8.00) is more or less Sybex's answer to the Osborne Leventhal book. It contains 600 pages and includes a page by page description of each of the op-codes (as does Leventhal) which would make the purchase of the Mostek/Zilog 'Z80 Programming Manual' unnecessary. It deals with the PIO and SIO, but only lightly, referring you to the next volume "Z80 Applications" which is not yet in print. I can't wait for that one. It has a number of useful subroutines by way of example, including a Binary search for symbol tables - just after I had written one! In consequence of my disgust at this blatant piece of upstaging on Zaks part I cannot claim any impartiality !

On Basic I found the following to be of use :

Illustrating Basic, Alcock: C.U.P. (Paperback recommended).

Microsoft Basic, Ken Knecht : Dilitium. On general theory, but exceedingly expensive - you have been warned.

Compiler Construction For Digital Computers, D. Gries, J. Wiley. (8.00-10.00 in paperback) probably the bible, but also probably slightly dated ? Not for beginners.

Writing Interactive Interpreters and Compilers, P. Brown: J. Wiley. (8.00). A book which is fun to read, and makes you feel that you could write your own compiler. Emphasis very much on Basic.

Understanding and Writing Compilers, R. Bornat: Macmillan. (6.00) paperback. Much heavier than Brown, but with more meat and also more oriented to Pascal/Algol type languages. Not for the beginner.

Structured System Programming, Welsh & McKeag: Prentice Hall. (14.00) Interesting in that it presents a Mini Pascal Compiler written in Pascal, and shows the importance of structure in writing clear programs.

System Programming, Donovan: McGraw Hill. (6.50). A useful book which looks at the theory and implementation of Assemblers, Compilers, Interpreters and Operating Systems. Although using IBM assembly language, it is quite clear, and forms a good introduction to these subjects.

On the subject of programming with style - has anyone looked inside the Nascom Basic ? It's only a 8080 basic, with input output patches all over it. It provokes the thought of how much faster/more compact it would be if it were written in Z80 code from the ground up. Benchmarks on the Xtal Basic and Nascom Basic might be interesting to compare.

Ads.

CLASSIFIED ADS

=====

For sale, Texas 'Silent 700' terminal, type 745. A completely portable thermal printer/terminal with built-in telephone modem (US standard, but alterable). 96 character ASCII character set. The whole thing is the size of a portable typewriter. Roughly 2 years old (inspection stamps dated Apr 78) and very little used. Thermal paper costs about 1.50 a roll from Texas. Would cost over 1500.00 if bought new, yours for 600.00 or near offer. Phone Dave, 01-427 0840.

Two Olivetti TE318 teletypes. Upper/lower case, ASCII, nice looking typeface, with paper punch/reader. 110 BAUD input. Fully working, but no guarantee though. Offers around 200.00 each. Phone James Roberts, 01-960 5430.

Doctor Dark's Diary

DOCTOR DARK'S DIARY - EPISODE 6.

I don't understand high finance, so I don't know why Nascom called in the receiver. I don't know why anyone would suddenly decide not to invest in something that is obviously a winner, but I do hope we will have it all explained to us. And if my Nascom-ERNIE interface works, I have no doubts at all, I'll lend Nascom my #250,000 prize, if it will help!

Fingers crossed now. As long as these get printed, I'll keep writing them.

FEEDBACK - 1.

Feedback 1 was going to be a short program (9 bytes) to give back the use of the backspace key when using 'L' to load free programs. Unfortunately, I don't seem to be able to get it to work. If the author of the letter can explain this, I will not name him!

FEEDBACK - 2+3

Mike Phillips of Wiltshire and W.S. Lounds of Lancaster both sent in Nas-Sys versions of Intabs. As Mr. Lounds has had his published in the Liverpool Software Gazette, here is the Mike Phillips' version.

```
0E00 ED 5B 0E 0C CF FE 2E 20 02 DF 5B FE 20 20 F5 DF
0E10 6A EB DF 66 EB 01 05 00 EF 3E 20 00 1A FE EF 20
0E20 0A D7 68 A7 28 2B F7 13 1A 18 F8 FE DD 28 28 FE
0E30 FD 28 24 FE ED 28 42 0E 1A 2A 0C 0C D5 11 96 00
0E40 19 D1 ED B1 06 03 28 0B 01 1B 00 ED B1 06 02 28
0E50 02 06 01 D7 38 18 AD D7 32 2A 0C 0C D5 11 AD 00
0E60 19 D1 ED B1 28 DE 2A 0C 0C D5 11 D1 00 19 D1 0E
0E70 0B ED B1 28 DC 06 02 18 DA D7 10 0C 2A 0C 0C D5
0E80 11 CB 00 19 D1 ED B1 28 BB 18 C6 06 01 DF 68 DF
0E90 69 13 1A 10 F8 C9 01 11 31 32 3A C2 C3 C4 CA CC
0EA0 CD D2 D4 DA DC E2 E4 EA EC F2 F4 FA FC 21 22 2A
0E0B 36 CB 06 0E 10 16 18 1E 20 26 28 2E 30 38 3E C6
0EC0 CE D3 D6 DB DE E6 EE F6 FE DF D7 43 4B 53 5B 73
0ED0 7B 09 19 23 29 2B 39 E1 E3 E5 E9 F9 00 00 00 00
```

It gets the 'RST' instructions right, and is even more amazing when you are told "this program was produced by hand disassembling the published routine to give a source program, after which the various calls were modified to run under Nas-Sys. This stage was not too easy for me, as I do not know T4 (or T2 or B-Bug), and a little guess work was called for." Mr. Phillips went on to suggest that it would be helpful if a list of sub-routine calls for the various monitors could be printed. Naturally (being from the Frog Star) I suggested he might wish to do it, and offered to supply him with copies of the T2 and T4 listings. All we need do now is wait.... (Or turn to the article in this issue -Ed.)

FEEDBACK - 4

Dr. L.C. Waring of Holywood, Co. Down sent me a tape of his "Tomcat" Editor Assembler which uses the TDL mnemonics - and he has versions for early monitors and Nas-Sys. So far, I've only had a single try at making it work. With my usual skill, I managed to produce all possible error messages in no time at all. If the same thing happens next time I try to use it, I suppose I'll have to read the instructions! It is hardly necessary to add that I suggested he submit it for inclusion in the program library.

Other people have written interesting letters, and I hope they aren't offended by not being mentioned. Now for some of my own work.

SORTING IT OUT.

There are many ways of sorting data into order, the fastest I have seen is the "Quicksort". The version below is adapted from a demonstration for the Commodore Pet (whatever that is) printed in the Liverpool Software Gazette (nearly as good as this magazine).

```
10 INPUT "NUMBER OF ITEMS TO SORT";NE
15 IF NE<255 THEN 30
20 PRINT "TOO LARGE." :GOTO 10
30 DIM A(NE),ST(10,2)
35 LE = 1: RI=2
40 FOR I =1 TO NE
50 A(I) = 100 * RND (1): PRINT A (I);
60 NEXT I
70 REM NOW THE SORT ITSELF
80 SP = 1: ST(SP,LE)=1:ST(SP,RI)=NE
85 LR = ST(SP,LE): RR = ST(SP,RI):SP=SP-1
90 I = LR:J=RR:X=A(INT((LR+RR)/2))
100 IF A(I)<XTHEN I=J+1:GOTO 100
110 IF X<A(J)THENJ=J-1:GOTO 110
120 IF I>JTHEN 130
125 W=A(I):A(I)=A(J):A(J)=W:I=I+1:J=J-1
130 IF I<=J THEN 100
135 IF I<RR THEN SP=SP+1:ST(SP,LE)=I:ST(SP,RI)=RR
140 RR=J
145 IF LR<RR THEN 90
150 IF SP<>0 THEN 85
155 PRINT : PRINT
157 REM HERE COME THE RESULTS
160 FOR I=1TO NE
170 PRINT A(I);
180 NEXT I
190 END
```

If you want to sort more than 255 items, the array ST() should be larger. DIM ST(30,2) will allow you to sort much larger arrays, whereupon time taken is extended considerably. It remains much faster than a "Bubblesort" however. The same program will sort strings, if A() is changed to A\$(), X to X\$ and W to W\$. At 2MHz it takes about 10 seconds to sort fifty words.

USEFUL POKES IN NASCOM BASIC

To prevent an automatic Newline being put into a printed line after 47 characters, use POKE 4162,255. To restore to normal after the above, use POKE 4162,47. To produce double line spacing on the screen, use POKE 4162,0. This information is from the Liverpool Software Gazette - if you know any "useful pokes", why not send them in ?

BASIC ROM ON MEMORY BOARD.

I must apologise for the ambiguous instruction I gave! I meant bend the pins of the wire wrap socket (NOT the ROM) out and solder to them. I bet you all realised what I meant, anyway!

FREE GRAPHICS SUBROUTINE

```
1000 CLS:X=0:Y=0:W=94:H=42:L=0
1010 FOR I=1 TO W: SET (X,Y):X=X+1: NEXT I
1020 IF L=0 THEN 1040
1030 W=W-1
1040 FOR I=1 TO H: SET(X,Y):Y=Y+1:NEXT I
1050 H=H-1
1060 FOR I=1 TO W:SET(X,Y):X=X-1: NEXT I
1070 W=W-1
1080 FOR I=1 TO H:SET(X,Y):Y=Y-1: NEXT I
1090 H=H-1:L=L+1: IF L<22 THEN 1010
1100 RETURN
```

"Not that old thing again !" you all say. 'Fraid so, but much slower than before.

Eds. comment: This sounds like a great idea for a competition. There will be prizes for:

- (1) the shortest (fewest bytes) and
- (2) the quickest

ways of achieving the above using BASIC. Machine code subroutines are not allowed, nor any crafty playing with the reserved words.

Entries should reach us by 1/8/80, and the nature of the prizes will be decided at some future date !

WHAT ?

Mindblowingwifeannoyingmainsgobblingkeyboardthumpingtapechewingmoneydemandingsquareeyegivingnauseacreatingteethgrindingbyteeating NASCOM
Who said Popsi adverts were exclusive ?

Index

Index to INMC Newsletters 1 - 7

=====

INMC

Committee and how it works	6/25
How it works	5/3
Services	6/0
Software library	3/19, 6/28, 7/35

HARDWARE

4MHz operation, Nascom 1	3/13, 4/11
4MHz operation, RAM (A)	7/22
8K Basic on RAM (A) boards	5/12, 6/15, 7/17
Aztec UHF modulator, fitting of, Nascom 1	1/2
Bit 7 and graphics characters	4/15, 7/10
Bits & PCs graphics review	4/18
Bits & PCs dual monitor and scratch keyboard review	7/8
Centronics printer interface connections and software	1/6
Comp 'joysticks' review	5/10
Comp S-100 bus review	6/11
Documentation errors, Nascom 2	6/9
Doubling tape I/O speed, Nascom 1	1/3, 2/2
Econographics review	7/14
Expansion, minimum, Nascom 1	6/31, 7/24
Expansion, Nascom 1, brief details	4/8
Hardware faults, Nascom 2	5/15
IMP printer review	7/12
Keyboard cabinet review	7/14
Memory plague	3/12, 5/15, 7/17, 7/22
Missing characters on display, Nascom 1	1/2, 6/27
Mk I 2.2 amp PSU correction	1/1
Mk I 2.2 amp PSU improvements	1/1
Multi-processors, start of	6/4, 6/19
NASBUS, brief description	1/1
NMI break generator, Nascom 1	4/14, 2/4
PIO operation	2/6, 6/4, 6/19
Port 0 floating inputs, Nascom 1 (and 2)	2/2
Power supply bussing, Nascom 1 iss B	3/13, 4/11
RAM (B) review	7/23
RS232 to standard Cannon DP25 plug connections	1/4
Serial I/O socket connections explained, Nascom 1	1/4
Snow plough, Nascom 1	1/2, 2/4
Tape I/O corrections, Nascom 1 iss B	1/3
Teletype interface, Nascom 1	1/3
Teletype UART speed setting, Nascom 1	2/3
TV syncs, Nascom 1	4/14
William Stuart graphics review	7/16

FIRMWARE

8K Basic compatibility	4/12
Assemblers, comparative review	5/17
CC Soft level B Tiny basic, brief details	4/6, 4/16
M5 (mini PILOT type) interpreter, mods	6/13
Mushroom Basic, brief details	5/9
NASBUGs and B-BUG, comparative review	3/16
NASBUG T4 description	2/11

NAS-DIS 1.0 review	7/28
Naspen review	6/2
NAS-SYS 1 a brief glimpse	4/21
Resident firmware review at June 1979	3/3
Super D-BUG review	7/29
Tape interface warning (see T1 tape patch), NASBUG T1	1/4
Tiny Basic, brief details	1/6
V & T assembler review	5/17
XTAL Basic, brief details	4/6
XTAL Basic 2.2 review	7/30
XTAL Basic 1.3 update	5/16
ZEAP 1.1 assembler, brief details	1/6
ZEAP 1.1 assembler, review	3/10, 5/17
ZEAP 2.0 assembler, review	5/17, 6/24
ZEAP 2.0 mods	7/19
ZEAP in EPROM	6/8
ZEN assembler review	5/17

SOFTWARE

8K Basic, RESTORE command	4/10
Breakpoint command, NASBUG (and NAS-SYS)	2/4
Centronics printer interface software and connections	1/6
Comparing two 16 bit registers	2/4, 6/7
Display of characters below 20H, NASBUG	2/4
'Fake jumps and calls'	6/7
Input ARGS, NASBUG and NAS-SYS	5/24
Interrupts and NAS-SYS 1	6/8
LDI, LDD, LDIR and LDDR instructions, use of	3/15
Memory map, suggested	6/17, 7/18
Memory map, video	7/27
NASBUG T1 tape input correction patch	1/5
NAS-SYS, correct use of	7/20
NULLS and NASBUG and NAS-SYS	4/16
PARSE routine and NASBUG	2/5
PRS routine, NASBUG	2/4
Sargon Chess	4/16
Software hints, 8K Basic	4/10, 4/13, 4/25, 7/6, 7/24
Software hints, machine code	2/5, 3/15, 4/7, 4/15, 6/7, 6/24
Software hints, Tiny Basic	2/17, 4/9
'T' and 'L' commands to load programs	4/9, 5/8
Tiny Basic, speeding up programs	4/9
Unsupported opcodes	3/5, 4/15, 5/10, 7/9

PROGRAMS

Chase (NASBUG, machine code)	4/26
Crash (NASBUG, machine code)	4/29
Demon (reset jammer) (NAS-SYS, machine code)	7/17
Double mastermind (NASBUG, machine code)	2/15
Fliza (8K Basic)	5/34
Fast array set up (Super Tiny Basic)	2/17
Fruit machine (NASBUG, machine code)	5/27
Go-Karting (NASBUG, machine code)	5/26
Hangman (NASBUG, machine code)	5/28
Hangman (8K Basic)	6/36
House keeping with strings (Super Tiny Basic)	2/17
INKEY\$ function with 8K Basic	4/13, 4/25

Jackpot (Tiny Basic)	5/31
JJ (NASBUG, machine code)	3/26
Length of strings (Super Tiny Basic)	2/17
Life (NASBUG, machine code)	4/28
Lollypop lady (NASBUG, machine code)	2/10
Lord (Super Tiny Basic)	5/33
Moonlander (NASBUG, machine code)	3/9
NAS-SYS, 8K Basic and the top line of screen	4/14, 4/23
Othello (NASBUG, machine code)	3/24
Piranha (NASBUG, machine code)	5/29
Piranha (NAS-SYS, machine code)	6/33
PRINT USING function with 8K Basic	7/18
Random Buzz Word Linker (NASBUG, machine code)	4/27
Random 'shuffles' and 8K Basic	4/17
React (NASBUG, machine code)	3/23
Repeat keyboard routines (NASBUG and NAS-SYS)	5/20, 6/21, 7/8
Reverse (NASBUG, machine code)	3/25
Tiny disassembler (NASBUG T4)	6/14

REVIEWS

Assemblers, comparative	5/17
Bits & PCs dual monitor and scratch keyboard	7/8
Bits & PCs graphics	4/18
Comp 'joysticks'	5/10
Comp S-100 bus	6/11
Econographics	7/14
IMP printer	7/12
Keyboard cabinet	7/14
NASBUGs and B-BUG, comparative	3/16
Nascom 2 assembly	5/13
NAS-DIS 1.0	7/28
Naspen	6/2
RAM (B)	7/23
Resident firmware at June 1979	3/3
Super D-BUG	7/29
V & T assembler	5/17
William Stuart graphics	7/16
XTAL Basic 2.2	7/30
Z80 Instant Programs, book	7/33
ZEAP 1.1 assembler,	3/10, 5/17
ZEAP 2.0 assembler,	5/17, 6/24
ZEN assembler	5/17

COMPUTERS ? They're all Arabic to me !!

پیش شش ۷۰

Assemblers

ASSEMBLERS & NAS-SYS

By R. O'Farrell

=====

Pending the availability of Zeap 2.0 at a reasonable price to existing users of Zeap 1.0 who have changed to Nas-Sys, a patch tape from DJ Software renders your existing ZEAP fully usable on Nas-Sys. DJ Software are the suppliers of Revas tapes (blessed by many who couldn't wait for PCW to finish publishing the program), and this patch tape (price 7.50) will probably prove more useful. The conversion is child's play for T4/B-Buggers (sounds very doubtful, that), but a little bit trickier for T2 users. Under your old monitor, you write a copy of the Zeap to tape using the 'W' command on T4/B-Bug, or 'D' on T2. Then you power down and change from the old monitor to Nas-Sys. Load the Zeap tape you have just written ('R' command on T4/B-Bug) Here is where T2 users have difficulty. They will have to write a small input routine to scan the input for Nascom backspace, clear screen and new line, and substitute ASCII characters for them. In any event, when the Zeap is loaded, do NOT attempt to execute it. Instead, play the patch tape into the machine. (No need for any commands). The patch tape will perform a series of modifications and insertions, which consist for the most part of replacing all system calls with the Nas-Sys 'RST' and 'SCAL' number, and a padder NOP to fill the three byte space. There are also a number of calls to an approx 300 (dec) byte extension at 1B08H. All this is done automatically by the tape. When the tape has finished, you can now enter Zeap (E F00) in the usual way. The tape is fully commented, in the sense that it tells you at all times what it is doing (but not necessarily in very great detail) and it is accompanied by a five page commentary sheet, which is hard copy of the tape.

On entry, you will notice one difference: the top line message is now on the second line from the top, and the serial number of your copy is replaced by "OPTION 00", with whatever option is set displayed. This top line performs a very useful function. It blanks out when the Assembler cannot accept a command, and restores when it can. The V (list) command now lists in blocks of eight lines (Newline to continue, any other key to terminate) and the F (find) command allows you to tell it what line to start or continue the search at. The N (Nas-Sys) exit command, when used with two arguments, allows lines of entry to be written to tape or externally with no line numbers - a simple text handler.

A very useful patch, allowing one to dispense with the schizophrenia of using and thinking for two monitors.

Zeap 2.0 would be useful as it gives a symbol table, (and possibly paginated listing ? (no - Ed.)), but having spent 30.00 on Zeap 1 I doubt if many will be tempted to spend the same amount on Zeap 2.

Those who are tempted to disassemble their Zeap 1 with Revas, or a similar disassembler, will find that F00H to F70H (approx) consist of workspace and data areas, and 1003H to 1268H consists of the dictionary. These should be defined as data areas for the disassembler. There may also be minor unpleasantness wherever a Restart is used under Nas-Sys, but they can be coped with.

I have been using Zen for over a year (as well as Zeap), and have patched Zen to run on Nas-Sys. It is a very fast assembler with one drawback - a lousy error handling function ! It aborts assembly on the first error it finds. In "Writing Interactive Interpreters and Compilers", P.J. Brown points out that a compiler spends most of its time in the error handling mode, and should be designed to do this well - Zen definitely doesn't. I have been tidying it up, and have written a number of Pseudo ops; TITLE to title a printed page, LIST and UNLIST to allow selective printout of sections during assembly. RCAL and SCAL to make life under Nas-Sys easier. I have also taken the DB, DW, DM, pseudo ops to Zilog/Mostek standards (DEFB etc). My next miracle will be to tidy up the error handling, so that it will do the first pass, and point all first pass errors. Then (perhaps) a macro facility ?

STOP !....(please?)

HALT !
=====

By Richard Beal

If you want to halt a program this can be done by including the code 76 in your program. When this HALT instruction is executed, the program counter stops being incremented and an endless stream of NOPs are executed. The Z80 CPU detects that it has halted and a LED on the Nascom lights to show that it is in a HALT state.

There are two ways to leave a HALT state. The first and most commonly used is to press Reset, which restarts the computer and reinitialises Nas-Sys.

The second method is for there to be an interrupt. In fact, if you were writing a program which was to do nothing at all except when an interrupt occurred, you could just code a HALT instruction. Note that if you intended the Z80 to re-enter the HALT state after the completion of the interrupt, then the code would be as follows:

76 18 FD

as the address of the HALT (pushed on the stack) at the start of the interrupt, is POPed off and incremented when a RETI instruction is encountered.

This explains why you can Single-step through HALT instructions. The HALT is executed once, but the NMI (Non Maskable Interrupt) generated by Single-stepping jumps out of the HALT state at the end of the instruction.

If you try to execute a program which has a HALT instruction as the first byte, you will find that it will not HALT. The reason is that the Execute command in fact Single-steps the first instruction and then executes normally. So if the program was:

76 it would not HALT

But if it was:

00 76 it would HALT.

BASIC

BASIC PROGRAMS THAT RUN THEMSELVES

by Richard Beal

=====

The Generate command in T4 and Nas-Sys provides an easy way of writing a tape which contains a machine code program. When the tape is fed in, the program automatically starts to execute.

Mr. J. R. Hunt has kindly written to us to point out that the same can be done for Basic programs. It is necessary to do the following, once the Basic program is in memory.

- a) Set up an output table for cassette output as well as output to the display, and activate this table.
- b) Use the PRINT statement to output the necessary commands to tape which are fed back in. For example PRINT "CLOAD" will output the CLOAD instruction to tape
- c) Return the output table to normal.

The idea can be extended further with a string of commands:

```
Print "MONITOR" (in case Basic is already in use - will give
                  an error message if it isn't)
PRINT "J"       (to run Basic)
PRINT          (reply to 'Memory size ?' question)
PRINT "CLOAD"   (to load program)
CSAVE "A"       (to save program to be loaded)
PRINT "RUN"     (to run the program that has been loaded).
```

It is necessary to insert delays between these commands to allow time for them to be processed when they are fed back in. The NULL command should be used for this. It is easiest to put all this into a little program at the end of the main Basic program, and run this when you want to create a self loading tape.

Below is a listing of a complete program which will create a self loading copy of itself, as an example. The tape produced can be fed in either after switch on, or when the computer is already in Basic. It can be made quite small by removing all the comments.

LIST

```
100 REM ** MAIN PROGRAM
110 INPUT "Type a number";A
120 PRINT A"times"A"is"A*A
130 GOTO 110
30000 REM ** PROGRAM TO WRITE A SELF-LOADING TAPE
30010 REM * SAVE ADDRESS OF OUTPUT TABLE 0C73H
30020 TS=DEEK(3187)
30030 REM * SET UP OUTPUT TABLE AT 0D00H
30040 POKE 3328,101:REM 65H FOR CRT
30050 POKE 3329,111:REM 6FH FOR SRLX
30060 POKE 3330,0 :REM END OF TABLE
30070 REM * POINT TO NEW TABLE
30080 DOKE 3187,3328
30090 REM * SET END OF LINE DELAY
30100 NULL 100
```

```
30110 REM * COMMANDS
30120 PRINT" REM * SELF LOADING BASIC PROGRAM"
30130 PRINT"MONITOR"
30140 PRINT"J"
30150 PRINT
30160 FOR I=1 TO 1000: NEXT: REM * DELAY
30170 PRINT"CLOAD"
30180 REM * SAVE PROGRAM, CALLING IT "A"
30190 CSAVE"A"
30200 REM * RUN COMMAND
30210 PRINT"RUN"
30220 REM * SET BACK TABLE AND NO NULLS
30230 DOKE 3187,TS
30240 NULL 1
30250 END
Ok
```

interface software

LEVEL A BASIC

A 2K BASIC Interpreter for the unexpanded Nascom 1

Level A BASIC has been written to give users of Nascom 1 systems the ease of programming in BASIC, without the need for extra memory or expansion boards. It is supplied in two 2708 EPROMs, which are plugged into the EPROM sockets normally occupied by the Nascom Nasbug or Nas-Sys monitor program.

LEVEL A Features.

Integer BASIC - range -32767 to +32767
Variables - A to Z
Array - single dimension
Functions - ABS, RND, RAM
Arithmetic operators - *, <, /, =, -, <>, +, <=, >, >=
Statements - LIST, RUN, NEW
Commands - REM, LET, PRINT, INPUT, IF, GOTO, GOSUB,
RETURN, SAVE, STOP

Level A BASIC was written by CCSOFT. It is supplied in two 2708 EPROMs and comes complete with a user manual that includes fitting instructions and example programs.

LEVEL A BASIC £25.00 + VAT + 30p P&P

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

CASSETTE HINTS

Cassette I/O Reliability =====

A few people have written to us and have indicated that they experience unreliable cassette I/O. As far as we are aware, if everything is set up correctly, very very few errors should be encountered at normal speeds. The same applies at double speed on a Nascom 1 or 2400 Baud on a Nascom 2.

If you have cassette I/O reliability problems then the check list below may help:-

- 1) Make sure that the tape transport is clean (wiping with a tissue will do for starters) i.e. that the heads, the pinch roller and capstan are free of loose tape material.
- 2) Check for earth loops. If the cassette recorder is earthed, and the Nascom is earthed (it should be) and there is an earth connection between them, then problems may arise. The safest one to disconnect is the one(s) between the Nascom and the recorder. That's the safest, the best one to remove would be the cassette mains earth. But on your own head be it.
- 3) The settings of the volume and tone controls can be quite critical. Too much volume on record or play back may cause the amplifiers in the cassette recorder to deliver a distorted signal. Too little will not drive the cassette interface on the Nascom. On recorders with one tone control (normally a treble cut) this should be set at or near the max. treble end (i.e. no treble cut). On others a flat or zero setting will normally be best.
- 4) On a Nascom 2, VR1 is best set up using an oscilloscope. Record about five minutes of tone (plug in the recorder, and just switch to record. Play back the tone, monitoring TP19 with the 'scope. Adjust VR1 to give as good a square wave as possible.
- 5) Use a reasonable quality tape. Since most cassettes are meant for music/speech, using them to store very fast transitions is not ideal. Some cheap tapes give a very low play back level even when recorded at a good level. This is not critical for sound but can cause problems for data.

A suggestion we have received from a member in Nottingham is to load the cassette output (when the speaker output is being used) with an 8 ohm resistor (or close to, but above this value) by connecting it across the output plug. See 'Cut that Noise' elsewhere in this issue.

Cut That Noise =====



If you are using a signal output from a cassette recorder that still has the speaker blaring away on play back then you may be interested in this idea from a member in Nottingham.

Connect a Light Emitting Diode (LED) and a small ordinary diode back to back with an 8 ohm resistor (or close to, but above this value) in series across the speaker output. This will then give a visual indication of 'tone - data - nothing' being output.

2400 Baud Cassette, 300 Baud TTY for N2

=====

Here is a very simple mod for Nascom 2 owners to give cassette I/O at 2400 Baud.

Connect TP20 to TP4 and TP21 to TP5. If you now select external UART clock you have the cassette interface running at 2400 Baud. The switch settings of LSW2 are :-

Switch -	1	2	3	4	5	6
Setting -	either	up	up	either	up	up

where switches 1, 2 and 3 select the transmission speed to the cassette and 4, 5 and 6 the receiving speed from the cassette. Switch 7 should be down to select cassette input. (Up selects TTY input.)

To make 300 Baud TTY available is a little more difficult but since many acoustic couplers and teletype devices work at this speed it may be necessary. Transmission is the easiest since only the clock speed of the transmission side of the UART needs setting and this is readily achieved by the existing switches on LSW2.

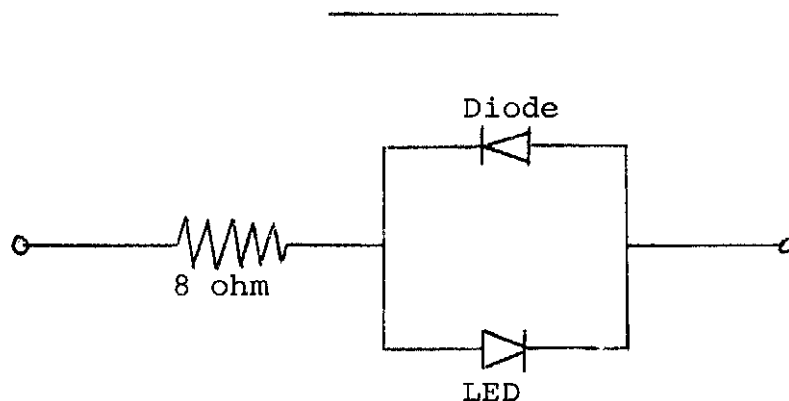
	1	2	3
110 Baud	either	up	down
300 Baud	down	down	either
1200 Baud	up	down	either

and with the mod above

2400 Baud	either	up	up
-----------	--------	----	----

Reception cannot be achieved in the same way because the UART receive clock for tape is derived from a Phase Lock Loop (PLL). The TTY input must have a 'hard' clock, as the PLL can not regenerate the clock from the incoming TTY signal. Therefore, to get different receive speeds for TTY input the appropriate clock signals must be gathered from various places and applied to the UART. This may be done by connecting the required clock from the input side of the transmit speed selection switches to the external receive clock input (TP5). The most usual TTY speed required is 300 Baud and the clock for this may be picked up from IC32 Pin 4 with LSW2 switch 1 down or from IC31 Pin 9. A switch will then be required to select between the 2400 Baud cassette clock (coming from the PLL circuit) and the 300/1200/2400 Baud TTY clock going into the external receive clock input (TP5).

Happy switching!



ZEAP 1.1→2.0

HOW TO CONVERT A ZEAP 1.1 FILE TO ZEAP 2.

=====

(The new workspace starts at 2000 HEX).

1. Make a note of the number displayed on the top right of the screen, when the file is loaded under ZEAP 1.1. Call this value TTTT.
2. Load the file and copy it to the new location by entering:
I 1B0D 2002 8000
3. Examine the address at 2002 - 2003. Call this LLLL. (Remember low order byte is stored first.)
4. Calculate TTTT - LLLL by entering:
A LLLL TTTT
and looking at the second value displayed. This gives the symbol table area. Put this value at 2002 - 2003.
5. Calculate LLLL + E4F5 by entering:
A LLLL E4F5
and looking at the first value displayed. This gives the file length. Put this value at 2000 - 2001.
6. Enter I 2000 0C80 6 (to store the start of the file).
7. Load ZEAP 2.0, and execute it by entering:
E 1000 8000 2000
(E D000 8000 2000 for the EPROM VERSION).
8. Enter N to exit from ZEAP.
9. Enter I 0C80 2000 6 (to restore the start of the file).
10. Execute ZEAP 2.0 by entering:
E 1003
(E D003 if you have the EPROM version).
11. Check that TTTT + 0F45 equals the value now displayed on the top line as Free = XXXX.

WARNING~8A

It has come to our notice that there is a potential fault on the Nascom 8 amp PSU. For some reason the '0 volt' rail is not connected to mains earth, which means that the chassis is 'floating'. Now this is alright until the nurk assembling the power transistors on the PSU heatsink leaves out the insulating washers. Connect it all up, and the chassis floats at about 12 volts. Earth the chassis, and bang !!!! Nascom say this has now been put right. But you have been warned.

NAS-SYS 3 - IMPROVED VERSION OF NAS-SYS 1

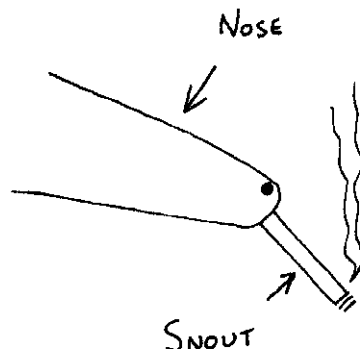
- =====
1. All the keys on the keyboard automatically repeat when held down. (Not the @ key.) The initial repeat delay and the repeat speed are adjustable.
 2. All routines are interruptable, so that interrupt can be used while executing NAS-SYS routines.
 3. The CRT routine allows data to be output anywhere in memory, so headings can be output to the top line of the display.
 4. The Read command has an optional parameter which allows cassette tapes to be read into any memory locations.
 5. The Tabulate command is enhanced in three ways. Firstly, ASCII values of the bytes are output as well as the usual hexadecimal tabulation. Secondly, a fourth parameter can be used to specify the output of additional values on each line, to allow for printers of different widths. Thirdly, a fifth parameter allows suppression of either the hexadecimal output or the ASCII output.
 6. All NAS -SYS routines can be single stepped, which makes it easier to test a program which uses NAS-SYS routines. By using the repeat keyboard feature, high speed single stepping is possible.
 7. The register display is enhanced so that it shows the two byte value pointed to by each of the main registers.
 8. Output from the Modify command is displaced two characters to the right, to improve readability.
 9. The External (X) command has additional options, and no longer fails to output nulls. This enables the NULL command in BASIC to work correctly.
 10. There are three new commands. P displays the stored user program registers. D executes a program at D000H and Y executes a program at B000H.
 11. The cursor blink rate is adjustable.
 12. There are three new routines. Repeat keyboard scan, Output two spaces, and a new routine which can execute any other routine.
 13. Both on breakpoint and on NMI, control passes through the \$NMI jump before displaying the registers, allowing a program to take alternative action.
 14. The B 0 command turns off the breakpoint completely, so that with appropriate hardware NAS-SYS can be executed in RAM.
 15. Support in NAS-SYS itself for the use of paper tape has been removed, so there is no longer a Load command and the Tabulate command does not output a checksum.
-

VERY OFFICIAL

To whom it may concern

Last month, in issue 7 of this publication, certain allusions were made concerning our client Lawrence, hereinafter (3 gns.) referred to as 'the long-haired Weirdo', and the unpublished failure of H.M. Police National Computer Unit of no fixed security. Notwithstanding hereinunder (3 gns. each) any further charge that may arise from analysis of certain substances taken from our client at the time of his arrest, viz. 2½ kilos of floppies (we understand these to be similar to uppers, downers, benders and twisters), we wish to make clear that of the current charge, that of obstructing PNCU's operating system in the course of its duties, he is INNOCENT because THE SOFTWARE WAS WRITTEN BY SOMEBODY ELSE quite possibly in the pay of the MAFIA (North London branch).

Incidentally, our client is not afraid to show his snout; it looks like this:



Allegations as to its contents may be made to the Editor; the first correct answer will win a prize.

Goosequill, Goosequill, Goosequill, Gasse-Turbyn
and Goosequill

Practising Solicitors
Expert Turf Accountants