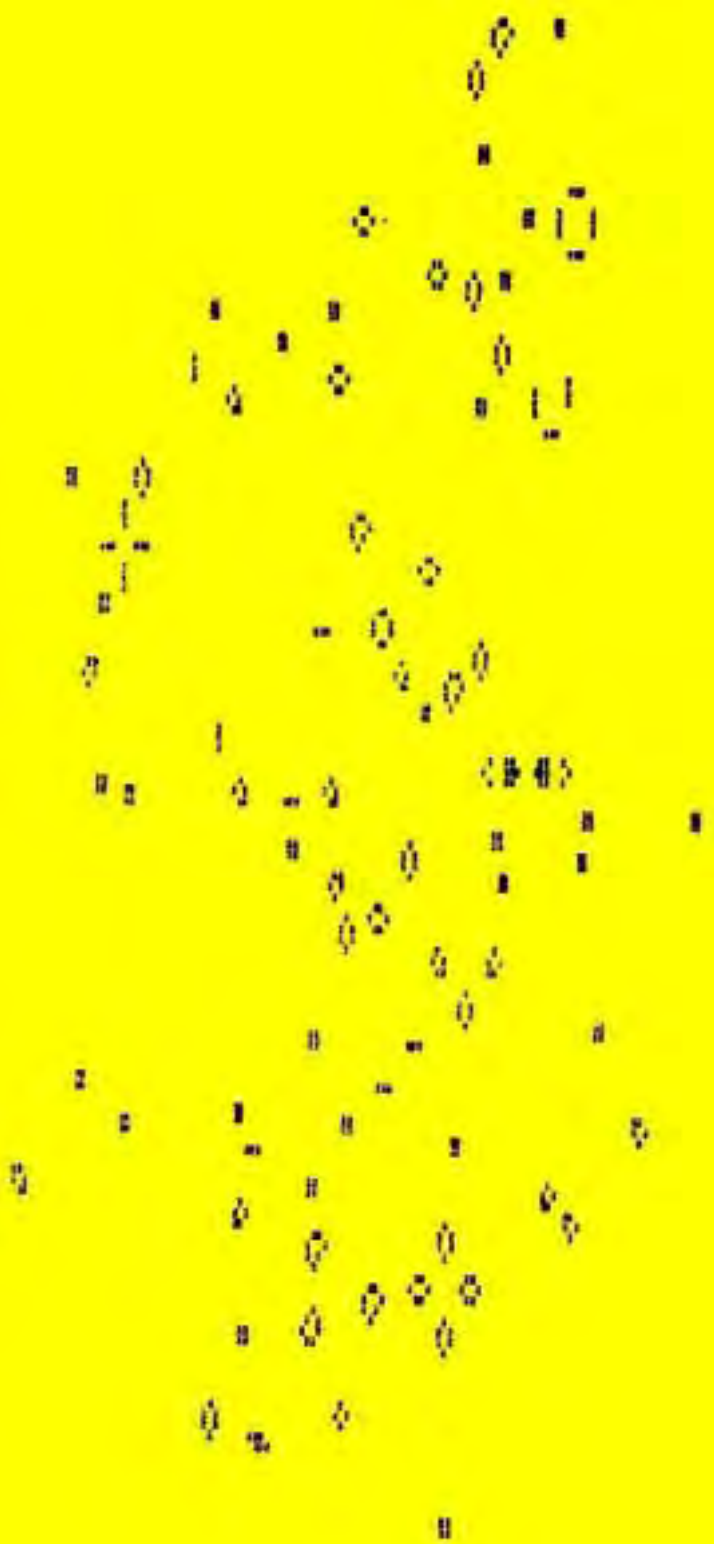
μP

MICROPOWER VOLUME 1, NUMBER 2



September, 1981 95 p

PROGRAM POWER PRICE LIST

ALL PROGRAMS SUPPLIED ON CASSETTE IN CUTS/KANSAS CITY FORMAT

THE SPACE SYNDROME

Lunar Lander Supreme (16K/B/G)	£9.95
Startrek II (32K/G/B)	£9.95
Invasion Earth (MC/G)	£8.95
Invasion Earth (MC/G – sound chip)	£10.95
Alien Labyrinth (16K/B/G)	£6.95
Super Startrek (16K/B)	£6.95
Cliff Invasion (B/G)	£6.95
Space Fighter (B/G)	£5.95

SOUND OUTPUT

AY-3-8910 Sound Chip	£6.95
60 Page Data Manual (No VAT)	£2.25
Sound Chip Interface Board	£13.50
Sound Chip Demo Program (MC)	£5.95
Audio Board and Speaker	£10.75
Music Box (16k/B)	£7.95
Road Race (MC/G)	£4.95
Cowboy Shoot Out (MC/G)	£3.95
Musical Break-out (MC/G)	£3.95

BOARD GAMES

GAMES GRAPHIC ROM	£12.90
TWO SOCKET (2716) ADAPTOR	£7.45
GAMES GRAPHICS ROM/ADAPTOR	£16.90
Sargon Chess Book	£11.70
Book with program	£19.50
Book/Prog./ROM/Adaptor	£35.00
Draughts (B/G) *	£7.95
Backgammon (16k/B/G) *	£7.95
*Please state Ordinary or ROM version	
3D Noughts & Crosses (B)	£3.95
Tantaliser (B/G)	£3.95
Minotaur (16k/B)	£3.95
Reversi (B)	£3.95
Submarine Chase (B/G)	£3.95

COTTIS BLANDFORD

Add a Cottis Blandford cassette interface to your Nascom 1 for reliable and fast loading of CUTS tapes at 300, 600, 1200 and 2400 Baud (NASCOM 2 format)

Kit £14.90

SPECIALITIES

WORDEASE Word Processor (MC)	£25.00
Basic File Handler (MC)	£17.50
Club Membership (32k/MC)	£9.95
Constellation (16k/B)	£6.95
VORTEX Graphics subroutines (MC/G)	£8.95
Mini Toolbox (MC)	£5.95
Graph Plotter (B/G)	£4.95
Vocabulary Tutor (B)	£5.95
Mathspack (32K/B)	£7.95
Nascount Personal Finance (16k/MC)	£9.95
Xtal Basic 2.2 (MC)	£35.00
Nascii (B/G)	£4.95
Prompt (B)	£5.95
Super life (MC)	£6.95
Indexed File Handler (B)	£5.95
Biorhythm (B/G)	£3.95

COMPUTER ASSISTED LEARNING

WIRRAL PILOT v4.0 (MC)	£12.50
Butterfly (B/G)	£5.95
Reading Test Cards (B/G)	£4.95
Abacus (B/G)	£3.95

FAST INTERACTIVE GAMES

Serpent (MC/G)	£5.95
Driver (B/G)	£4.95
Demonoes (B)	£3.95
Lumber jack (MC)	£3.95
Slalom (B/G)	£3.95
Sheepdog Trial (B/G)	£3.95
Death Run (B/G)	£3.95
Dracula's Castle (B)	£3.95
Secret Agent (B/G)	£3.95

MISCELANEOUS

Keys of Kraal (Adventure) (24K/B/G)	£8.95
Blackjack (B/G)	£5.95
Labyrinth (B/G)	£4.95
Spider (B/G)	£4.95
Mindbender (B)	£4.95
Fruit Machine (B/G)	£4.95
Beetle (B/G)	£3.95
Stockmarket (B)	£3.95
Hammurabi (B)	£3.95
Scramble (B)	£3.95
Code Breaker (B)	£3.95

B = Nascom Basic (state ROM or Tape)
MC = machine code
G = Nascom Graphics

The programs require 8K RAM unless otherwise stated

Please add 55 p per order post/packing, and VAT at 15%.

PROGRAM POWER, 5, WENSLEY ROAD, LEEDS LS7 2LX
Tel. (0532) 683186

CONTENTS

EDITORIAL

We have received quite a lot of feedback from the first issue by now, and most of the comments have been favourable. In fact I am left with a sneaking suspicion that all the people who liked the magazine have been kind enough to write and say so, while those who thought it not worth reading haven't bothered to communicate their opinions.

It is very pleasing to know that a particular article has been of interest - we can use the letters to persuade the author to write for us again - but we would also like to know what displeased you. If you can be abusive wittily we shall publish your letter.

Of course, the best letters, complimentary or uncomplimentary, are those that end " . . I am enclosing an article for publication.". We can never get too many of this type of letter, so keep sending them in.

We shall be producing two more issues this year, in November and December, and we have some very interesting articles planned. For example, we have the design for an interface which reads TRS 80 tapes, together with the software to control the interface and then convert TRS 80 Basic programs to run under Nascom Basic; a design for a 2708/2716 programmer; a machine code Lisp interpreter; a program to interconvert Crystal Basic and Microsoft programs; and the usual selection of hardware and software tips.

LETTERS

Dear Sir,

Congratulations on the first issue, although it was possibly a bit 'heavy'. Your printer is obviously capable of correspondence quality print in condensed mode, but the cumulative effect of continuous bold print is to make the text harder to read. I know that they are expensive, but it would look much nicer with a daisy-wheel printer, surely you can scrounge one somewhere.

The one real criticism that I would make is that after devoting a substantial portion of the magazine to PCG design, the circuit diagram was unintelligible to a non-technical reader. The principle seemed simple enough and was well explained, but I think that a full circuit diagram would have been better or - dare I suggest - a Vero layout. It would have taken space, and possibly contributor-badgering, but you are liable to brickbats from readers who misunderstood the project as described.

I am currently writing an article on a very simple project which I will send along to you. We mustn't forget that a growing number of Nascom owners have bought their machines built and are not part of the merry band of track-hackers that some of these modifications are aimed at.

Yours sincerely,

C. L. Corner
Royston

* We had to publish this letter because it's the most critical one we received. As we pointed out in the article on the PCG, the construction involves a lot of tedious hard-wiring of the address and data lines, and if you don't fully understand what is neessary then it is best not to start. I agree entirely about the printing, so if any reader has a spare daisy wheel printer . . .

Dear Sir,

Here is a query for your reader. Does anyone know of a simple circuit to give the equivalent of a typewriter shift lock key for the Nascom 2?

Yours sincerely,

R. C. Taylor,
Penrhyndeudraeth

* The keyboard is software controlled, so the addition of any shift lock circuit will mean changing the monitor EPROM software. There is a spare input line to port 0 (bit 7), and if this was connected to a switch so that it could be at + 5 volts or 0 volts, the state of this line could be used to invert the action of the shift key by a slight modification to the monitor. Have our readers any other ideas?

Dear Sir,

I at present own a Nascom 1, to which I am in the process of building on more memory. I intend to install a Gemini EPROM G805 card which will carry Enertech 16K f.p. Pascal or Hisoft Naspas (12K). Have there been any article comparing these compilers? Are they based on the Jensen-Wirth or UCSD Pascal? Does the Enertech compiler generate P-code? Has any Nascom user done what I

intend to, and what were his problems and their solutions?

Yours sincerely

D. W. M. Fawcett
London

* Surely one of our readers can answer some or all of these questions.

Dear Sir,

Could readers recommend a cassette recorder which will reliable record and play back at 2400 baud and which is available from a retail outlet in this country,

Yours Faithfully,

T. Richardson,
York

* I have used a Hitachi TRQ 295R for computer tapes since I got my Nascom, and it is very reliable at 2400 baud (it 'almost' worked at 4800 baud). It would be interesting to hear from readers on this one, so that we could get a consensus on the 'most reliable' recorder.

Dear Sir,

I was very interested in the 'Snowdinger' article in issue 1 of the magazine. I have a Nascom 2, and although screen disturbance is a minor problem it is nevertheless present. In fact. it seems to become more pronounced after the machine has been on for an hour or two. I'd be very glad to see a article for the Nascom 2.

Another hardware request: is there any simple way to recover the bottom two lines of the Nascom 2 graphics characters,

Yours faithfully,

M. Bisacre,
Windsor

* Yes indeed there is. A modification, involving bending a few IC pins and adding half a dozen wire links, has been published in the Liverpool Software Gazette. It not only recovers the bottom lines, but also improves screen stability. Are enough people interested for us to get the original authors to write a short article for Micropower?

Dear Sir,

I would like to see published in the magazine information (or where to get it) on the keyboard supplied with the Nascom 2. As you know, there is no diagram or info in the kit, and I feel this could be of use to many N2 owners,

Yours faithfully,

P. Harvey
Redruth

* The first article in this issue contains a circuit diagram of an 'upgraded' Nascom 1 keyboard, which is identical with the Nascom 2 keyboard.

*** + * + * + * + * + * + * + * + * + ***

NASCOM 1 KEYBOARD UPGRADE
by Zebedee
(OR HOW TO LOSE CONTROL)

The basic instructions for this modification have already appeared in the series 'Nascom Notes' in the Liverpool Software Gazette, but it was felt that a more detailed article would perhaps encourage the less intrepid enthusiast and show that, though hardware changes are not always easy, with a little care and planning most people can produce a Job that they are proud of and that works. This article is intended to reduce the planning side to the assembly of the parts required. The finished product will enable the constructor to make full use of the powerful on-screen editing facilities of Nas-Sys 1 or 3.

WHAT IS NEEDED.

1. Ten keys and key-tops.
2. One 22 ohm resistor
3. One 1 kohm resistor.
4. One 2.2 kohm resistor.
5. Wire (preferably single strand, insulated).
6. The usual soldering tools, solder, a sharp knife, and a fine drill (1 mm).

Most of the above are easy to obtain, but the keys could prove difficult. They must be Licon keys in order to work satisfactorily in the circuit. Do not use simple switches - they will not work. Enquiries at various Nascom dealers drew a blank, in spite of the original adverts offering spare keys at 50p each. However, it was found that a complete keyboard could be bought separately, and simple arithmetic proved that almost five individuals could be supplied from one keyboard. This does mean that the key tops have 'artistic' legends on them, and that a volunteer is required to operate a solder-sucker (so that's how it got it's name!), but the cost came out the same and it worked. A further word of warning - there are straight keys and angled keys on the market. They all work, but they differ in looks. Members of our local club have now devoured two spare keyboards, so the modification is well tried and cannot be too difficult (even ONJ managed it!).

MODIFICATIONS

The final modified circuit is shown in figure 1, and the differences are easily seen when compared with the Nascom 1 keyboard diagram. Figure 2 shows a view of the underside of the keyboard to help identify the locations specified in the instructions. Note that the four leads at the bottom of the key are not equally spaced - pin A is separated by the widest gap from the other three, which are in order pins B, C and D. The following steps now need to be taken:-

a) drill board to receive the new keys
b) cut certain tracks
c) add the components and the links

a) DRILL

It is best (and easiest) to make a template from a small strip of metal, measured off to the pins of a row of existing keys (a length of four keys is sufficient). Drill this

template for the four pins per key, and it is then a simple matter to place these holes over the end two keys' pins in a row and drill new keys in the space beyond, thus ensuring correct alignment. If the new keys are now taken and aligned with their respective holes, they should neatly latch into the support plate of the keyboard.

The new keys are positioned as follows (looking at the underside of the board):-

Two to the left of the top row of keys
One to the right of the second row
One at each end of the third row
One at the right end of the fourth row
Two at each end of the space bar

The last four keys are the most difficult to locate accurately. It should be noted that they are directly below the next row.

N.B. Early Nascom keyboards may have tracks which cross the area required by the new keys. These tracks must be re-routed by means of wire links to allow the drilling (or use this keyboard for spare keys and modify a more recent one). Later keyboards leave significant gaps in just the right places.

b) CUT

A steady hand and a very sharp knife are the requirements. Check twice before you make a cut. Make two about 1 mm apart in each track, then gently prise the between the cuts away from the board.

Remember ... each key has four pins, A B C D

Cut the track leaving each of the following locations:-

"Newline"	key, pin D	( )
"—"	key, pin D	( )
"X"	key, pin C	( )
"Z"	key, pin C	( )
"K"	key, pin C	( )
"L"	key, pin C	( )
"Q"	key, pin C	( )
"O"	key, pin C	( )
"R"	key, pin D	( )
"Z"	key, pin A	( )
"W"	key, pin B	( )

Tick off each operation in the space provided to avoid making any errors. The worst part of the whole job is now completed.

c) ADD

Try to keep the wiring neat and close to the board. A small quantity of adhesive spotted at odd points helps to hold wiring in place. Remember to sleeve

all of the leads and wrap the wire ends around the pins for stability before soldering.

Solder the 22 ohm resistor from the common point of the inner two resistors near IC2 to pin 10 of IC1. ( )
Solder the 2.2 kohm resistor from the common point of the two outer resistors near IC2 to pin 11 of IC1. ( )
Solder the 1 kohm resistor from pin B of the GRAPHICS key to pin 9 of IC1. ( )

Link IC3 pin 8 to IC3 pin 12 ( )
Link IC3 pin 6 to IC3 pin 13 ( )
Link IC3 pin 10 to IC3 pin 11 ( )
Link IC3 pin 9 to IC1 pin 11 ( )
Link IC3 pin 11 to SKT pin 7 ( )

Link GRAPH key, pin A to LEFT key, pin B ( )
Link LEFT key, pin A to UP key, pin B ( )
Link UP key, pin A to DOWN key, pin B ( )
Link DOWN key, pin A to RIGHT key, pin B ( )
Link RIGHT key, pin A to CH key, pin B ( )
Link CH key, pin A to] key, pin B ( )
Link] key, pin A to [key, pin B ( )
Link [key, pin A to 4 key, pin A ( )
Link NL key, pin D to CH key, pin C ( )
Link CH key, pin D to BS key, pin C ( )
Link - key, pin D to CTRL key, pin C ( )
Link CTRL key, pin D to new SHFT key, pin C ( )
Link new SHFT key, pin D to IC5, pin 1 ( )
Link X key, pin C to UP key, pin D ( )
Link UP key, pin C to F key, pin D ( )
Link Z key, pin C to LEFT key, pin D ( )
Link LEFT key, pin C to D key, pin D ( )
Link K key, pin C to DOWN key, pin D ( )
Link DOWN key, pin C to M key, pin D ( )
Link L key, pin C to RIGHT key, pin D ( )
Link RIGHT key, pin C to , key, pin D ( )
Link Q key, pin C to GRAPH key, pin D ( )
Link GRAPH key, pin C to 3 key, pin D ( )
Link O key, pin C to [key, pin D ( )
Link [key, pin C to P key, pin D ( )
Link R key, pin D to] key, pin C ( )
Link] key, pin D to 4 key, pin C ( )
Link Z key, pin A to new SHIFT key, pin B ( )
Link new SHFT key, pin A to S key, pin B ( )
Link W key, pin B to CTRL key, pin A ( )
Link CTRL key, pin B to E key, pin A ( )

That’s the job completed, and although there is a fair bit of work involved, everybody who has done it agrees that it is worth the effort. A partial upgrade would be possible, using only selected keys, but that would mean sorting out another set of links. All the difficult and time consuming work is done in these simple to follow instructions, so . . . what are you waiting for?

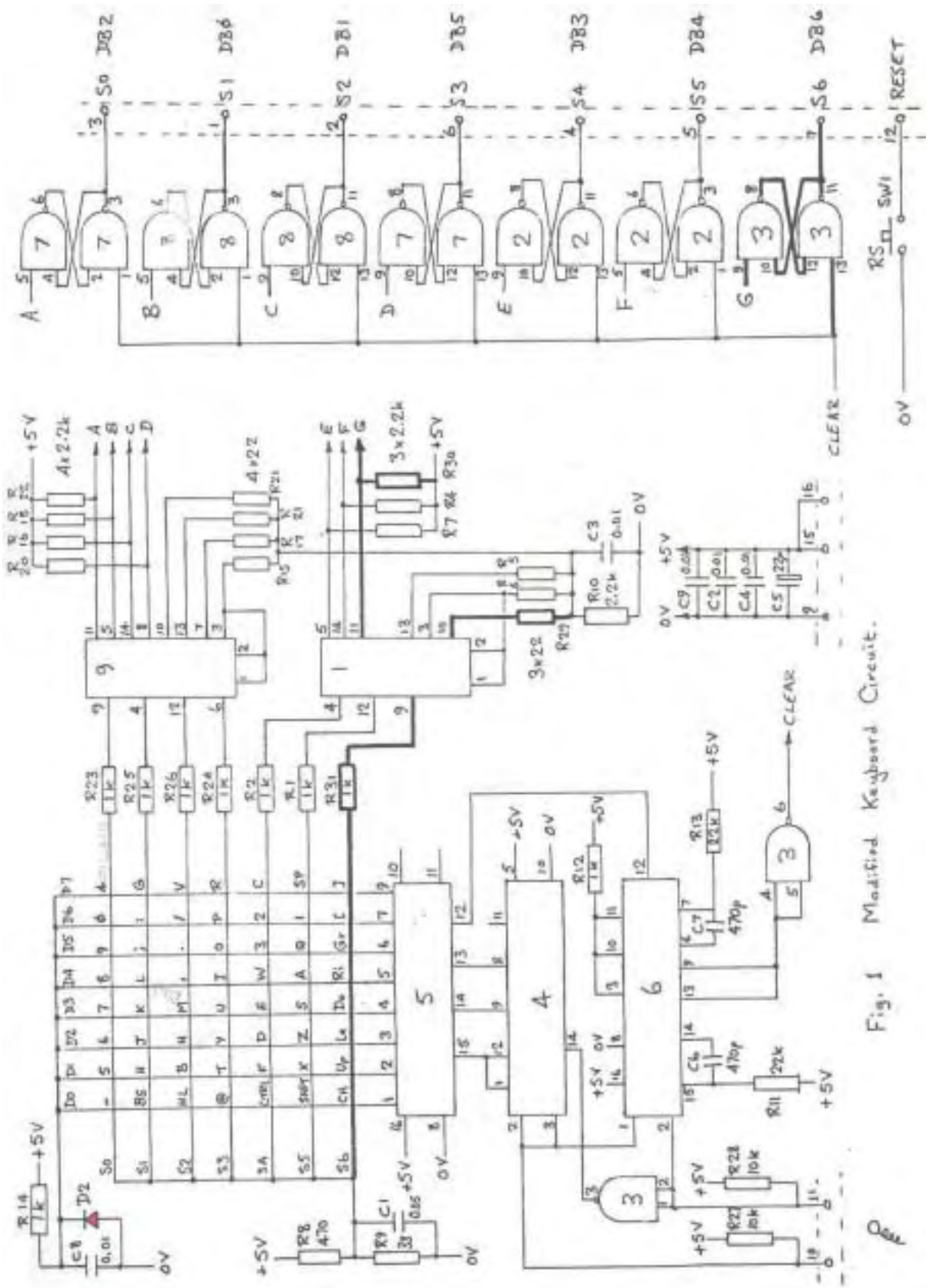
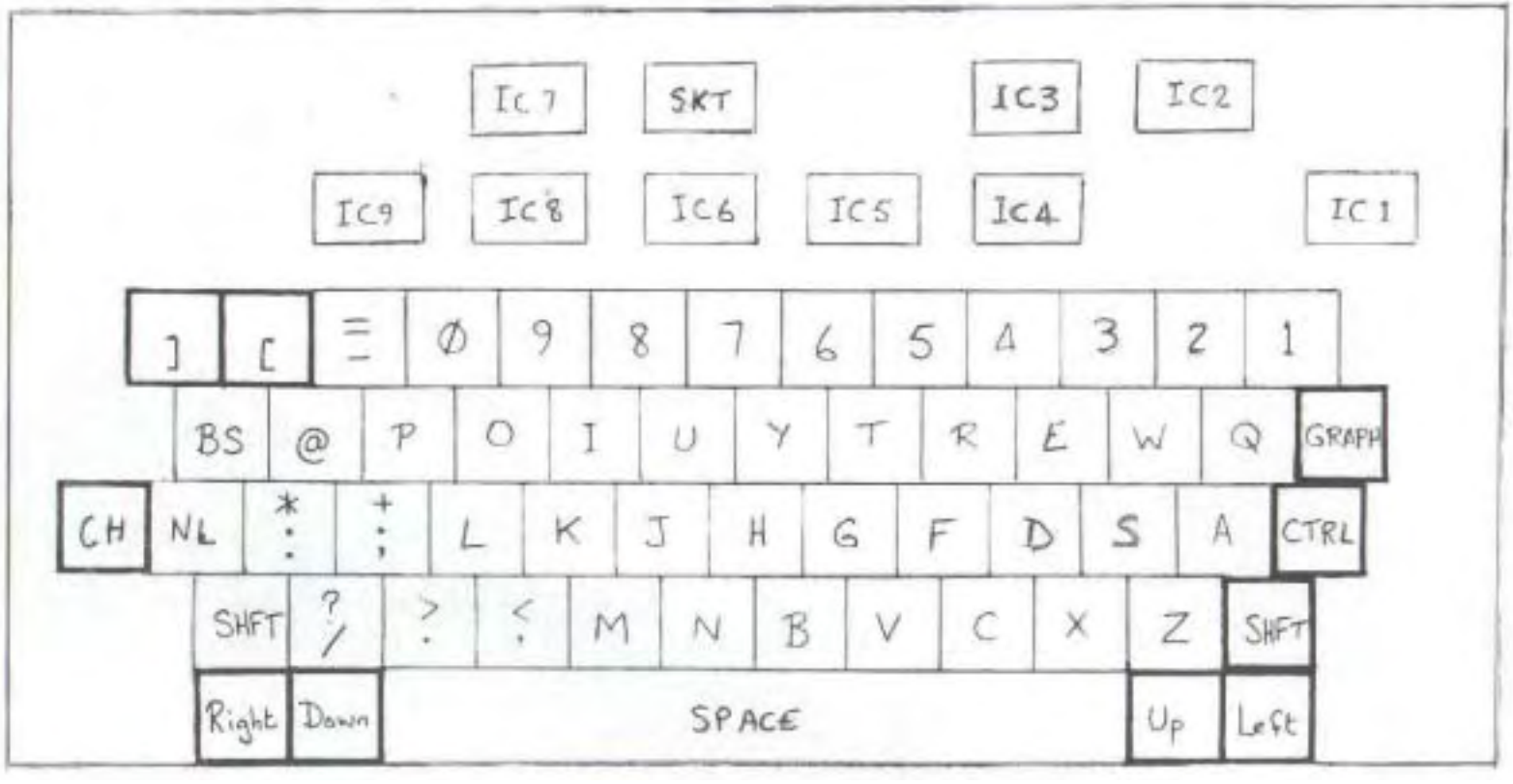


Fig. 1 Modified Keyboard Circuit.



Underside view of key showing pins

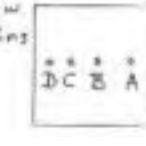


Fig 2 Stylised Representation of Keyboard Underside

NEW EPROMS FOR OLD
by R.A. Gibson

If you look back at the first issue of Personal Computer World, which featured the Nascom 1 on it's front cover, you will find that 2708 EPROMs were offered for sale at a price of £31.15 each. In the latest issue of the magazine 2708s are advertised at £2 each in one off quantities, and 2716s are £2.50. Now a 2708 can hold 1024 bytes of data, while a 2716 has twice the capacity, so on economic grounds it would seem to be sensible to use 2716s in your system rather than the 1K device. This will double the amount of memory that you can cram into a given number of sockets, and there are other advantages too. The 2716 is most readily available in the single-rail form, which requires only a +5 volt supply. The details given in this article are specifically for the single rail 2716. The 2K device consumes less power than the 2708, and therefore runs much cooler. However, the most significant advantage is that while many 2708s do strange things at 4 mhz, I have never come across a 2716 which did not operate perfectly at this clock rate without wait states.

Fortunately the two chips are very similar in pin assignment so the changes necessary to run 2716s on the RAM A card or in the monitor sockets of a Nascom 1 are fairly simple. In both the 1K and 2K EPROMS pins 8 - 1,23 and 22 (in that order) are address lines A0 - A9, pins 9 -11 and 13 - 17 are data lines D0 to D7, pin 12 is Ground, pin 18 is used for programming the EPROMs (also for power-down mode in the 2716), pin 20 is the chip select line, CS, and pin 24 is the +5 volt supply line. This leaves, only two pins which are used differently in the devices for normal operation; pin 19 is the +12 volts supply for the 2708, but is the high address line, A10, in the 2716; pin 21 is taken to -5 volts in the 2708, and +5 volts in the 2716.

Here then are the full details of the modifications necessary to fit 4 x 2716 to a RAM A card, which means that this card can carry 8K of EPROM data. The description assumes that you are holding the RAM card in front of you with the plated edge connector at the bottom and the component side away from you, the soldered side towards you.

1) Disconnect pin 19 of the EPROM sockets from the +12 volt line. A short wide copper track runs from pin 19 of the topmost socket to a plated through hole. Either cut this track with a sharp knife, or cut away the copper around the hole by hand, using a small sharp twist drill. This disconnects all the EPROM sockets from + 12 volts.
2) Similarly cut the longer wide track running from pin 21 of the topmost socket. This disconects the - 5 volt line.
3) Connect pin 21 of the top socket to + 5 volts, A convenient point to conect to is the wide track coming from the right hand edge of the board to within 1/2 inch of pin 24 of the top EPROM. The +5 volt lines can be identified by a short downward branch approximately 1/4 inch from the lefthand end.
4) A thin track approximately 1 inch long connects two plated through holes

2.7 inches from the bottom edge and 3.3 inches from the left edge of the board. Cut this track somewhere in the middle.

5) Connect the left hand plated through hole of the latter track to A12 at the edge connector – this is line 42 of Nasbus.

6) Connect the right hand plated through hole of the track to pin 19 of the bottom EPROM socket.

7) Select the addresses for the four sockets by connecting pad 5 to the appropriate two decode pads (remember, you've got 8K there now). The two 4K blocks selected need not be adjacent, but they must have different values for A12, because this line is used in the decoding. For example you could use blocks C000 and F000, but not C000 and E000.

8) Because the modifications have inverted the order of the decoding lines the addresses of the EPROM sockets are no longer in the correct order. Thus if you have selected addresses C000 – DFFF you will find that the start address are:-

IC27 C000 IC28 D000 IC29 C800IC30 D800

That completes the mods for the RAM A card. For just a little work, and using no extra chips, you get an extra 4K out of the board by replacing 2708s.

The Nascom 1 owner can also modify his main board to take 2716s in the monitor position; if you put a 2716 in each socket you can have two monitors on your machine, selected by a switch on the CS lines. Here are details of the modifications required.

1) On the soldered side of the board a wide track runs from pin 21 of IC38 to a plated through hole near the edge of the board. Either cut this track with a sharp knife, or break the plated through connection with a small sharp twist drill. This disconnects both EPROM sockets from the -5 volt line.

2) Also on the soldered side of the board, a wide track connects pin 19 of IC38 to the +12 volts lines. This track should be cut with a sharp knife near to pin 19.

3) Connect pins 21 and 24 of IC38 with a short wire link

4) Connect an insulated wire link from the solder pad of pin 19, IC38 to the solder pad of pin 2, IC36. Thus provides address line A10 for the 2716s.

5) Tie pins 4 and 5 of IC36 together. This gives a 2K decode instead of two separate 1k decodes.

6) Remove IC44 from its socket, bend pins 8, 11 and 13 to a horizontal position, and replace the IC.

7) Fit a single-pole double-throw switch at some convenient point in your computer. Link the centre of this switch to pin 8 of IC44. The two outer connectors of the switch are tied to +5 volts through 4k7 resistors; they are then linked to pads 8 and 11 of IC44.

You can now fit a 2K monitor in each of sockets IC38 and IC39. The switch selects which monitor is in use.

nascom

.... Computing for everyone

Nascom products, designed and built in Britain, are now backed by Lucas one of Britain's foremost industrial companies. This is a vindication of the innovative design of the Nascom computer and an assurance of its future.

NASCOM-1

Kit Price **£125** +VAT

Built Price **£140** +VAT

12" x 8" PCB carrying SLS I MOS packages 16 1K MOS memory packages and 33 TTL packages. There is on board interface for UHF or unmodulated video and cassette or teletype. The 4K memory block is assigned to the operating system and video display, leaving a 1K user RAM. Complete with keyboard.

NASCOM 2

Kit includes all parts to build CPU board which has resident 8K Microsoft BASIC and 2K NAS-SYS 1 monitor for machine code programming. Included with kit is a fully assembled LUCAN QWERTY SOLID STATE KEYBOARD specially designed to exploit the potential of the NAS-SYS monitor. Other interfaces include video to monitor or domestic TV, Kansas City standard cassette interface (300, 1200 baud) or RS232, 20mA teletype interface and two uncommitted parallel ports.

RAMBOARD

SERIES B card and gives user option of 16K-48K DYNAMIC RAM. This board can be arranged in page mode to allow use of up to 4 with NASCOM 2. Boards are fully buffered but PAGE MODE facility is an optional extra. This card can be used at 4MHz without wait state.

16K **£100** + VAT 32K **£115** + VAT 48K **£130** + VAT

I/O Board

This board provides support for six eight bit parallel ports, one serial port and a sophisticated counter / timer unit. Only the support circuits is provided, the main devices, from 0 to 6 in., are added separately.

£45.00 +VAT

TO BE INTRODUCED
Colour graphics board.

WATCH IT!!
Nascom products fully assembled and packaged in the near future!

Please send SAE for list of distributors.

Optional Packs for I/O Board

PIO Pack	Contains one MK3881 PIO	£12.00 +VAT
UART Pack	Contains one 6402, 8017 UART and the components for a crystal controlled V24/ RS232C interface	£16.00 +VAT
CTC	Contains one MK3882 Counter timer circuit	£14.00 +VAT

NASCOM IMP PLAIN PAPER PRINTER

The Nascom IMP (Impact Matrix Printer) features:
● 80 lines per minute ● 80 characters per line ● Bi-directional printing ● 10 line print buffer ● Automatic CR / LF ● 96 characters ASCII set (includes upper / lower case) ● 8 / 6 ● Accepts 8" paper (pressure feed) ● Accepts 9 ½" paper (tractor feed) ● Tractor / pressure feed ● Baud rate from 110 to 9600 ● External signal for optional synchronisation of baud rate ● Serial RS232 interface.

£325 +VAT

NASCOM FIRMWARE

CPU card can accommodate either 8K of static memory or 8270B EPROMS. This allows for inclusion of standard firmware on board.

ASSEMBLER Version 2.1 of ZEAP (280 Editor Assembler Package). A comprehensive line editor is provided in addition to an assembler operating in standard Z80 mnemonics. ZEAP can take advantage of special features of NAS-SYS, which was itself developed on this assembler. Supplied on tape **£30.00** plus VAT or in 4 x 2708 EPROMs at **£50.00** plus VAT.

DISASSEMBLER The NAS-DIS 3K disassembler reverses the effect of assemblers such as ZEAP by turning machine code into assembler program, automatically labelling and cross-referencing to produce a complete program listing. Supplied in 2 x 2708 EPROMs at **£37.50** plus VAT.

NASPEN
Naspen is a 2K word processor supplied in two EPROM's. It provides facilities for text entry, modification and is capable of finding specified words or characters in the text block and of operating a printer remotely. **£30.00** plus VAT.

NAS-SYS 3 THE NEW OPERATING SYSTEM FOR NASCOM 2.
Supplied in 1 x 2716 EPROM.
NAS-SYS 3 is the latest in the current series of Nascom monitors and includes features such as adjustable keyboard repeat and cursor speed, full interrupt handling and a number of powerful routines and commands making this probably the most comprehensive 2K monitor ever written for a microcomputer. **£25.00** plus VAT.

See us at the Personal Computer World Show Stand B28

New A moulded case for the Nascom computer

New Twin floppy disks

Lucas Logic

Nascom Microcomputers
Division of Lucas Logic Limited
Wetton Road Wedgnock Industrial Estate Warwick CV34 5PZ
Tel: 0926 497733 Telex: 312333

MONITOR.COM

By Chris Blackmore

THE SCENARIO

You have just added a disc drive or two and CP/M to your Nascom 2. After a few minutes gloating, you try the Digital Research editor, ED.COM) and find that it is a pig to use. Then you discover that the assembler provided only knows 8080 codes, instead of speaking Z80; even worse, it uses weird mnemonics designed to confuse anyone who is used to the Zilog set.

THE ANSWER

You could buy Diskpen and Macro-80 to solve your problem - if you have enough money left after lashing out on the discs. If you did, you still won't be able to run all your old programs. In this article I provide instructions that will enable you to produce a CP/M program that will pretend to be Nas-Sys 1. I also provide some machine code which enables the new version of Nas-Sys to read from and write to the disc.

To avoid possible problems with copyright, not to mention space in the magazine, the Nas-Sys code is not given. If you got your Nascom by legal means, you will find it in your manual.

THE METHOD

1) Initialise a disc, and copy DDT.COM onto it. This dynamic debugger is the closest CP/M comes to a monitor program like Nas-Sys; it can do some clever tricks, but is not exactly "user friendly".

2) Type DDT followed by Enter. This, of course, runs DDT.COM. In future I will put Enter so-and-so for anything that you have to type with Enter after it.

3) Clear some memory to work in, otherwise things can get very untidy, and you won't be sure if you typed the code you find when you come back from a tea break.
Enter F100, B00 ,0.

4) Type in the section of Nas-Sys 1 from 0108H to 07FFH in it's proper addresses in memory. Enter S108 to do this. Notice that it isn't as easy to type in hexadecimal code with this system as it is with Nas-Sys. Also notice that, where the listing should show all of the strings following EF, it only shows the first byte. Fortunately, the machine code in the listing does show what should be there.

5) Type in the start of Nas-Sys 1, normally found between 0000H and 0107H, at addresses 0800H to 0907H, using the S command instead of DDT again.

6) Amend the code you have just entered as follows, using the S command again

Address	Old Code	New Code
0800	31 00 10	00 00 00
015A	21 0A 08	21 0A F8
01F5	11 0A 08	11 0A F8
01FD	11 BA 0B	11 BA FB
021D	11 0A 08	11 0A F8
0220	21 4A 08	21 4A F8
022F	21 8A 0B	21 8A FB
078E	0A 03	15 09
0792	0A 03	06 0A
07A6	0A 03	3B 09
07B8	0A 03	67 0A

7) Enter S100 and insert C3 08 09, which is a jump past the much-modified version of Nas-Sys you have typed in to some new code.

8) Enter S108 and type in the following code:-

```
ED 73 03 01 31 00 10 CD 24 09 C3 00 00 31 00 10
CD 24 09 ED 7B 03 01 0E 00 C3 05 00 21 00 00 11
00 08 06 00 CD 31 09 06 08 1A 4E 77 79 12 23 13
10 F7 C9 CD B7 09 CD 24 09 11 5C 00 0E 13 CD 05
00 11 5C 00 0E 16 CD 05 00 FE FF CA 98 09 2A 0C
0C ED 5B 0E 0C B7 ED 52 19 D2 8B 09 11 80 00 01
80 00 ED B0 22 0C 0C 11 5C 00 0E 15 CD 05 00 FE
00 CA 56 09 FE FF CA 98 09 3A 7C 00 3D 32 7C 00
C3 6F 09 11 5C 00 0E 10 CD 05 00 CD 24 09 DF 5B
11 5C 00 0E 10 CD 05 00 CD 24 09 EF 44 69 72 65
63 74 6F 72 79 20 66 75 6C 6C 2E 0D 00 DF 5B CD
F1 09 EF 6E 61 6D 65 2E 0D 00 DF 63 21 5C 08 36
00 23 06 08 1A 77 23 13 10 FA E5 CD F1 09 EF 74
79 70 65 2E 0D 00 DF 63 E1 06 03 1A 77 23 13 10
FA 36 00 21 7C 08 36 00 C9 EF 50 6C 65 61 73 65
20 65 6E 74 65 72 20 66 69 6C 65 20 00 C9 CD B7
09 CD 24 09 11 5C 00 0E 0F CD 05 00 FE FF CA 52
0A 11 5C 00 0E 14 CD 05 00 21 80 00 ED 5B 0C 0C
01 80 00 ED B0 ED 53 0C 0C FE 00 CA 19 0A FE 01
CA 45 0A 3A 7C 00 3D 32 7C 00 C3 19 0A 11 5C 00
0E 10 CD 05 00 CD 24 09 DF 5B CD 24 09 EF 4E 6F
20 73 75 63 68 20 66 69 6C 65 2E 0D 00 DF 5B EF
43 6F 70 79 72 69 67 68 74 20 28 43 29 20 31 39
38 31 20 43 68 72 69 73 20 42 6C 61 63 6B 6D 6F
72 65 2E 0D 00 DF 5B 00 00 00 00 00 00 00 00 00
```

9) Enter GO to return to CP/M..

10) Enter SAVE 10 MONITOR.COM to put the program on disc.

THE OPERATION

To use the program enter MONITOR; the screen will clear and the Nas-Sys prompt

will appear. Then proceed as if using ordinary Nas-Sys, with the following commands added:-

Enter D to return control to CP/M (or press RESET).

Enter Pxxxx yyyy to Put a section of memory onto the disc, where xxxx is the start of the block and yyyy is the end of it.

Enter Fxxxx to Fetch a section of memory from disc, where xxxx is the start of the memory into which you want the data to be put.

THE LIMITATIONS

Because of the changes made to the monitor, programs that decide which monitor they are being run with by reading the first byte of the monitor will need to be modified. This includes ZEAP you will find.

When you are asked for a file name and type by the P and F commands, there is no check whether your input is sensible. File names in lower case in the directory are very difficult to get rid of, a fact you may or may not find useful.

Programs that write direct to the VDU RAM, or read from it, will need to be modified to take account of the change in the address of this RAM in CP/M systems. This does not apply to programs making use of the monitor's screen handling routines, of course.

THE COMMERCIALS

If you don't fancy all the typing that this project will need, send me a properly initialised disc (sysgen, etc.) and a pound for post, packing and not a lot of profit, and I'll save you the effort. The address is:-

Chris Blackmore,
31, Herne Rise,
Ilminster,
Somerset,
TA19 0HH

* + * + * + * + * + * + * + * + * + * + * + * + *

CP/M is a trade mark of Digital Research Ltd.

Nas-Sys is a trade mark of Nascom Microcomputers Ltd.

Chris Blackmore is Doctor Dark of the INMC magazine, and the author of the VORTEX graphics program.

*** - * - * - * - * - * - * - ***

MODIFICATIONS TO TINY BASIC FOR NAS-SYS
By Buxton

This article is intended to show how the Tiny Basic Interpreter –V2, published the Merseyside Nascom Users Group program book, can be changed to run under Nas-Sys 1/3, rather than under monitor T4 for which it was originally intended. In addition, the modification provide full screen editing of the source text. Assuming that a copy of the interpreter is in store, all that is required is to change the locations specified below to the new code, replace the old editor routine at £17C5 with the GETLN routine given in the full assembler listing, then save the new version of Tiny Basic on tape. Please note that old programs written for TBI-V2 will have to be altered, since the end of line marker is now £0D, rather than the £1F used by T4 and the earlier monitors. Other changes are:-

1) The command separator is now ':' rather than ';'
2) Format control in print now takes the form [5 if you wish the number to be printed in 5 spaces
3) 'CLEAR' is no longer active
4) The 'DUMP' command is replaced by 'SAVE'
5) Any number preceded by £ is now hexadecimal which makes POKEs very much easier. This may be used anywhere,
   e.g. FOR X = £C to £FF
6) When reading a tape you must first poke the ASCII code for the letter R, £52 or 82 decimal, into ARGX in the workspace (£C0B, 3083), or the tape will be verified, not Read
7) If you are using Nas-Sys 3 you must also poke a zero into location ARGN (£C2B, 3115), or the program may be loaded at the wrong location.
8) The input function X = IN (port no.) becomes X=CODE(£14AC)(port no.)

I hope that these notes and the list of modifications will prove useful to everyone who has asked for an updated version of TBI-V2.

Address	Old Code	New Code	Comments
1014	1E	0C	Clear screen
1019	56	53	Change V2 to S2
103C	1F	18	
103F	1F	0D	
1142	3B	3A	Command separator : not ;
1145	1F	0D	
114C	1F	0D	
114F	1F	0D	
1156	23	5B	[replaces £ for formatting
1175	1F	0D	
11C4	CD0C07	DF5200	Read tape
11D8	CD0004	DF5700	Write tape
11DE	Change to DF6421210C7E23C9		£ denotes hex. Not Clear
1343	1F	0D	
150E	3B	3A	Command separator : not ;
1517	1F	0D	
1529	1F	0D	
152D	1F	18	
1533	1F	0D	
1558	1F	18	

155E	1F	0D	
1581	1F	0D	
15F5	1F	18	
15FA	1F	0D	
1665	CD4A0C	F70000	CRT Routine
1669	1F	0D	
1678	1F	0D	
1691	CD4A0C	F70000	CRT Routine
169D	CD4A0C	F70000	CRT Routine
16D0	CD4A0C	F70000	CRT Routine
16DB	CD4A0C	F70000	CRT Routine
1705	44554D50	53415645	'DUMP' now 'SAVE'
175D	91DE	9341	CLEAR is no longer active
177D	494E94	2391DE	'IN' vector becomes '£'
	AC9440	944000	
17A8	Change to E5D5DF7BD1E1C9		Keyboard routine
17B3	CD4D0C	DF6200	Keyboard
17BA	CD4D0C	DF6200	Keyboard
17CE	CD4A0C	F70000	CRT routine
17D4	30FB	0000	Not needed if new GETLN used
17DD	1D	08	Backspace
17EA	1F	0D	
17FD	1F	0D	

NEW GETLN ROUTINE FOR TINY BASIC TBI-V2

17C5	F7	GETLN	RST 16	Print prompt char.
17C6	EF 18 00		PRT CCR	and CRLF
17C9	E5		PUSH HL	Save HL
17CA	AF		XOR A	Clear Accumulator
17CB	32 BF 0E		LD (BUFFER-1), A	Clear Marker
17CE	DF 63		SCAL INLIN	Get screen line
17D0	21 2F 00		LD HL, 47	
17D3	06 30		LD B, 48	
17D5	19		ADD HL, DE	Point to end of line
17D6	7E	SCN1	LD A, (HL)	Look for space
17T7	FE 20		CP £20	Space character
17D9	20 10		JR NZ, CHRFND	
17DB	2B		DEC HL	Move left across
17DC	10 F8		DJNZ SCN1	line but not off
17DE	21 C0 0E		LD HL, BUFFR	
17E1	36 0D	ELINE	LD (HL), £0D	Insert CR
17E3	23		INC HL	
17E4	23		INC HL	
17E5	36 FF		LD (HL), £FF	Insert terminator
17E7	2B		DEC HL	
17E8	EB		EX DE, HL	
17E9	E1		POP HL	Restore HL
17EA	C9		RET	
17EB	EB	CHRFND	EX DE, HL	
17EC	11 C0 0E		LD DE, BUFFR	Point at BUFFR
17EF	48		LD C, B	Get length in BC
17F0	06 00		LD B, 0	
17F2	ED B0		LDIR	Copy to BUFFR
17F4	EB		EX DE, HL	
17F5	18 EA		JR ELINE	

In addition, change £1042 from £3A to 00.

** .. ** .. ** .. ** .. ** .. **

USING PIXEL GRAPHICS FROM ASSEMBLER

By G. N. Evans

The Basic ROM on the Nascom contains plenty of useful routines for the assembler programmer. The problem is finding out where they are and, more importantly, how to drive them. I have discovered most of the useful routines, and I am using many of them in a Basic Compiler which I am currently writing.

Below I have given details on how to make use of the pixel graphics routines. Should you try to access a non-existent point, then Basic would take over, issue an error message, and then remain in Basic. To prevent this I have written a routine, CHECK, which tests the validity of the co-ordinates; the carry flag is set on return if they are not valid. The routine also converts, the y co-ordinates so that y = 0 corresponds to the bottom of the screen and y = 47 to the top, which is much more logical and convenient than Microsoft's ordering of the y axis.

Here is an example of the use of the routines to draw a vertical line through the centre of the screen:-

```
11 00 18      VERT    LD DE, £1800     POINT (24,0)
D5            NEXT    PUSH DE          Save current point
CD 1F 80              CALL CHECK       Legal co-ordinates?
38 0C                 JR C, ERROR      If not, jump
CD 00 80              CALL SETP        Set point
D1                    POP DE           Recover current point
1C                    INC E            Next point up screen
7B                    LD A, E          Transfer to accumulator
FE 30                 CP 48            Has whole line been set?
20 F0                 JR NZ, NEXT      Repeat if not
DF 5B                 SCAL £5B         Repeat to monitor, etc.
              ERROR   ...              Put error routine here
```

ROUTINES TO USE THE PIXEL GRAPHICS

```
                      ORG £8000

                      ADDRESSES IN BASIC ROM

              BCSPOS  EQU £FF31
              BSET    EQU £FF43
              BRESET  EQU £FF58
              BPOINT    £FFED
```

SET (D, E)

```
8000  CD1680  SETP   CALL CSPOS     Calculate screen posn.
8003  CD43FF         CALL BSET      Use Basic ROM
8006  C9             RET
```

All register corrupted (except alternate set)

RESET (D,E)

```
8007    CD1680      RESET     CALL CSPOS      Calculate screen posn.
800A    CD58FF                CALL BRESET     Use Basic ROM
800D    C9                    RET
```

All registers corrupted (except alternate set)

POINT (D,E)

```
800E                POINT     CALL CSPOS      Calculate screen posn.
8011    46                    LD B, (HL)      Pick up char. on screen
8012    CDEDFF                CALL BPOINT     Use Basic ROM
8015    C9                    RET
```

NZ if point set, Z if reset
All registers corrupted (except alternate set)

CALCULATE SCREEN POSITION

```
8016    6A          CSPOS     LD L, D         Low byte of x co-ord
8017    2600                  LD H, 0         High byte set to zero
8019    E5                    PUSH HL         Put on stack for ROM
801A    1600                  LD D, 0         Set high of y to zero
801C    C331FF                JP BCSPOS       Basic ROM does rest
```

On exit HL = Screen Address
A = Pixel information
All other registers corrupted

CHECK CO-ORDS ARE WITHIN RANGE AND CONVERT THE
Y CO-ORDS SO (0,0) IS AT THE BOTTOM LEFT CORNER

```
801F    7A          CHECK     LD A, D         X Co-ordinate
8020    FE60                  CP 96           Screen width
8022    3F                    CCF
8023    D8                    RET C           End if D > 95
8024    7B                    LD A, E         Y co-ordinate
8025    FE30                  CP 48           Screenheight
8027    3F                    CCF
8028    D8                    RET C           End if E > 47
8029    FE2D                  CP 45
802B    3005                  JR NC,CH2       Jump if 44 < E < 48
802D    3E2C                  LD A, 44
802F    93                    SUB E           Carry will not be set
8030    5F                    LD E, A         New y co-ord in E
8031    C9                    RET
8032    3E5C        CH2       LD A, 92        Sort out top line
8034    93                    SUB E           Carry will not be set
8035    5F                    LD E, A         Carry will not be set
8036    C9                    RET
```

On exit D and E contain new co-ords.
Carry set if out of range

.. ..

io systems ltd.

NASCOM GRAPHICS

VERY HIGH RESOLUTION FOR NASCOM 2

380 x 220 individually addressable points

FEATURES :

- fully bit mapped from dynamic RAM
- software controlled
- software supplied for point-plot, line-draw,
  -block-shading and display control
- mixed text and graphics
- real time plotting
- display size variable to suit memory available

Price ... £55 + 15% VAT (post free)

EPROM PROGRAMMER

FEATURES :

- programs: 3-rail : 2708, 2716
  and single rail : 2758, 2508
  2716, 2516
  2732, 2532

- EPROM type selected by plug-in modules - 3 modules
- supplied with simple wiring diagrams for all EPROM types
- driven from NASCOM 1 or 2 PIO
- powered from NASCOM and transformer (supplied)
- software supplied for READ / PROGRAM / VERIFY

****** CAN BE USED WITH OTHER MACHINES WITH 2 PARALLEL PORTS

Price ... £ 63 + 15% VAT (post free)

Both products built & fully tested supplied with comprehensive documentation and full instructions for simple installation

Send SAE for free data sheets

AVAILABLE NOW direct from:-

io systems ltd.
6 LALEHAM AVENUE, LONDON NW7 3HL.
Tel. 01-959 0106

HANDS (FURTHER) ON

By Viktor

USING THE CASSETTE INTERFACE

When data from a microcomputer is stored on tape the eight bits of each byte being stored are written sequentially onto the tape, usually with the addition of a number of bits denoting the start and end of a byte. In many micros this task is performed by software, but in the Nascom a device called a UART (Universal Asynchronous Receiver Transmitter) does all the work, and the processor merely sends each byte to be stored to the UART and then sits an waits while this chip writes the bit pattern through the cassette interface.

To be effective the method of storing the data bits has to distinguish clearly between a 1 and a 0. In the original cassette interface fitted to the Nascom, a 1 is stored as a pulse of 'noise' at a frequency of 1.95 kHz, while a 0 is simply represented by the absence of such a signal. The system worked fairly well at the low speed of operation which was originally chosen, but it is very susceptible to spurious signals, caused by poor tapes or by external 'noise' carried along the mains leads, which the interface then interprets as a 1.

A design for an improved cassette interface, which was much more reliable and which would operate at much higher data rates, was published in P.C.W. in December 1978, and this interface, always referred to as the 'Cottis-Blandford' after the designers, was adopted for the Nascom 2. It stores data in the CUTS format (Cassette Users Tape Standard, also known as Kansas City format), which represents a 1 by a 2400 Hz and a 0 by a 1200 Hz signal. The original Nascom 1 tapes are not compatible with CUTS tapes, but the Cottis-Blandford interface is available in kit form, and it can be fitted to a Nascom 1.

Two recording speeds are available on the standard Nascom 2, 300 Baud and 1200 Baud. The term Baud refers to the data transfer rate in bits/second, but don't forget that this includes the start and stop bits, so that at 1200 baud it takes about 9 seconds to transfer 1K of data. As noted in INMC 80, issue 1, it is possible to run Nascom 2 interface at 2400 Baud by linking TP20 to TP4 and TP21 to TP5. The various Baud rates are selected by switches 1 to 6 on LSW/2. The switch positions for the various rates are:-

U = up (nearest edge of board) D = down X = either

		Switch	1	2	3	4	5	6
300	Baud		D	D	D	D	D	D
1200	Baud		U	D	D	U	D	D
1200	Baud		X	U	U	X	U	U

At higher Baud rates you will have to use better quality tapes and a better tape recorder, because the date is being stored at a higher density, but it is well worth persevering, because once your interface is operating reliably at 2400 Baud you will find even 1200 to be slow, and 300 will seem interminable. By modifying the board it is even possible to operate at higher speeds – 4800 or 9600! However, the best baud rate for exchanging tapes with other Nascom users seems to be 1200.

Judging by the experience of our local club, it is not pssible to recommend any particular recorder. Some manage quite successfully with cheap, battery only recorders. However, I would suggest that you get as good a one as you can afford, preferably with tone and volume controls and a tape counter. Look also for one that, in addition to a standard DIN socket, has a separate speaker or earphone socket, as I have found that I get the best results by writing to the tape through the DIN socket but reading with TP7 on the Nascom 2 board connected to the external speaker socket. Also, take care to use screened leads to connect the Nascom to the tape recorder, and make sure that the screening is grounded by connecting it to TP8.

Having acquired a decent recorder you will have to find the settings for reading and writing by trial and error. Once you have found the optimum positions, and, if necessary, adjusted VR1 for the correct Baud rate, you should not need to alter them again. If subsequently you find you cannot read your own tapes, you should suspect dirt on the tape heads, a bad connection somewhere, or a poor quality tape. Reading other peoples tapes can be quite a problem; often their (or your!) tapes are misaligned, or the level they record at is wrong for your system. It is not uncommon for two Nascom owners who can both read their own tapes with complete reliability to be unable to load a single block of each others tapes. Of course, each thinks the other's tape system is to blame.

Well, that's enough on hardware, so back to Basic.

GETTING BASIC – Episode 2

Like most Basics, Nascom's Microsoft can be used in both direct and indirect modes. In indirect mode a line number is first typed, followed by one or more commands. The commands are not carried out as they are entered, but are stored internally to be executed later. The lines of commands are stored in numerical order, and it is normal to use line numbers which increase on steps of 10, so that lines can be inserted later to modify the program. The order in which you enter the lines is unimportant; when you LIST the program you will find that they have been sorted by the BASIC interpreter into numerical order. The ability to insert lines into a program is very important. A first attempt at writing a program always contains logical errors (well, my programs do!), usually called 'bugs', and these can be eliminated by inserting and/or deletion of lines, so remember to leaves gaps between your line numbers. Deletion of a line is simple – just type the line number and press enter; erasing the whole line from the screen (e.g., by typing 'escape') does not eliminate the line. It is still in the internal store, and when you list the program you will find it has not been removed.

In direct mode a command is typed in without a line number, and it is executed as soon as you press enter. The command is not stored internally, and it is therefore lost after is has been executed. For example, if you type PRINT 10*2 and then press Enter the computer will work out the value of 10 multiplied by 2 (Basic uses the asterisk for the multiplication sign) and print the answer. Many commands, for example LIST and LINES, are used exclusively or mainly in the direct mode, while

others, for example DEF, can only operate in indirect mode.

The command LIST instructs the computer, to display the lines it has stored. It will normally display 5 lines at a time, but this can be changed by entering the command LINES x, where x is the number you wish to display. The number x refers to stored Basic lines, not to screen lines; thus if any of the program lines contains more than 48 characters (see below), data may be scrolled off the screen even if the value of LINES is less than 15. The value of LINES will remain unchanged until a new value is entered, or until you do the next 'cold start' of Basic, when it will be reset to the default value of 5.

You can break out of a listing at any point by typing 'escape' (shift/enter). You can also start a listing at any particular line merely by adding the appropriate line number to the LIST command. For example, LIST 800 will list the program from line 800. The LIST command is particularly useful for editing purposes, because if you are using Nas-Sys, any line on the screen can be modified as required, and when you press enter the new version replaces the old, just as if you had re-entered the whole line (which you would have to do with most micros). However, if you break out of a program (with 'escape') and then LIST in order to edit the program, be sure to clear the screen first, because the interpreter will accept all the characters on a line, including any left behind by the program, and some very peculiar lines can be produced, particularly when there are graphics characters on the screen.

LINE FORMAT

The Nascom screen holds 16 lines of 48 characters. If you type in a Basic line which, including the line number and any spaces in the line, is exactly 48 characters long, as the last character is typed the cursor will move to the start of the next line. However, the line will not have been stored, because storage does not occur until the 'enter' key is pressed. You must return the cursor to some point on the original line (anywhere will do) and then press 'enter'.

In fact, Nascom Basic will accept up to 72 characters on a line, but the power and flexibility of Nas-Sys screen editing is lost once you exceed 48. One method of entering more than 48 characters per line is to enter the one-byte graphics codes which represent the Basic reserved words. For example, pressing the space bar with the graphics key depressed give code £A0, which represents the command LIST. The PRINT command is a special case – this can be entered by typing a question mark (the reserved word is not, of course, stored as a question mark, the code is actually £9E). Try entering a line number, followed by a series of question marks separated by colons. Enter the line and list the program – you will see that the interpreter produces a series of PRINT commands. However, you will find that any characters that overflow the first line will be lost when you try to edit this monster. If you look in the computer's memory by entering T10F8 1110 from the monitor you will see a series of £9E codes, separated by £3As, which is how the Basic interpreter stores the line.

Another method of entering longer lines is to use the monitor's X command, which controls the serial output. Return to the monitor by pressing reset, enter X0, and

return to Basic with Z. Notice that the rate at which characters are output, for example when you LIST a program, is now dependent on the baud rate set for the cassette interface, because data is sent to the serial port as well as the screen. You will find that you can now enter up to 72 characters in each Basic line. However, editing facilities are severely restricted, and I would personally recommend that you stick to a maximum of 48 characters per line, because the slight increase in memory used is more than compensated for the ease of editing produced.

AND NOW FOR THE FUN!!

```
10 CLS:Z$ = “PROGRAM TITLE”:FOR I = 1 TO LEN(Z$)
STEP 1
20 POKE 3017 + I +((48 – LEN(Z$))/2),ASC
(MID$(Z$,I,1)
30 NEXT
```

The above bit of code, or something similar, is often found at the beginning of Basic programs. It writes the program title in the centre of the top line of the screen; this roundabout method has to be used because Basic PRINT command will not write to the top line if the system monitor is Nas-Sys 1. So what is happening? It is best to break the program down into its individual commands and functions, and study each in more detail. Apart from the CLS command, which merely uses a monitor to clear the screen, these are, in order in which I shall look at them:-

POKE
LEN(Z$)
ASC(Z$)
MID$(Z$,I,1)
FOR . . TO . . STEP . . NEXT

POKE(X,Y)

This command directly modifies memory location X, changing its contents to Y. In the previous article we changed screen locations £09E3 from a space (£20) to a ‘bell’ (£07) using the monitor. The same result in Basic is produced by entering POKE 2531,7. Here basic, as always, uses the decimal value for the memory location and the number being inserted (9x256 + 14x16+3=2531). Thus to insert the letter A in this position you would enter POKE 2531,65 (A is represented by the hexadecimal code £41, and 4x16 + 1 = 65). Of course, you can use POKE to modify any memory location, not just the screen RAM. For example, many Basic programs use sections of machine code either for increased speed of operation , or to do something which cannot be done by Basic. The code is often stored in the Basic program as a list of decimal numbers which are then POKEd into a suitable space. The two-byte start address of this machine code is then stored in locations 4100 and 4101, so that the Basic interpreter knows where to find it when needed. This can then be done by two POKE instructions, but it is more usual to use the DOKE instruction, which I shall seal with in the next article. However, you must be very careful when

using POKE instructions, because if you get the numbers wrong you may corrupt the Basic program, change the value of a variable, or affect the operation of the monitor. For example, try typing POKE 3111,1/Enter. You will find that your keyboard is now in 'typewriter' mode, and you will have to use the shift key to type upper case letter. To change back, enter POKE 3111,0 (Basic will accept commands in lower case). Similarly, POKE 3111,4 will change the keyboard to 'graphics' mode.

I will now sneak a quick look at the PEEK command (although it isn't in the above list), because it is complementary to POKE. PEEK(X) tells you the value stored at memory location X. Thus PRINT PEEK(2531) will print the (decimal) value stored at the screen centre. This also illustrates the difference between Basic commands and Basic functions. A command tells the machine to do something; a function asks it a question, but you have to use a command to do something with the answer. Thus entering PEEK (2531) has no obvious effect, you must precede it with a PRINT command for the answer to appear on screen.

STRINGS

A string is simply a series of characters. In the program above the string variable, Z$, is first defined by equating it to the expression between quotation marks, PROGRAM TITLE, so that when the program subsequently meets Z$ it knows that it refers to this expression.

LEN(Z$)

This function gives the length of Z$, i.e., the number of characters, including spaces, in the string. Thus if you enter PRINT LEN ("THIS IS A STRING") the computer will produce the answer 16. The same result is obtained with PRINT LEN(Z$) is Z$ has previously been defined as "THIS IS A STRING".

ASC(X$)

This returns the decimal value of the ASCII code of the first character of string X$. Thus ASC("FRED") has the value 70, because the ASCII code for F is £46 and 4x16 + 6 = 70.

MID$(X$,I,J)

You often need to be able to access specific groups of letters in a string, and the most versatile of the basic string handling functions for this purpose is MID$(...). It extracts J consecutive letters from the middle of a string, starting at the Ith letter. Note that this function returns a string, not the ASCII equivalent of a string, so if you type PRINT MID$("MISS PIGGY",6,3) the computer will print PIG.

FOR . . TO . . STEP . . NEXT

Since the first thing that most new computer owners do to try out Basic is type:-

```
10 FOR A = 1 TO 50:PRINT "HELLO":NEXT
```

FOR . . NEXT loops will presumably be familiar to everyone. STEP can be omitted if you want to increment in steps of one; it has only been included in the program

example above for completeness. When writing programs FOR . . NEXT loops be sure not to change the value of the loop variable within the loop, or you may find the program gets ‘hung up’ in the loop. When the program exits from a loop the loop variable is one STEP greater then the value set by the TO limit. Try entering:-

FOR A = 1 to 50:NEXT:PRINT A

when you will receive the answer 51.

Right, we can now analyse the operation of the title printing program. Line 10 clears the screen, defines Z$ as PROGRAM TITLE, and sets up a loop to scan this string letter by letter. Line 30 is merely the NEXT part of this loop. However, line 20 looks quite complex, and certainly refutes claims that Basic programs bear any relationship to simple English.

In fact the line consists of a single command of the form POKE X,Y, but here both X and Y are complex expressions. The memory location into which data is being inserted is 3017 + I + (48 – LEN(Z$))/2. Now 3017 is one less than the decimal address of the start of the top line on the Nascom screen, so as I is increased from 1 in the loop 3017 + 1 steps through the memory locations at the start of this line. To this value an offset (48 – LEN(Z$))/2 is added. This merely deducts the string length from the line length and halves the result, to ensure that the title is centred. If the number of characters in the string is odd the offset will not be an integer, but the POKE command used only the integral part of the expression, try entering POKE 2531.7,7.3. Thus as the program cycles round the loop, data is inserted into the central LEN(Z$) locations of the top line. The value inserted at the Ith point is ASC(MID$(Z$,I,1)), that is, the ASCII code corresponding to the Ith character in the string.

So one by one the letters of the title appear at the top of the screen. Easy, isn’t it? Well, perhaps it is too easy that I can be accused of making a simple subject complicated, but the point of analysing this short program in such detail is that what you learn can be carried over to help you write your next program.

Finally, here is a simple program to print out the complete ASCII and graphics set, with a space between each character:

10 CLS:FOR A = 1 TO 255:PRINT CHR$(A);:NEXT

In the next article I shall cover further Basic commands, including the double PEEK and POKE commands DEEK and DOKE, and examine the syntax and use of the PRINT command.

* ** ** ** ** ** ** ** *

COMMAND INPUT

While waiting for you to enter a command the Nascom is in a routine called INLIN, sitting in a loop in which it scans the keyboard and the serial input, prints any character received, and continues until it a NEWLINE (£0D) is obtained. It then sets the DE register pair to the address of the start of the line that the cursor was on when the NEWLINE was received, and returns from INLIN to the main monitor routine, which is called PARSE. Here it tests the character at the start of the line entered; if this character is a space it looks in the workspace at address £0C2B, which is a location called ARGX, to discover what the previous command entered was. Normally a line with a space at the start is ignored, that is, the monitor recalls the INLIN routine and waits for further input; however, if the previous command was S (single step), the monitor behaves as if this command had been re-entered, so that you can continue to single step through a program merely by pressing NEWLINE.

If the line does not start with a space the first character is tested to find if it is an upper case letter; for any character outside the range A – Z an error message is produced, and INLIN is then recalled. When an upper case letter is found this is stored at ARGX and also at £0C0A (ARGC), and the monitor proceeds to read any hexadecimal values entered on the line, storing them in the workspace in ten locations, ARG1 – ARG10. the number of values found is stored at £0C0B (ARGN). If any of the values found are invalid (contain illegal characters, or too many character), or if more than 10 values are entered, once again an error message is produced and INLIN is recalled.

If all the tests are passed, the first three hexadecimal values on the input line are loaded into HL, DE and BC, HL is saved on the stack, and the location of the subroutine address is calculated by adding twice the value of the ASCII code for the command letter to the command table base address - £0706 for Nas-Sys 1, £700 for Nas-Sys 3. Thus for command A (ASCII £41) the routine address is stored in the two bytes £0788, £0789 for Nas-Sys 1, and at £0782, £0783 for Nas-Sys 3. As usual, the low byte of the address is stored in the first location, the high byte in the second. The routine address is transferred to HL, and the HL register pair is the exchanged with the top two bytes of the stack. This restores ARG1 to HL, and leaves the address of the routine to be executed on the stack. The program then performs a RET (return from subroutine) instruction, which sets the program counter to a value POPed from the top of the stack; the net result is that the machine enters the command routine with any hexadecimal values specified in the workspace , and the first three values in HL, DE and BC.

You should note that on RESET the region of the workspace ARG1 – ARG10 is set to zero. When you enter a command with a given number of arguments only these argument locations are changed; the remainder are unaltered. Thus if you

enter Wxxxx yyyy to write the memory from xxxx to yyyy to tape, you can obtain a second copy of the same region merely by entering W, because the original parameters are still in the workspace, providing that you haven't pressed RESET. Similarly, if you set the number of lines to be displayed in the tabulate command, this value is retained until re-entered, or set to zero by RESET.

THE INDIVIDUAL COMMANDS

ARITHMETIC Axxxx yyyy

This command gives the sum and difference of the two sixteen-bit values, together with the 'offset' for use in relative jump instructions. For example, to perform a relative jump from £0C80 to an instruction at £0CA0 the offset necessary is given by entering AC80 CA0, when the monitor will respond with 1920 0020 1E. Thus the necessary relative jump is:-

£0C80 18 1E JR £0CA0

Note that the address of the relative jump instruction is entered, not the address of the offset byte. If the offset lies outside the permissible range for relative jumps two question marks are printed in place of the offset.

BREAKPOINT Bxxxx

The use of the breakpoint command, Bxxxx, to insert restart £E7 at address xxxx for the debugging of programs was covered in the first article in this series. The breakpoint function is turned off by entering B0. Nas-Sys 1 assumes that there is ROM at address £0000, so it goes ahead and 'inserts' £E7 at this address; of course, this has no effect because you cannot change the ROM by writing to it, so the breakpoint is inoperative. In Nas-Sys 3 no attempt is made to insert £E7, or to replace the original code at the breakpoint, if the breakpoint address is zero. This means the breakpoint routine can be used if you have RAM at address 0000, i.e., in disc systems.

COPY Cxxxx yyyy zzzz

The command copies a block of data zzzz bytes long from address xxxx to address yyyy. For general copying of data it is best to use the intelligent copy command, I, which thinks about what it is doing to ensure that data is not overwritten (see below). In fact the C command is most useful for filling a block of memory with a specific byte. Thus if you modify £1000 to £00, and then execute C1000 1001 400 you will fill the 1K block starting at £1000 with zeros. Because I never use the Copy routine I have replaced it with a command to compare two blocks of memory, for which a listing was given in the first article in the series.

DJUMP D

In Nas-Sys 3 command D causes a jump to £D000, which is the start address of the ROM version of Zeap 2. Of course, any software which runs £D000, in ROM or RAM, for example a printer initialisation routine, can be accessed by the D command. The Nas-Sys 1 command table contains the address of the error message for command D (location £078E). Of course, with an Eprom Programmer it is easy to change this address to £D000, or to any other useful address, such as the start of a disassembler (£C400 for Nas-Dis).

EXECUTE Exxxx

In many ways this is the most important command, since you use it to run all machine code programs. The first thing the execute command routine does is to set the value stored at location £0C26, known as CONFLG, to -1. It then throws away its address by POPing it off the stack into the AF register pair. Consequently any program which is to be accessed with an E command cannot return to the monitor with a simple RET instruction, whereas a program accessed by some other command letter, such as the Compare routine mentioned above, can end in such a way.

The E routine next checks the number of hexadecimal values in the input line, ARGN; if it is not zero, HL must contain the specified start address of the program to be executed, and this is stored in the workspace at £0C69. If no address has been entered, execution will continue from the last address stored here. The registers BC, DE, HL, AF and SP are then loaded from the Register save area in the workspace (£0C61 - £0C6C) and the execution address is put at the top of the stack. When a program is interrupted at a breakpoint the current registers are saved in the Register Save Area. When the program is continued fro the breakpoint the above manipulations ensure that all the registers are restored to the values they had before the break. However, they have the result that when a program is entered by the E command the contents of HL, DE and BC bear no relationship to the first three values in the input line; they either contain the values stored at the last breakpoint, or are set to zero if there has been a RESET since the last break. Of course, the values are retained in the workspace at ARG1, ARG2 and ARG3 and may be recovered if needed.

The contents of AF are pushed onto the stack and bit 3 of port 0 is set; this activates a circuit which sends a non-maskable interrupt to the processor after 4 M1 cycles. After recovering AF the routine comes to a return-from NMI instruction, RETN. Because the specified start address is at the top of the stack, this caused a jump to the routine to be executed. Three M1 cycles have now occurred, therefore as soon as the first instruction of this routine has started the NMI line is activated. Consequently, the processor is interrupted at the end of the first instruction, and this causes a jump to

the NMI handling routine. Here bit 3 of port 0 is reset, and CONFLG is tested to see if the program has arrived at this point from an E command (in which case CONFLG is not zero), or from some other source, such as a single step command, a breakpoint, or a hardware NMI. If it came an E command CONFLG is now set to zero, so that a subsequent NMI is handled correctly, if a breakpoint has been entered Restart 32 (£E7) is inserted, and the NMI handling routine jumps back to the executed program with a RETN. At last the program you started with Exxxx is running. You will see what happens if CONFLG is zero when we discuss the single step command.

F (Error)

F is not a valid command letter in either Nas-Sys 1 or Nas-Sys 3, and an error message is produced when it is entered. I use the command for a Find routine, which searches from a specified address for a string of up to 9 bytes.

```
2B          FIND     DEC HL          HL holds start address
3A 0B 0C             LD A, (ARGN)    Get number of values
FE 02                CP 2            Must be at least 2
30 03                JR NC, VALID    If so, continue
DF 6B                SCAL ERRM       iF not, print Error
C9                   RET             and end routine
3D          VALID    DEC A           Get string length
4F                   LD C, A         Save string length in C
41          SRCHLP   LD B, C         Transfer to B
11 0E 0C             LD DE, £0C0E    Point DE to first byte
1A                   LD A, (DE)      Get first byte
23          SRCH1    INC HL          Look for first byte
BE                   CP (HL)         Found it?
20 FC                JR NZ, SRCH1    If not, continue to look
05                   DEC B           Only on byte to find
28 09                JR Z, CALLDS    If so, go to display
13          SRCH2    INC DE          If not, look for second
13                   INC DE
1A                   LD A, (DE)      Get next in string
23                   INC HL          Go to next memory byte
BE                   CP (HL)         Are they the same?
20 ED                JR NZ, SRCHLP   If not, start again
10 F7                DJNZ SRCH2      Look for rest of string
D7 08                RCAL  TAB       All found, so tabulate
CF                   RST 8           Wait for keypress
FE 1B                CP “ESCAPE”     Is it an escape?
20 E4                JR NZ, SRCHLP   Look for next string
DF 6A                SCAL CRLF       Output, CR
C9                   RET             End routine
```

* * TAB routine starts here * *

The TAB routine used by the above program is the same as the one in the comparison routine, and the code for this should be entered at the position marked above. Of course, if you use both the compare and the Find routine,

you would only need to enter the subroutine once, making the necessary adjustments to the subroutine call instructions.

The command is entered as Faaaa xx yy zz . . ., where aaaa is the start address for the search, and xx, yy, zz . . ., where aaaa is the start address for the search, and xx, yy, zz, . . . is the string of bytes (up to 9 bytes long). However, the above program was written for use with Nas-Sys 1; when used with Nas-Sys 3 strings of three or more bytes can have strange effects on the tabulate command, because the extra arguments affect the formatting of the display. The routine is relocatable, that is, it may be used at any memory locations. The start address should be stored at £0792 for NAs-Sys 1; if you must use it with Nas-Sys 3, the appropriate location is £078C.

GENERATE Gxxxx yyyy zzzz

The command produces a tape copy of a program between addresses xxxx and yyyy, which loads automatically and then self executes at address zzzz. The G routine first resets the output table to send data to both the CRT and the serial port., and outputs the following characters: Newline, E, 0, Newline, R, Newline. A delay is inserted between each character. It then calls the write routine to send data from xxxx to yyyy to the tape. Finally, it outputs and E, the start address, zzzz, and a Newline. A tape produced by the Generate routine is played directly into the machine with the monitor in the INLIN loop. Because this loop scans both the keyboard and the serial input the data received is treated as if it had been entered from the keyboard. Thus the first Newline sets the cursor to the beginning of a line, E0/Newline resets the monitor, and R/Newline, executes the read command. The machine then reads in the data stored by the write command, and finally Ezzzz/Newline starts the program at address zzzz.

The G command suffers from two drawbacks; firstly, the tape LED is not turned on until the Write or Read commands are invoked, which makes it inconvenient in systems which use the tape LED to control the tape recorder motor; secondly, if the program is loaded incorrectly the command still goes ahead and runs it, which can have unfortunate results.

HALF DUPLEX TERMINAL H

When the H command is executed, the monitor is put in a loop in which the INLIN routine is continually scanned. However, when a Newline is received and the INLIN routine is left the input line is not dissected by the PARSE routine, but INLIN is recalled. Consequently commands are not accepted, and the only way to escape from this command is to press RESET. If your Nascom is connected to a printer, this command will enable you to use the system as a typewriter, suppressing the machines normal response to valid commands and error messages on invalid input. Of course, any commands that you need to give, such as Kn to set the keyboard options, U to activate the printer routine, or Xn to control external input /output, will have to be entered before the H command.

INTELLIGENT COPY Ixxxx yyyy zzzz

This command is used to copy a block od data zzzz bytes long from xxxx to yyyy. The routine first subtracts yyyy from xxxx; if xxxx is greater the normal copy command is used, as data cannot be overwritten. If yyyy is greater, the pointers to the two data blocks (data written from, data written to) are reset to the top ends by adding the length of the blocks less one, and the data is then copied from the top down; this ensures that data is not overwritten.

BASIC COLD START J

Command J causes a jump to £FFFA, which is the 'cold start' address for the ROM version of Nascoms Microsoft Basic. The cold start initialises the Basic workspace; as part of this process it uses a small section of memory in the Basic Interpreters text area, with the result that a cold start not only 'kills' any program already entered, but also corrupts two or three lines near the start of such a program, which makes recovery of a program after a cold start a much more fiddly process.

KEYBOARD Kxx

When this command is executed the low byte of the first hex value on the input line, which is held in the L register, is stored at a location called $KOPT (£0C27). When a character is input from the keyboard it is first converted to ASCII, and the value at £0C27 is then tested. If this value is zero, as after a RESET, the keyboard character remains unchanged. If bit 0 of (£0C27) is set and the character is a letter, bit 6 of the character is inverted. This reverses the effect of the shift key for letters, i.e., it changes upper case to lower case, and vice versa; the keyboard now operates in the typewriter mode. If bit 2 of (£0C27) is set, bit 7 of the character is inverted; this reverses the effect of the graphics key. The value stored at £0C27 is retained until changed by a subsequent K command, or returned to zero by RESET.

In the next article I hope to conclude the the discussion of the commands, and give a listing of an improved Read routine, which does not load data if the checksum is incorrect

* > < * > < * > < * > < * > < * > < * > < *

I have just received the latest copy of Mid-Sussex Micro Club News. It's a club for all micros, but Nascom gets more than it's fair share of attention in the newsletter. As they have offered use the use of any of their material I hope to give you a sample of their output in the near future.

The Amateur Computer Club of North Staffordshire mets on the third Wednesday of each month. Meetings are usually held in the Talbot Hotel, Station Road, and start at 7.30 pm. The meetings a fairly informal, with usually 12 – 20 people present. It is a general club and the members have a wide range of machines – Nascom, Tandy, Apple, Video Genie, etc. Anyone is welcome to the meeting, whether they own a computer or not. For more information please contact the Chairman, Michael Turner, on Stoke-on-Trent 324639 in the evenings, or write to him at 542, Lightwood Road, Lightwood, Stoke-on-Trent ST3 7EH.

Will any Nascom owners in New Zealand please contact M.A.Fox, 53 Hellyers Street, Birkdale, Aukland 10. Mr. Fox is a committee member of the New Zealand Micro-Computer Club, and he wishes to form a Nascom Users Group.

Here is a letter from the Lincoln group; the information should have been included in last months list, so as a penance we will print it in full.

Dear Editor,

May I wish you all success with the new magazine. Most of the currently available crop of mags. Seem to have disappeared under a morass of waffle about the socio/politico/business implications of micros. Hard factual articles and good software tipsare virtually non-existent.

Anyone wishing to contact a group of Nascom 1 and 2 owning enthusiasts in the Lincoln are can contact me at the address below (or by phone). We have 8 machines between us, including a disc based N1 and an Epsom equipped N2. We have both "meddlers" (harware) and "thinkers" (software), with a reasonable degree of expertise. We usually meet on Wednesday evenings, but weekend sessions are not unheard of. We are always glad to meet new enthusiasts, even when they are just passing through the area.

Suggestions for the magazine? What about a cheap Modem design – the we can have a national NASCOM-NET! Similarly a graphics tablet (with hi-res graphics) – has anyone designed one yet?

Best Wishes

John Clifford
448, Newark Road,
Lincoln LN6 8RX
Tel. Lincoln 21607

FOR SALE:-
RAM B board with 48K, no write protect circuitry, excellent condition. £100 or offers. Tel. Stoke (0782) 324639 evenings

PROGRAM POWER

NASCOM 1 & 2

LUNAR LANDER SUPREME (16K/B/G) – Classic spacecraft landing simulation. Short, medium and long range scans show planet surface in varying detail
Continuously updated STATUS REPORT gives vertical, horizontal & relative velocity, altitude, fuel level, G factor & surface scan for suitable landing site & skill selections
Brilliant graphics £9.95

STARTREK II (32K, B, G) enthralling, real time version from our Invasion Earth author, using M/C code sub-routines to great effect. Special features include larger galaxy, shielded homing warheads (fired by Klingons). time slots & non stop action £9.95

INVASION EARTH (MC/G) – New improved version! 4 complexity ratings 10 overall speeds Variable shot speeds & alien descent rate 4 invader types Intelligent homing exploding, angled, direct, multiple warhead & radio jamming missiles £8.95

INVASION EARTH (MC) as above with SOUND EFFECTS using AY-3-8910 CHIP £10.95

"NASCOUNT" PERSONAL FINANCE (16K/MC) Make life simpler with this finance planner Budget income/expense month by month and highlight likely surpluses & deficits Can be used to check bank account & record past income/expenses 50 entries each period Five digit codes with analysis by code & sub code Calculate cumulative cash flow to specified month end Output to cassette & printer £9.95

CONSTELLATION (16K/B) Turn your screen into a telescope & view the stars from any point in the Northern Hemisphere at any time & date Display stars by magnitude, identifying number or constellation The telescope can be raised & lowered, zoomed in & out Also output of star map to printer £6.95

** NASCOM 1 Cottis Blandford cassette interface for N2 format, reliability & fast load £14.90
B - Nascom BASIC (State Tape BASIC if required)
MC – Machine Code G – Nascom Graphics 8K RAM required unless otherwise stated
ALL PROGRAMS SUPPLIED ON CASSETTE IN CUTS & KANSAS CITY FORMAT

WORDEASE WORD PROCESSOR (MC)
Professionally written 4K word processor 14 line window on text buffer & extensive on screen editing facilities insert & delete characters lines and paragraphs Text manipulation copy from one section of text to another or read in additional material from tape to any point in the text FIND & REPLACE facility Text buffer size according to available memory Exceptional formatting capability commands embedded in text allow complete flexibility e.g. variable tab position indent, line length & page length Use of up to 10 'MACROS' permits automatic inclusion of headings, footings & other 'text repeats', & also automatic page numbering.
Output to printer can vary character delay, inhibit line feeds & force upper case if required.
An extensive manual is supplied (itself prepared on Wordease) (MANUAL ONLY) £1/refundable against program order **£25.00**

VORTEX (MC) (State 16, 32 or 48K) Speed up your display of pixel graphics. Cassette holds 29 separate routines to be called from BASIC Extensive instructions and examples supplied Give your programs that professional touch! £8.95

CLUB MEMBERSHIP (32K/B) Create a file of 200 Members containing Name, Address, Date of Joining, Number, Remarks, Paid or not Amend or Delete Comprehensive search & sort routines Partial or complete listings Output to cassette & printer. £9.95

Super Startrek (16K/B)	£6.95
Alien Labyrinth (16K/B/G)	£6.95
Super LIFE (MC/G)	£6.95
Cliff Invasion (B/G)	£6.95
Space Fighter (B/G)	£5.95
Cowboy Shoot out (MC/G/Sound)	£4.95
Fruit Machine (B/G)	£4.95
Road Race (MC/G/Sound)	£4.95
Labyrinth (B/G)	£4.95
Renumber (MC)	£4.95
WIRRAL PILOT V4 G	£12.00

CLIFF INVASION (B/G) the aliens have landed in droves You have one remaining laser base Your only chance shoot the ground from under them as they descend the cliffs towards you Landslides created. Errors in direction & elevation of shots are costly 3 levels of skill Like all aliens they breed like rabbits! **£6.95**

MUSIC BOX

Now you can make music with NASCOM Easy to follow program allows you to key in old favourites or have fun composing your own tunes 7 octave range with staccato option 9 tempos Set note duration or tap in rhythm as required
Comprehensive editing Delete, insert or amend notes
Single step forward & backwards through tune Add new lines within declared array size.
The program includes tape generating and play back routines & is supplied with 2 demonstration melodies & instructions for connecting your Nascom to an amplifier/speaker such as our unit below
Min 16K required – please state T4 or Nas-Sys/2 or 4 MHz
Only ~~£9.95~~ ***7.95***

AUDIO INTERFACE BOARD/SPEAKER

Compact & ready assembled, suitable for use with 'MUSIC BOX' & other 'sound effects' programs. 3 sample connections Complete with instructions on programming for sounds £10.75

AY-3-8910 SOUND CHIP

Program up to 3 independent channels with music & sound effects! Supplied with detailed write-up ~~£8.50~~ ***6.45***
SOUND CHIP INTERFACE BOARD – Using the PIO, program up to four sound chips at once ie 12 separate programmable sounds Each board contains an interface allowing a further board to be attached. Only simple link changes required. Connect to an amplifier/speaker such as our unit above £13.00
SOUND CHIP DEMO PROGRAM – First mode gives direct entry to chip registers, making experimentation simple & thus rapid appreciation of chip's potential Second mode turns keyboard into 7 octave piano displaying state of registers & notes (up to 3) being played £5.95
60 page Data Manual (no VAT) £2.25

BOARD GAMES

Games Graphics ROM/Adaptor	£18.00
SARGON CHESS – Book (no VAT)	£9.50
with program	£19.50
Book/Program/ROM/Adaptor	£35.00
Draughts (B/G) *	£7.95
Backgammon (16K/B/G) *	£7.95
(*state ORD or ROM version)	

Please add 55p/order P & P + **V.A.T. @ 15%**
Sae for FULL CATALOGUE (Now over 50 items!)
PROGRAM POWER
5, Wensley Road,
Leeds LS7 2LX
Telephone (0532) 683186

VOCABULARLY TUTOR (B)
French & German – learn vocab. the easy way. Translation tests in both directions. Results of tests given incl. Results of answers. Also Quick 'self-test' option to save keying in. New vocabulary can be easily added. Especially useful for new students

£5.95

PROMPT (B)
Devised to take the strain out of learning text for acting, after dinner speeches etc. this program on a line by line system gives you either a word by word 'prompt' or a letter by letter in each word 'prompt'. The fewer 'prompts' required the greater you score. Excellent value at

£5.95

PROGRAM POWER

PRESENTS
"THE KEYS OF KRAAL"
AN ADVENTURE PROGRAM for NASCOM

Legend has it that KRAAL – known by the Bedouin as the 'Temple of the Undead' – houses a fabulous treasure and the four Locks of Eternity. It is believed that anyone who finds the key to one of the locks will break the curse of Kraal, release the souls of lost adventurers and escape with treasure of untold proportions. No-one has yet lived to test this theory

The temple is inhabited by Monsters and Magical Beings. Your sword and arrows may be sufficient to destroy Gargoyles, Minotaurs, Mummies & The Cyclops etc., but you will need the various spells you find to combat JUBILEX, ASMODEUS, GERYON and other magical beings. Beware also the Vampire Bats who will sap your strength requiring you to find a life-giving Elixor, and the SPIDER GODS whose attentions are usually fatal !

.....

The program required 24K RAM and is exceptionally well presented. Nine chambers are depicted at one time with Monsters & Demons continually moving within their cells, and making 'real-time' attacks. Swords flash, arrows fly & spells home-in on the victim! Each game is played against the clock & can be saved on tape after generating it – play it again and again (Nascom Basic/Graphics)

Send NOW for this excellent program. Only £8.95

to **PROGRAM POWER**
5, Wensley Road,
Leeds LS7 2LX
Tel (0532) 683186
or send SAE for full Catalogue

Please add V.A.T. @ 15%
+ 55p/order P & P
SEND FULL DETAILS OF YOUR NASCOM