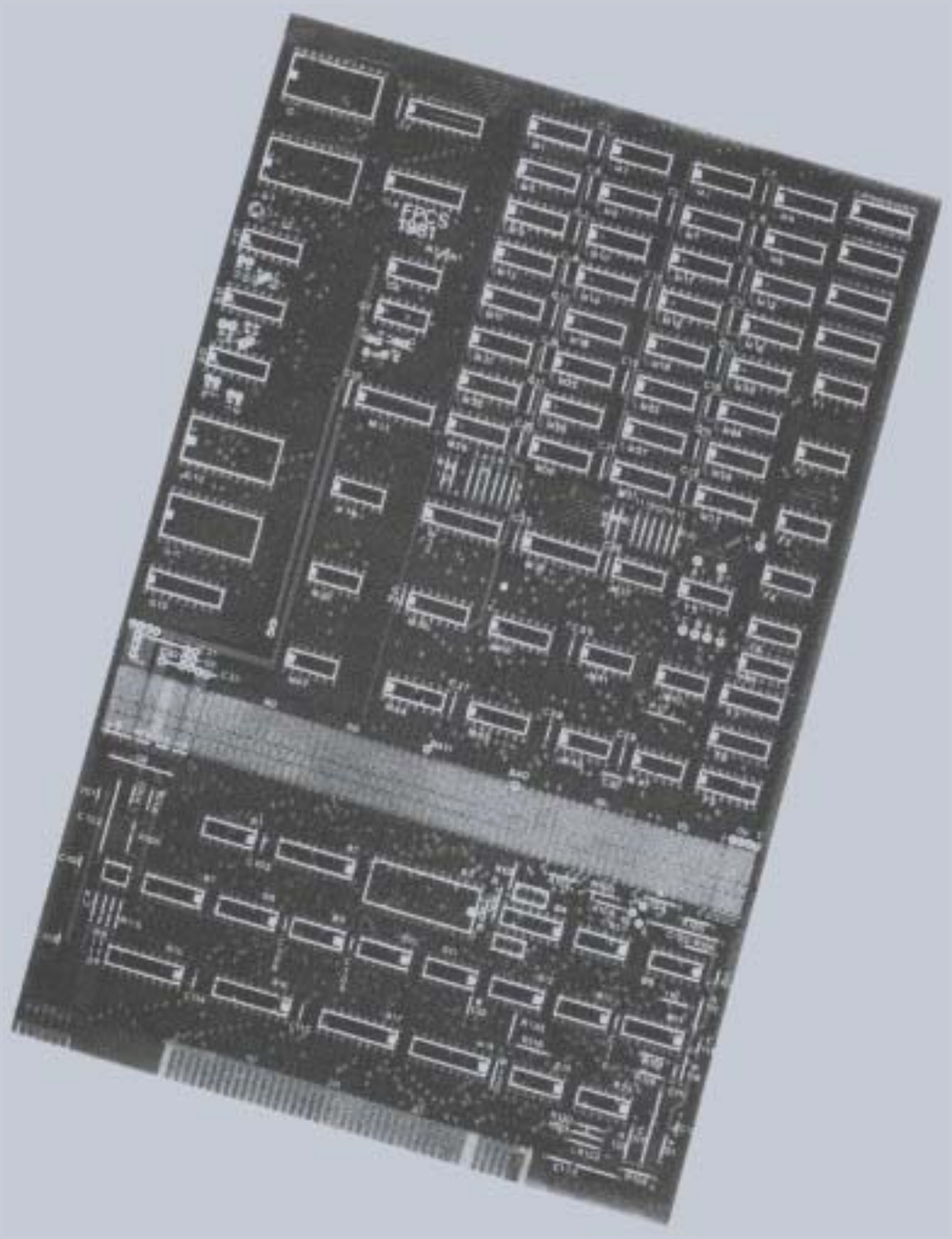
μP MICROPOWER VOLUME 2, NUMBER 2

A MAGAZINE FOR NASCOM USERS



April, 1982 £1.00

Q. WHAT'S that superb-looking P.C.B. on the Front Cover?
A. The NEW 64K RAM, PCG and Buffer Board from MICRO POWER LIMITED

From WHO ?

MICRO POWER LIMITED The company formed by PROGRAM POWER to handle the marketing and distribution of their hardware and associated products, as well their excellent range of software.

Q. Great. But what's so special about this new board?

A. 1. High Quality, double-sided PCB with plated through holes
2. It will take up to four 16K blocks of dynamic RAM, mappable at any 4K Boundary
3. Room for an on board programmable character generator
4. Nascom 1 Buffer Board, providing buffering facilities, power-on/ reset logic (incl. re-set to any 4K boundary), and accepting the 8K Basic ROM

Q. How much, then and are they available?

A. **Only £39.50 + Vat & 55p postage.**

And available in production quantities from the middle of June.

Q. Can I secure one of the first now?

A. Certainly, Just send a cheque or credit card order (neither of which will be banked until date of delivery).

---

* * WORDEASE IN ROM * *

Our wordprocessor is now available in a 4K ROM. This single package wll provide you with the following facilities:-

Comprehensive on screen editing
Copy and reposition blocks of text
Read text in from tape
Find and replace routine
Insert embedded printer control codes
Set Tabs, Indents, Line Length, Page Length
Single or Double Spacing
Use your own parallel or serial printer routines, or those contained in Wordease
Automatic output of page number
Print all the text or any section
Turn justification on/off

In 2732	£25.00+35p Postage+V.A.T.
Other EPROMS	Price on application

---

* Ribbon Refills for Epsoms *

Why pay £6.00 or more. Refill your old Epsom cartridge quickly and cleanly with one of our ribbon packs.

£2.95+VAT

CONTENTS

EDITORIAL

I intend to use this issue's editorial for another plea for material. We can only continue to publish the magaqzine if we get enough articles to fill our pages, so please dig out some contribution and send it in.

In the reader replies on the back of the subscription forms the two main requests were for programs and reviews of hardware. So if you have added some interesting goodies to your system and are prepared to write a short article about them, or if you have an interesting program you would like to see published, then we shall be particularly interested to hear from you.

Remember, it is no use waiting for someone else to write articles if you are not prepared to contribute yourself. The magazine is written for enthusiasts, by enthusiasts, and it will only continue to exist with your enthusiastic support.

MODIFYING THE NASCOM 2 BOARD FOR 2732 EPROMS

by D. A. Boyd

At present, 4K byte EPROMs offer the best value for money in ROM program storage. For example, a number of retailers sell 2732s for around £4.00 to £4.50. This article describes the modifications needed to fit 2732 EPROMs to the Nascom 2 main board, with bank selection to select one-of-four 8K byte EPROM banks, or the Nascom Basic ROM.

The flexibility built into the Nascom 2 board means that 4K EPROMs can be fitted by rewiring the link blocks. The 2732 requires two extra address lines, wired from the 'special' link block, LKB9. Bank selection is done by a small board which takes control of the chip select decoder, IC46. The bank selection board plugs into the socket occupied by the memory selection links LKS, and needs 9 soldered connections to the main board. Bank 0 is always selected on reset, and the active bank is changed by OUT 3, n, where n is 0 - 4 as required. Port 3 WR is also spare on the Nascom 2, and could be used to provide a safety interlock against accidental bank switching.

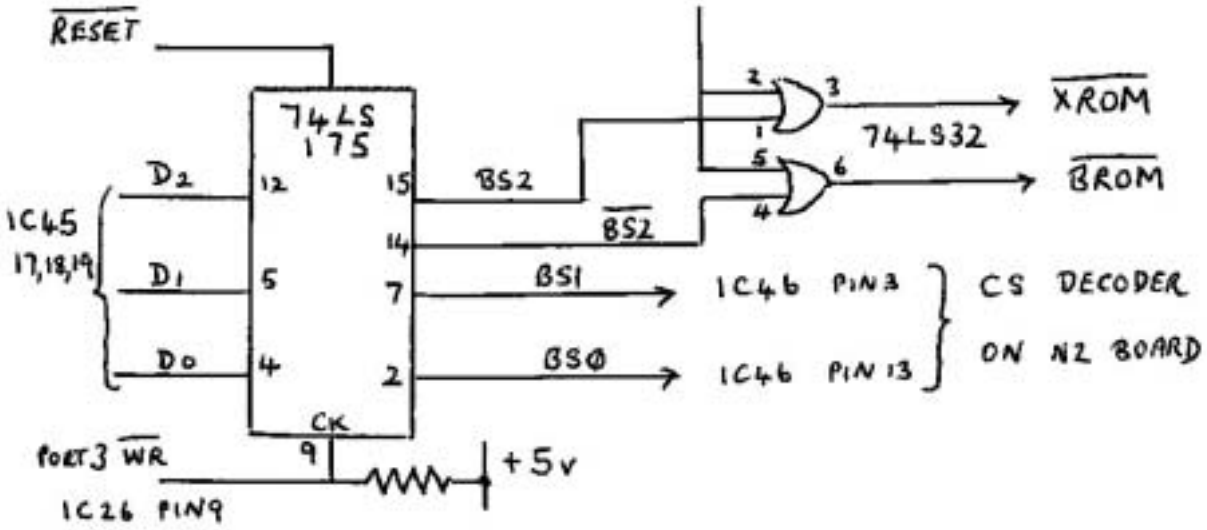


Fig. 1 BANK SELECTION LATCH

The bank control board is shown in figure 1. The 74LS175 latch is a three bit output port, strobed by the inverted PORT 3 WR signal from the I/O decode PROM IC26. Latched data bits BS0 and BS1 are connected to pins 13 and 3 of the chip select decoder IC46, to select one-of-four EPROM banks. These two IC pins

must be bent sideways so that they miss the socket, and can be wired to the Bank Select board. Data bit BS2 and its complement are gated with E000-FFFF from the MD PROM in a quad OR gate, 74LS32. A logic 0 on BS2 will select XROM and hence the EPROM block, and a logic 1 will select the Basic ROM (BROM). The CLR input of the 74LS175 is wired to the system RESET line, ensuring that bank 0 is selected on reset or power-up.

Wiring of the link blocks LKB1 - LKB8 is shown in figure 2. Addresses A10 and A11 are wired from the workspace RAM linkblock, LKB9. Link switches LSW1/7 and LSW1/8 should be up. Note that the wiring scheme shown is only suitable for 2732s; Texas 2532s do not have the same pinout.

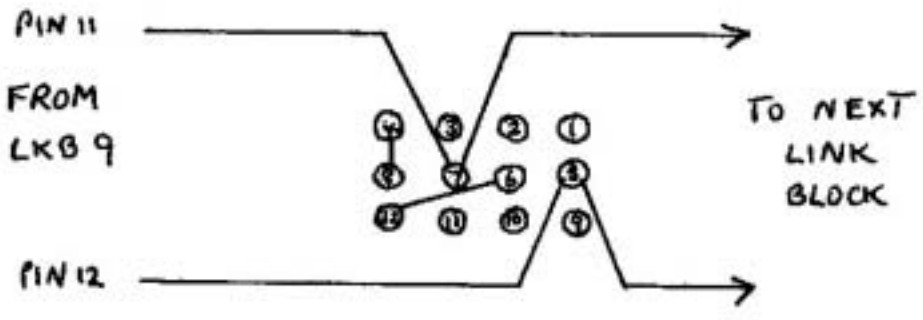


Fig. 2 LINK BLOCK WIRING FOR 2732 EPROMS

One further complication of this method of bank selection is that the addresses E000-EFFFH are confined to EPROM block A, and addresses F000-FFFFH are in block B. This is best explained by the table bellow:-

	Bank 0	Bank 1	Bank 2	Bank 3
E000-EFFF	A1 (IC35)	A2 (IC36)	A3 (IC37)	A4 (IC38)
F000-FFFF	B5 (IC39)	B6 (IC40)	B7 (IC41)	B8 (IC42)

My own system Jumps to D000H on reset, where a menu display and control program select the required bank and execute at the appropriate address. Software intended for execution in RAM is copied down to 1000H by the control program.

Figure 3 shows a suitable vero layout for the bank selection unit. The unit is shown from the component side.

It is connected to the header plug LKS by means of 9 veropins attached to the copper side of the board at the marked locations. Note that the breaks in the copper tracks are not shown in the layout.

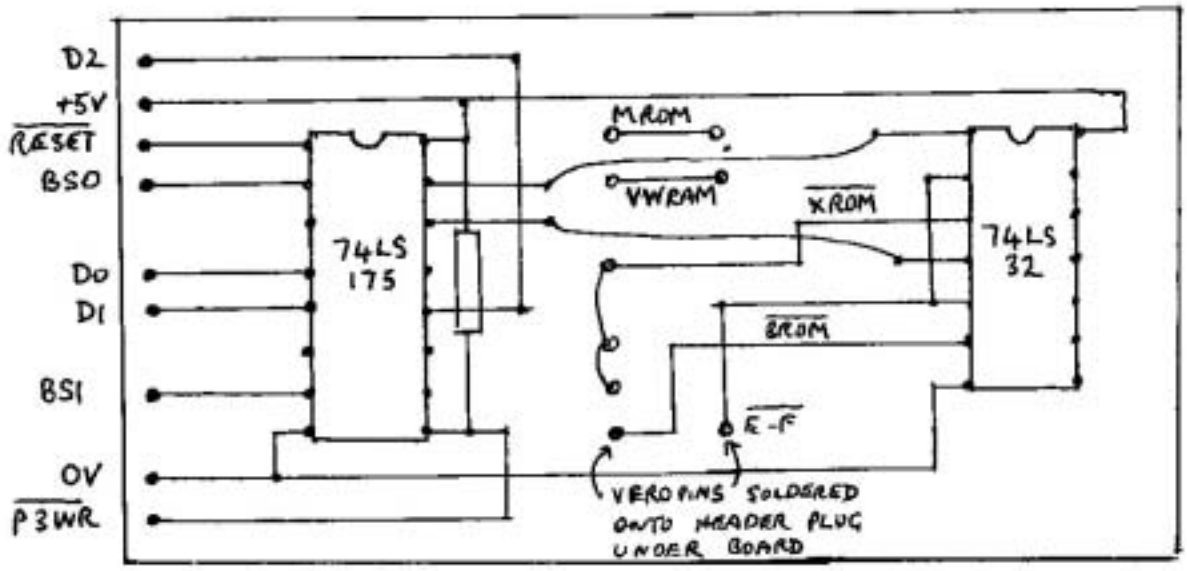


Fig. 3 VERO LAYOUT OF BANK SELECTION UNIT

THE POOR MAN'S DISC

LOGIC CONTROLLED TAPE DECKS

by David Elliott

Most computer hobbyists work on a shoestring budget, and finding the money to buy expensive disc drives can be difficult. But anyone who has to rely on cassette systems for storing and retrieving programs knows the frustration of having to search through tapes for the required program, and the inherent unreliability of the tape system when using domestic audio cassettes. One reason for this unreliability is the variable quality of the tape in most cassettes, and some so-called 'digital' tapes are just as bad, with drop outs caused by pinholes in the magnetic coating and even folds in the tape being common. Everyone has their favourite tape, and I use TDK-C46 tapes, which I have found to be very high quality.

Nevertheless, the average hobbyist is unlikely to be able to discard cassette tape as his storage medium until the cost of disc systems reduces drastically, and so a way has to be found to make the system more flexible and more reliable. The introduction of the cheap logic-controlled cassette deck onto the hi-fi market led us to consider controlling such a deck from the Nascom output port, and writing a cassette operating system to give many of the features of disc drives at a fraction of the cost, whilst retaining the standard Nascom tape format, allowing complete compatability with standard tapes.

This is how E.C.O.S. was born. The Elliott Cassette Operating System is an attempt to enable all the hard work of locating, storing and reliably retrieving programs from tape to be carried out by the computer. The hardware couldn't be simpler, as most logic-control led decks have a convenient remote control socket enabling easy interfacing with the computer. The deck selected was a Scott 665DM which costs £75, but many similar decks are on the market at around £75 - £80. Audio quality is not paramount, and can even be said to be a disadvantage in some respect , as the program information is encoded as a series of audio tones of 1200 hz and 2400 hz, and it would be better to suppress frequencies much outside this range. Another problem encountered was that of level matching. The standard cassette interface on the Nascom expects to receive a high level signal (1 - 2 volts peak to peak), but most hi-fi decks are intended to feed a high quality amplifier and are therefore designed to give a relatively low level output, typically 50 - 100 mv into 600 - 50K ohms. This was the case with the Scott deck, and a simple two stage transistor amplifier had to be interposed between the output and the Nascom cassette input in order to produce the required level. Another alternative would have been to use the headphone output, but this was not as reliable.

Control of the cassette deck was via the remote socket, and it was found that all the front panel buttons controlling the tape transport were brought out to an

eight pin socket on the rear panel, and that simply taking the required pin to logic low briefly was enough to operate an internal latch which enabled the appropriate function. This was eventually done direct from the Nascom output port, but could easily have been done via a reed relay similarly driven. Only one modification was made to the deck internally, and this was to bring a connection out from a reed relay which monitored the tape digital counter, pulsing to logic low ten times every digit. This enables ECOS to calculate tape speed, and hence use timing to fast-forward and reverse the tape, to speed up searching operations.

Having achieved total control of the tape transport via the computer, the next task was to decide upon the features to be included in the operating system, and the additional information to be recorded at the start of the tape and as header to each program stored. At the start of each tape there is a header which gives the tape number and name, and at the start of each program on the tape there is a header giving program number; name (up to 16 characters); type of record (machine code, zeap file, basic, data file, erased); length of program; and execution address. This information enables ECOS to create a catalogue on request by reading off the headers, fast-forwarding automatically between programs, and fast rewinding when the end of tape marker is found. Any tape erors which occur are handled automatically by ECOS which rewinds one block and attempts to reread the faulty block up to four times before abandoning it, leaving the standard '?' to denote an uresolved tape error. This sometimes occurs with old tapes not recorded on the deck. A dodge sometimes used in this case is to re-read the block with only one of the stereo channels, and this usually retrieves the situation. Once recorded on the logic deck via ECOS loading is usually foolproof, and the computer can be left to 'do its own thing'.

Once loaded ECOS is controlled via a menu which gives one letter command for:-

A -- Assembler (warm starts Zeap)
C -- Catalogue (prints last directory used)
D -- Directory (print a list of programs on tape)
E -- Erase program
G -- Load and execute program
I -- Initialise tape (create tape header)
L -- Load program
N -- Nas-Sys (returns to monitor)
R -- Read tape (Nas-Sys read)
V -- Verify (Verify program written under ECOS)
W -- Write program (under ECOS)
Z -- Write Zeap file

As it Stands, ECOS is 3K long, and with additional refinements it is intended to put it in EPROM and interface Nas-Sys to it, providing ECOS functions direct from Nas-Sys. ECOS by its nature is inherently machine-specific, but more details of the software will be published if there is a demand.

SCAL KBD, DF 61

The keys of the keyboard are connected in an array of eight rows by seven columns (six columns in the case of the Nascom 1 keyboard). Each row of keys is connected to one output line of a chip known as a 'BCD to Decimal Decoder/Driver'. This chip accepts a four-bit binary pattern and if it represents a valid decimal digit (i.e., if it is in the range 0 - 9) it drives the corresponding output line to zero volts. Three of the input lines are taken from a binary counter, which merely counts a series of clocking pulses and outputs a corresponding binary number; the fourth bit is derived from the clocking pulse, in such a way that one of the eight lines used by the key rows is activated only for a short period after the clocking pulse has been received.

Each key is effectively a miniature transformer, the magnetic circuit of which is only complete when the key is depressed. Thus as a row is pulled to zero volts by the decoder/driver, a pulse is output on by each key in that row which is down. The pulses are amplified and output to the data bus via a buffer which is only enabled when port 0 is read. Thus the circuitry produces a sequential scan of the keys and makes the information on which keys are pressed available to the processor.

The clocking pulses which step the binary counter for the keyboard scan are produced under the control of SCAL KBD. The routine starts by 'flipping' bit 1 of port zero, that is, taking this bit to 1 for a short period, and then returning it to 0. This sends a pulse along the 'Keyboard Reset' line, which resets the binary counter to zero. The status of row 0 is then read, complemented so that keys pressed are represented by 1's, and saved in the bottom byte of the KEYMAP region of the workspace (£0C01 - £0C09).

The routine now scans each row of the keyboard by flipping bit 0 of port 0 eight times. Each time bit 0 is flipped the next key row is selected; the routine then reads the status of the keys in that row, complements the result, and saves it temporarily in the D register. It then looks to see if it differs from the status obtain the last time the row was scanned, which is stored in the appropriate byte of KEYMAP. If there has been no change the routine continues to scan successive rows. After all the rows have been scanned the carry flag is reset and the routine is terminated.

If a change is detected, a short delay (approximately 2.7 msec at 4 mhz) is inserted and the status is re-read. This is designed to remove spurious inputs caused by key bounce. The value obtained on this second read is stored in the E register, and the first read is recovered from D; this value is 'Exclusive ORed with the mapping

byte, so that the accumulator now contains a 1 at each position corresponding to a changed bit. The accumulator is now rotated right so that successive bits pass into the carry flag while a single bit is rotated left in the D register and the rotations are counted in the C register; this process stops as soon as the carry flag is set. The result is that D now contains a single bit set corresponding to the pressed key with the lowest column number, and C contains that column number.

D is now used as a mask to select the appropriate bit from the second read of the row (contained in E), and to compare it with the same bit of the mapping byte. If the two are identical, it is assumed that the first 'change' was spurious, and the routine continues with the next row. When the two are different, the mapping byte is updated and the status of the bit in E is tested. If this bit is zero, the change was caused by the release of a key, and no further action need be taken. However, when the bit is set a key press has occurred, and the routine now has to determine the ASCII code of the key.

The information identifying the key pressed is first combined into a single byte in which bits 0 - 2 represent the column number, obtained from the C register, bits 3 - 6 represent the row number, from the B register, and bit 7 is set to 1 if the shift key was pressed. A table is then searched for this identifying byte; The HL register pair is set to the top byte of the table, BC is loaded with the length of the table, and the search is carried out by the CPDR instruction (Compare, Decrement, and Repeat). The table is arranged in such a way that when the byte is found the C register contains tha ASCII code of the corresponding key. If the byte is not found in the table, the search is repeated for the unshifted byte (i.e., with bit 7 reset). If this second search also fails, the carry flag is reset and the subroutine aborted.

When an ASCII code has been obtained from the table the subroutine proceeds to test the shift, graphics and control keys, and the Keyboard Option byte (£0C27), modifying the ASCII code in the appropriate manner. Finally, the carry flag is set, to indicate that a valid key press has been detected, and the routine is terminated.

If two or more keys are depressed simultaneously, the key in the lowest row will be detected first; if the keys are in the same row, the one with the lowest column number will be detected first. The corresponding mapping byte will be updated, and on the next scan no change will be detected in the first key, so the routine will now deal with the next key in the row or column priority. You will note that row 0 is scanned twice; once at the beginning of the routine, when its status is stored at £0C01, and once at the end of the routine when it is scanned as row 8 and its status stored at £0C09. The first scan is a special one carried out because the status of the shift key is needed whenever a keypress is detected; in the second scan

the keys in row 0 are read in the standard way.

SCAL IN, DF 62

This input routine picks up the address of the input table from the workspace at £0C75; this table contains a series of numbers representing Nas-Sys subroutines, and these are called in turn until a zero entry is reached. The address at £0C75 normally points to a table which contains two subroutines, the keyboard input and serial input, but it can be reset by the U or the X commands, or it can be changed by the user to point to a table of his own. In Nas-Sys 1 the keyboard routine in the input table is DF 61, but in Nas-Sys 3 it is the repeat keyboard routine, DF 7D, which itself calls DF 61 as a subroutine.

SCAL INLIN, DF 63

This routine calls SCAL 7B, which blinks the cursor while waiting for a key to be pressed. When a key is pressed the character is printed and if it was not a carriage return the routine loops back to SCAL 7B. When a carriage return is detected, the current address of the cursor, which was moved to the start of the next line by the carriage return, is loaded into HL from £0C29, DE is set to -64, HL and DE are added together and interchanged, and the subroutine is terminated. The result is that on return DE contains the address of the start of the line which the cursor was on when carriage return was pressed.

SCAL NUM, DF 64

This routine converts a single string of hex characters into a sixteen bit number. On entry the DE register pair should point to the start of the string, although leading blanks are ignored. As each character is obtained from the string it is tested for validity (i.e., is it ASCII 0 - 9 or ASCII A - F). If an invalid character is detectd the carry flag is set and the routine ends. Each valid character is converted to a binary number in the range 0 - 15, counted in location £0C20, and the four bit value is rotated left into £0C21, using the instruction RLD (Rotate Left Decimal). This instruction transfers the bottom four bits in the accumulator into the bottom four bits of (HL), the bottom four bits in (HL) are transferred into the top four bits, and the top four bits of (HL) are transferred back to the bottom four bits of the accumulator. The HL register pair is now incremented and the RLD instruction repeated. The result is that the characters in the string are successively converted into a sixteen bit number in NUMV. If the number overflows on the second RLD instruction, this indicates that the hexadecimal string represented a number greater than £FFFF, the carry flag is set, and the routine terminates. When the end of the string is encountered, marked by a space character or a null character (screen margin) the routine returns with the carry flag reset, the value of the string in NUMV,

and the number of characters in the string in NUMN.

SCAL CRT, DF 65

This outputs the contents of the accumulator to the screen. The routine first tests for a null (00) or a line feed (0AH); these are ignored. The next control code to be handled is Clear Screen (0CH). On receipt of this code 48 spaces are written in the top line (£080A - £0839), 16 nulls are written in the margin (£83A - £849), the line is copied 16 times to fill the whole screen, and the cursor is repositioned to the top left.

The remaining control codes are tested for in turn and the appropriate action taken. Normal characters are inserted at the current cursor position and the cursor moved one space right. If this takes it into the screen margin, the margin bytes are skipped. When the cursor moves beyond the limits of the screen RAM, the screen is scrolled by copying the bottom 14 screen lines (£084A - £0BB9) up one line, clearing the bottom line, and repositioning the cursor to the bottom left.

SCAL TBCD3, DF 66

The contents of the HL register pair are printed as a four-character hexadecimal number followed by a space by this routine. The H register is first transferred to the accumulator and printed out using SCAL TBCD2, DF 67; the L register is then treated similarly, and a space character is then output.

SCAL TBCD2, DF 67

This routine prints out the contents of the accumulator as two hexadecimal digits; it differs from SCAL B2HEX, DF 68, which forms the last part of the subroutine, only in that it adds the byte being printed to the checksum in register C.

SCAL B2HEX, DF 68

The contents of the accumulator are rotated right four times so that the most significant nibble becomes the least significant nibble. SCAL B1HEX, DF 7A, is then used to print the necessary hex digit. The accumulator contents are recovered, and the appropriate digit for the bottom nibble is printed. The method of converting each nibble is rather clever; 90H is added to the nibble and the DAA (Decimal Adjust Accumulator) instruction is applied to the result. This produces a value in the range 90H - 99H, with the carry flag reset if the original nibble is in the range 0 - 9. When 40H is added with the carry flag, the result lies in the range 30H - 39H, or 41H - 46H; i.e., the appropriate hex digit has been produced!

NASCOM USERS

Take a look at the NASCOM APPROVED HS-IN STORAGE SYSTEM. Where else can you get features like these . . .

A full on screen instant display of the catalogue
Auto vertification of each file as it is written.
CRC error checking.
Link selectable 2Mhz or 4Mhz option.
Fast data transfer rate of 6000 bps.
Powered from NASBUS.
8" sq NASBUS compatible PCB.
Far more reliable than any floppy disk system.
112K on-line storage with 2 drive system.

The HS-IN has a Command Set which makes it a floppy-disk "look-alike". It can load an 8K program in under 11 seconds and can store up to 56K (28 files) on each side of tape. Why spend £700 on a floppy disk system when the less expensive HS-IN system has a command set like this . . .

BRIDGE THE GAP BETWEEN EXPENSIVE FLOPPY DISK SYSTEMS AND UNRELIABLE CASSETTES.

B – Write a Basic file
C – Instant display of catalogue.
D – Delete file.
J – Jump to Basic.
N – Jump to NAS-SYS.
Q – Warm start to NASPEN text editor
R – Read a file.
T – Transfer file to another drive.
W – Write a file.
X – Exit and rewind cassettes.
Z – Warm start to Basic.

This Mini-Cassette Storage System is technologically far ahead of anything like it on the market and is extremely reliable into the bargain. AND THE COST?

Single Drive System built and tested	£199
Double Drive System built and tested	£279

Carriage £3.50.

OFFER NO 1
ALL RAM B boards supplied until April 30th come with an **EXTRA 32K FREE** on board.

OFFER NO 2
NASCOM 2 built, 48K RAM B board built, 3A PSU
ONLY £360 + VAT. **SAVE £37.50** SEE OFFER 1

OFFER NO 3
NASCOM 2 built, 48K RAM B board built, 3A PSU, HS-IN SINGLE DRIVE SYSTEM
ONLY £530 + VAT. **SAVE £66.50** SEE OFFER 1

ALL OFFERS END APRIL 30TH 1982

OFFER NO 4
NASCOM 2 built, 48K RAM B board built, 3A PSU, HS-IN single drive system, EPSON MX80FT-1, NASPEN & ALL CABLES **ONLY £900** + VAT. **SAVE £125.50** SEE OFFER 1

OFFER NO 5
NASCOM 3, 48K RAM B built, Gemini Intelligent Video Card (IVC)
ONLY £540 + VAT. **SAVE £65** SEE OFFER 1

OFFER NO 6
SHARP MZ80K WITH SUPER GRAPHICS +5 GAMES, EPSON MX80FT-1 WITH PAPER.
ONLY £825 + VAT. **SAVE £229**

MICRO-SPARES JOIN MICROVALUE GROUP

MICRO-SPARES have now become the MICROVALUE GROUP member supplying Scotland and now add super new products like the Gemini, Sharp & Epson to the MICRO-SPARES range.

QUALITY MEMORIES

AT PRICES THAT CANNOT BE BEATEN IN THE U.K. MICRO-SPARES can supply these memories in quantities from 1 to 10,000+. Parts delivery is fast – orders received by 4.30pm are shipped same day.

All memories are guaranteed for 1 year from date of purchase. Memories supplied are good quality but should you have a faulty part a replacement will be sent as soon as the part is received – without question.

Thousands of memories have already been supplied to Manufacturers Computer Traders, Government Bodies and Individuals all over the U.K. and the continent. If you are buying in large quantities please telephone for price. Official orders are welcome!

		1–49	50–249
2114L	(200ns & 300ns) low power Suitable for Acorn Atoms	96p	93p
2114N	(200ns & 300ns)	96p	93p
4116	(250ns)	63p	63p
	(200ns)	65p	63p
	(150ns)	75p	73p
2708	(450ns)	1.40p	1.34p
2716		2.05p	1.92p

Buy an EPSON PRINTER – Get a COMPUTER FREE!

YES – YOU WILL RECEIVE A FREE SINCLAIR ZX81 WITH EVERY ONE OF THE MODELS BELOW. EVEN THOUGH THE PRICE IS GOOD – BUT HURRY – THIS OFFER LASTS AS LONG AS THE ZX81's ARE IN STOCK

EPSON MX80T	£359 + VAT
EPSON MX80FT1	£399 + VAT
EPSON MX80FT2 (new type)	£465 + VAT
EPSON MX100	£575 + VAT

PAYMENT AND DELIVERY
Payment is by Cheque, Postal Order, ACCESS, VISA etc. PLEASE add postage and VAT. Postage on component orders under £30 is 50p. All in stock items sent same day. All none Kit items have a 1 year guarantee. Official orders welcome. Discount on large orders by arrangement.

SUPPLIERS TO TRADE
LOCAL GOVERNMENT
EDUCATION
INDIVIDUALS
INDUSTRY

EDINBURGH

Micro-Spares

19 Rosebum Terrace, Edinburgh EH12 5NG.
Tel: 031-337 5611.

BARCLAYCARD VISA

SCOTLAND

COMPUTERS
PERIPHERALS
COMPONENTS
& NATIONWIDE
MAINTENANCE

EPROM PROGRAMMER/CHECKER/READER

by C. Bowden

This article continues the listing of the controlling software for the Eprom programmer.

```
350         VERFY4      CALL RESET1
351                     LD A, (ERRFLG)
352                     CP 0FFH             ;   IF 0FFH, THERE WERE ERRORS
353                     JR Z, VERFY5        ;   SO SKIP O.K. MESSAGE
354                     CALL CLRCRT
355                     LD HL, TEXT16       ;   ELSE SAY COMPARISON O.K.
356                     LD DE, 090BH
357                     LD BC, 1BH
358                     LDIR
359                     JR VERFY6
360         VERFY5      DEFB SCAL, TDEL     ;   HOLD DISPLAY 2 SECS
361                     DEFB SCAL, TDEL
362                     LD HL, TEXT5        ;   ERROR MESSAGEREEN 363
364                     LD DE, 090BH
365                     LD BC, 24
366                     LDIR
367         VERFY6      DEFB SCAL. TDEL     ;   2 SECS PAUSE
368                     DEFB SCAL, TDEL
369                     JP RESTRT           ;   EXIT ROUTINE
370     ;   ********************************
371     ;   ROUTINE TO COPY EPROM INTO RAM
372     ;   ********************************
373         TRNFER      LD HL, TEXT9        ;   'FILLED FROM EPROM'
374                     LD DE, 0A4BH
375                     LD BC, 11H
376                     LDIR
377                     CALL RAMADR
378                     LD DE, 0000H
379         TRNFR1      LD A, (ROMFLG)
380                     CP D
381                     JR Z, TRNFR2        ;   JUMP IF ALL DONE
382                     CALL ENABLE
383                     IN A, (ADATA)       ;   GET BYTE FROM EPROM
384                     LD (HL), A          ;   STORE IT IN MEMORY
385                     CALL COUNT          ;   INCR. ADD., DISABLE CHIP
386                     INC HL              ;   NEXT MEMORY LOCATION
387                     INC DE              ;   INCREMENT BYTE COUNTER
388                     JR TRNFR1           ;   CONTINUE TILL FINISHED
389         TRNFR2      CALL RESET1
390                     CALL MESS19         ;   ALL DONE MESSAGE
391                     JP RESTRT
392     ;   ********************************
393     ;   ROUTINE TO CHECK IF FULLY ERASED
394     ;   ********************************
395         ERASED      LD DE, 0000
396         ERA1        LD A, (ROMFLG)
397                     CP D
398                     JR Z, ERA3          ;   JUMP IF FINISHED
399                     CALL ENABLE
400                     IN A, (ADATA)       ;   GET BYTE FROM EPROM
401                     CP 0FFH             ;   IS IT 'FF'
```

```
402                 JR NZ NOTERA        ;   IF NOT, JUMP
403                 CALL COUNT
404                 INC DE
405                 JR ERA1             ;   KEPP GOING
406     NOTERA      LD HL, TEXT14       ;   EPROM NOT ERASED
407                 LD DE, 090BH
408                 LD BC, 1AH
409                 LDIR
410                 JR ERA4             ;   JUMP TO END OF ROUTINE
411     ERA3        LD HL, TEXT15       ;   EPROM ERASED
412                 LD DE, 090BH
413                 LD BC, 15H
414                 LDIR
415     ERA4        CALL RESET1
416                 DEFB SCAL,TDEL      ;   2 SECS DELAY
417                 DEFB SCAL TDEL
418                 JP RESTRT
419 ;   ***********************************
420 ;   ROUTINE TO OUTPUT EPROM TO PRINTER
421 ;   ***********************************
422     OUTPUT      CALL ROMADR         ;   GET NORMAL ADD. OF ROM
423                 LD DE, 0000H        ;   BYTE COUNT
424     BUFF        LD IY, LINBUF       ;   POINT TO 16 CHAR. STORE
425     OUT1        LD A, (ROMFLG)
426                 CP D                ;   SEE IF ALL DONE
427                 JR Z, OUT6
428     OUT2        LD HL, (STOR1)      ;   GET OFFSET
429                 ADD HL, DE          ;   FORM ROM ADDRESS
430                 LD A, H             ;   AND PRINT IT
431                 CALL PRTHEX
432                 LD A, L
433                 CALL PRTHEX
434                 CALL GAP            ;   4 SPACES
435     OUT3        LD B, 16            ;   16 BYTES PER LINE
436     OUT4        CALL ENABLE
437                 IN A, (ADATA)       ;   GET BYTE
438                 LD (iY), A          ;   SAVE CHARACTER
439                 PUSH BC
440                 CALL COUNT          ;   INC. ADD., TURN CHIP OFF
441                 POP BC
442                 INC IY
443                 INC DE
444                 DJNZ OUT4           ;   LOOP FOR 16 BYTES
445                 CALL PRTLIN         ;   NOW PRINT LINE
446     OUT5        JR BUFF             ;   LOOP TILL ALL DONE
447     OUT6        CALL RESET1
448                 CALL MESS19         ;   COMPLETED MESSAGE
449                 JP RESTRT           ;   BACK TO START
450     ;       ***********************************************

455 ;   ROUTINE TO PRINT ALL 16 CHARS. IN LINE BUFFER
456     ;       ***********************************************

457     PRTLIN      LD B, 16            ;   NO. OF CHARS IN BUFFER
458                 LD IY, LINBUF       ;   IY POINTS TO BUFFER START
459     PRLIN1      LD A, (IY)          ;   GET CHARACTER
460                 CALL PRTHEX         ;   PRINT HEX AS 2 ASCII CHARS.
461                 LD A, 20H           ;  SPACE BETWEEN BYTES
```

```
462                 CALL PRINT          ;   PRINT IT
463                 INC IY              ;   NEXT CHARACTER
464                 DJNZ PRLIN1         ;   LOOP FOR 16 CHARACTERS
465                 CALL GAP            ;   PRINT 4 SPACES
466                 LD B, 16            ;   NOW DO 16 ASCII CHARACTERS
467                 LD IY, LINBUF       ;   START OF BUFFER
468     PRLIN2      LD A, (IY)          ;   GET CHARACTER
469                 CP 20H              ;   IS IT A CONTROL CHAR.?
470                 JP M, DOT           ;   IF SO, PRINT A DOT
471                 CP 7BH              ;   IS IT GREATER THAN z?
472                 JP P, DOT           ;   IF SO, PRINT A DOT
473                 JR DOT1
474     DOT         LD A, "."
475     DOT1        CALL PRINT          ;   PRINT IT
476                 INC IY              ;   NEXT CHARACTER
477                 DJNZ PRLIN2         ;   LOOP UNTIL 16 DONE
478                 LD A, CR            ;   PRINT CARRIAGE RETURN
479                 CALL PRINT
480                 LD A, LF            ;   AND LINE FEED
481                 CALL PRINT
482                 RET
483 ;   ************************************
486 ;   ROUTINE TO O/P CHARACTER TO PRINTER
487 ;   ************************************
488     PRINT       PUSH AF             ;   SAVE CHARACTER
489     PR1         IN A, (HSHAKE)      ;   HANDSHAKE, BIT 7 PORT 0
490                 AND 80H
491                 JR Z, PR1
491                 POP AF              ;   RECOVER CHARACTER
492                 DEFB SCAL, SRLX     ;   NAS-SYS SERIAL O/P
493                 RET
494 ;   ****************************
496 ;   SUBROUTINE FOR 4 SPACE GAP
497 ;   ****************************
498     GAP         LD B, 4             ;   NUMBER OF SPACES
499                 LD A, 20H           ;   SPACE CHARACTER
500     GAP1        CALL PRINT          ;   PRINT
501                 DJNZ GAP1           ;   LOOP UNTIL DONE
502                 RET
503 ;   ************************************
504 ;   PRINT HEX CDE AS TWO ASCII CHARS.
505 ;   ************************************
506     PRTHEX      PUSH AF             ;   SAVE CHARACTER
507                 AND 0F0H            ;   GET TOP 4 BITS
508                 RRCA                ;   SHIFT TO BOTTOM 4 BITS
509                 RRCA
510                 RRCA
511                 RRCA
512                 CP 0AH              ;   IS IT MORE THAN 10
513                 JP P, ADD37A        ;   IF SO, JUMP
514                 ADD A, 30H          ;   CONVERT TO ASCII 0 – 9
515     PRTH1       CALL PRINT          ;   PRINT IT
516                 POP AF              ;   RECOVER CHARACTER
517                 AND 0FH             ;   GET BOTTOM 4 BITS
518                 CP 0AH              ;   MORE THAN 10?
519                 JP P, ADD37B        ;   IF SO, JUMP
520                 ADD A, 30H          ;   CONVERT TO ASCII 0 – 9
521     PRTH2       CALL PRINT          ;   PRINT IT
```

```
522                     RET
523     ADD37A          ADD A, 37H              ;   CONVERT TO ASCII A - F
524                     JP PRTH1
525     ADD37B          ADD A, 37H
526                     JP PRTH2
527 ;   ****************************
528 ;   GENERAL SUBROUTINES
532 ;   ****************************
533     ENABLE          LD A, 00                ;   ENABLE CHIP
534                     OUT (BDATA), A          ;   CONTROL PORT B
535                     LD A, 20H
536     STABLE          DEC A                   ;   WAIT FOR CHIP
537                     JR NZ STABLE
538                     RET
539 ;   ************************************
542 ;   SCROLL TO CLEAR CRT, CURSOR TO BOTTOM
543 ;   ************************************
544     CLRCRT          LD B, 0FH               ;   SCROLL 15 TIMES
545     CLR1            DEFB SCAL, CRLF         ;   TO CLEAR SCREEN AND
546                     DJNZ CLR1               ;   LEAVE TOP LINE INTACT
547                     RET
548 ;
549     RESET           LD A, 2AH               ;   12V, RESET, WE, OE BITS SET
550                     JR RESET2
551     RESET1          LD A, 0AH               ;   RESET, WE, OE BITS SET
552     RESET2          OUT (BDATA), A
553                     LD B, 10H
554     WAIT6           DJNZ WAIT6
555                     RES 3, A                ;   END OF RESET
556                     OUT (BDATA), A
557                     RET
558 ;   ************************************
559 ;   O/P SHORT PULSE TO INC. ADDR. COUNTER
560 ;   ************************************
561     COUNT           LD A, 6                 ;   COUNT PULSE ON
562                     OUT (BDATA), A
563                     LD B, 10H
564     WASIT7          DJNZ WAIT7
565                     LD A, 2
566                     OUT (BDATA), A          ;   TURN COUNT PULSE OFF
567                     RET
568 ;   **********************************
569 ;   SET PIO PORT A – INPUT, B – OUTPUT
570 ;         BOTH PORTS TO MODE 3
571 ;   **********************************
572     STPIO1          LD A, 0FFH
573                     OUT (BCTRL), A          ;   MODE 3
574                     LD A, 00
575                     OUT (BCTRL), A          ;   OUTPUT
576     STPIO2          LD A, 0FFH
577                     OUT (ACTRL), A          ;   MODE 3
578                     OUT (ACTRL), A          ;   INPUT
579                     RET
580 ;   **********************************
582 ;   SET PIO PORT A TO MODE 3, OUTPUT
583 ;   **********************************
584     STPIO3          LD A, 0FFH
585                     OUT (ACTRL), A          ;   MODE 3
```

```
                    LD A, 00
                    OUT (ACTRL), A      ;   OUTPUT
                    RET
;   **************************
;   ROUTINE COMPLETED MESSAGE
;   **************************
    MESS19      CALL CLRCRT             ;   CLEAR SCREEN
                LD HL, TEXT19           ;   ROUTINE COMPLETE
                LD DE, 090BH
                LD BC, 16
                LDIR
                DEFB SCAL, TDEL         ;   WAIT 2 SECONDS
                DEFB SCAL, TDEL
                RET
;   ************************************
;   GET START ADDRESS OF 1K OR 2K BLOCK
;     OR 4 DIGIT TYPE NUMBER OF EPROM
;     OR MEMORY START ADDRES OF EPROM
;   ************************************
    ROMADR      LD HL, TEXT18           ;   “NORMAL ROM START ADDR?”
                LD DE, 09CBH
                LD BC, 22H
                LDIR
                LD A, 00
                LD (SCNFLG), A          ;   SET JUMP BACK FLAG
                JR SCANT1
    RAMADR      LD HL, TEXT6            ;   MESSAGE TO CRT
                LD DE, 09CBH            ;   FOR ADDRESS
                LD BC, 2EH
                LDIR
                LD A, 0FFH
                LD (SCNFLG), A          ;   SET FOR JUMP BACK TO HERE
    SCANT1      LD HL, TEXT10           ;   PROMPT “ADDRESS ??”
                LD DE, 0B0BH
                LD BC, 12H
                LDIR
                LD HL, 0B19H            ;   SCREEN ADD. FOR ENTRY
    SCAN1A      LD B, 20H
                LD DE, STORLN
                LD A, 20H
    CLRLIN      LD (DE), A              ;   CLEAR LINE STORE
                INC DE
                DJNZ CLRLIN
                LD IY, STORE            ;   STORE FOR ENTRIES
                LD D, 4                 ;   FOUR KEY ENTRIES
    SCAN2       XOR A
    SCAN3       DEFB SCAL, KBD          ;   GET ENTRIES FROM KEYBOARD
                JR C, SCAN4
                JR SCAN3
    SCAN4       CP “O”                  ;   ONLY ACCEPT ENTRIES
                JP M, SCAN2             ;   IN THE RANGE 30H – 39H
                CP “:”
                JP P, SCAN5
                LD (HL), A              ;   PRINT IF O.K.
                SUB 30H                 ;   CONVERT TO 0 – 9
                LD (IY), A              ;   STORE IT
                INC HL                  ;   NEXT SCREEN ADDRESS
                INC IY                  ;   NEXT STORE
```

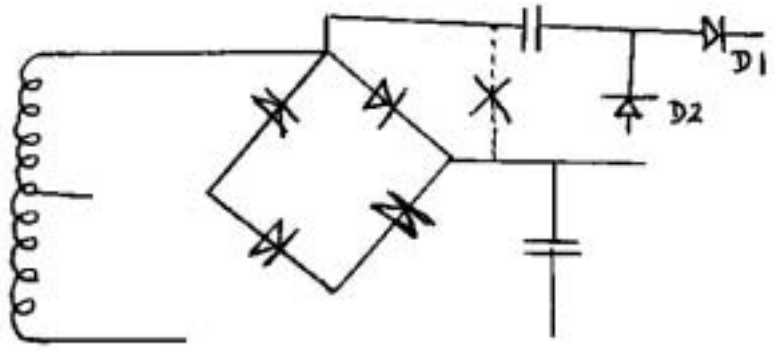
```
648                 DEC D               ;   REDUCE ENTRY COUNTER
649                 JR NZ, SCAN2        ;   JUMP IF NOT FINISHED
650                 JR SCAN7            ;   JUMP TO SCAN7 WHEN DONE
651     SCAN5       CP “A”              ;   IS IT ASCII A – F?
652                 JP M, SCAN2         ;   IF NOT, REJECT IT
653                 CP “G”
654                 JP P, SCAN2
655                 LD (HL), A          ;   PRINT IT IF O.K.
656                 SUB 37H             ;   CONVERT TO 10 – 15
657                 LD (IY), A          ;   STORE IT
658                 INC HL              ;   NEXT SCREEN LOCATION
659                 INC IY              ;   NEXT STORE
660                 DEC D               ;   REDUCE ENTRY COUNTER
661                 JR NZ SCAN2         ;   JUMP IF NOT DONE
662     SCAN7       LD HL, 0A4BH        ;   SAVE THIS MESSAGE IN CASE
663                 LD DE, STORLN       ;   ENTRY IS TO BE CHANGED
664                 LD BC, 20H
665                 LDIR
666                 DEFB SCAL, CRLF     ;   SCROLL CRT
667                 DEFB SCAL, CRLF     ; TWICE
668                 LD HL, TXT11A       ;   CORRECT –Y/N?
669                 LD DE, 0B0BH
670                 LD BC, 14H
671                 LDIR
672                 XOR A
673     SCAN8       DEFB SCAL, KBD      ;   GET ANSWER
674                 JR C, SCAN9
675                 JR SCAN8
676     SCAN9       CP “Y”              ;   ANSWER “YES”?
677                 JR Z SCAN6          ;   IF SO, JUMP TO SCAN6
678                 CP “N”              ;   ANSWER “NO”?
679                 JR Z SCAN10         ;   IF SO, JUMP TO SCAN10
680                 JR SCAN8            ;   REJECT OTHER REPLIES
681     SCAN10      CALL CLRCRT         ;   CLEAR SCREEN
682                 LD HL, STORLN       ;   RESTORE SAVED MESSAGE
683                 LD DE, 0A4BH
684                 LD BC, 20H
685                 LDIR
686                 LD A, (SCNFLG)      ;   FIND OUT WHERE TO JUMP
687                 CP 00
688                 JP Z, ROMADR        ;   IF ZERO, JUMP TO ROMADR
689                 JP RAMADR           ;   ELSE WAS FROM RAMADR
690                 JP SCANT1           ;   BACK TO GET ADDRESS AGAIN
691     SCAN6       LD IY, STORE        ;   POINT TO FIRST ENTRY
692                 LD A, (IY)          ;   GET FIRST
693                 RLCA                ;   ROTATE BITS 4 TIMES
694                 RLCA                ;   TO PUT VALUE INTO
695                 RLCA                ;       INTO      MOST      SIGNIFICANT
NIBBLE
696                 RLCA
697                 ADD A, (IY+1)       ;   ADD SECOND VALUE
698                 LD H, A             ;   SAVE IN H REGISTER
699                 LD A, (IY+2)        ;   GET 3RD ENTRY
700                 RLCA                ;     PUT IN MOST SIGNIFICANT
NIBBLE
701                 RLCA
702                 RLCA
703                 RLCA
```

```
704                     ADD A, (IY+3)           ;   ADD 4TH ENTRY
705                     LD L, A                 ;   PUT IN L REGISTER
706                     LD (STOR1), HL          ;   SAVE THE ADDRESS
707                     RET                     ;   BACK TO ROUTINE
708   ;   ****************************
709   ;   MESSAGES USED BY THE ROUTINES
710   ;   ****************************
711       TEXT1         DEFM   /EPROM PROGRAMMER/
712   TEXT1A          DEFM   /PRESS KEY ‘A’ FOR TYPE 2708 EPROM/
713       TEXT1B        DEFM  “KEY ‘B’ FOR TYPES 2516/2716”
714       TEXT2         DEFM  /PRESS P FOR ROUTINE TO PROGRAM EPROM/
715       TEXT2A        DEFM  /C TO COMPARE EPROM WITH MEMORY/
716       TEXT2B        DEFM  /T TO TRANSFER EPROM INTO MEMORY/
717       TEXT2C        DEFM  /E TO CHECK EPROM IS FULLY ERASED
718       TEXT2D        DEFM  /D TO DUMP EPROM TO PRINTER
719       TEXT3         DEFM  /TURN OFF PROGRAMMER WHILE CHANGEING
EPROM/
721       TEXT4         DEFM  /PRESS C TO CONTINUE/
722       TEXT5         DEFM  /ROM AND RAM DO NOT MATCH/
723       TEXT6         DEFM  /ENTER START ADDR(HEX) OF/
724                     DEFM  / 1K OR2K BLOCK TO BE /
725       TEXT7         DEFM  /COPIED INTO EPROM/
726       TEXT8         DEFM  /COMPARED TO EPROM/
727       TEXT9         DEFM  /FILLED    FROM EPROM/
728       TEXT10        DEFM  /START ADDRESS  ????/
729       TEXT11        DEFM  /KEY    -/
730       TXT11A        DEFM  “IS THIS CORRECT?-Y/N”
731       TEXT12        DEFM  /YOU MUST TYPE AN ‘A’ OR A ‘B’/
732       TXT13A        DEFM  /TYPE – 2708      /
733       TXT13B        DEFM  “TYPE – 2516/2716”
734       TEXT14        DEFM  /EPROM NOT FULLY ERASED/
735       TEXT15        DEFM  /EPROM IS FULLY ERASED/
736       TEXT16        DEFM  /COMPARISON OK. – NO ERRORS. /
737       TXT17A        DEFM  “SAME EPROM  -  Y/N?”
738       TEXT18        DEFM  /NORMAL R.O.M. START ADDR.(HEX) - ?/
739       TEXT19        DEFM  /ROUTINE COMPLETED/
740       TEXT20        DEFM  /CAUTION :ONLY ONE EPROM AT A TIME./
741     TEXT21        DEFM /OBSERVER THE EPROM HANDLING PRECAUTIONS/
742   ;   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
743                 .DEPHASE
744   ;   THIS DENOTES THE END OF A PROGRAM BLOCK IN
745   ;   THE MACRO 80 ASSEMBLER
746   ;   IT HAS NO EQUIVALENT IN ZEAP
747   ;   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
748                   .PHASE 0D00H
749       THIS DENOTES THE START OF A NEW BLOCK
750       IT IS EQUIVALENT TO THE ORG PSEUDO-OP IN ZEAP
751       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
752   ;   PROGRAM WORKSPACE AREA
753       SCNFLG        DEFS 1                  ;   JUMP BACK FLAG
754       ERFLG         DEFS 1                  ;   ERROR FOUND FLAG
755       ROMFLG        DEFS 1                  ;   1K OR 2K ROM FLAG
756       STORE         DEFS 4                  ;   STORE FOR KEY ENTRIES
757       STOR1         DEFS 2                  ;   MEMORY START ADDRESS
758       STORLN        DEFS 20H                ;   TEMPORARY MESSAGE STORE
759       LINBUF        DEFS 10H                ;   BUFFER FOR PRINTER O/P
760                     .DEPHASE                ;
761   ;   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
762                 END
763  ;    **********************************
```

This completes the listing of the control software.

A number of errors have come to light in the previous articles. The first is an error in the circuit diagram for the programmer power supply. The negative lead of the 470uP, 64V capacitor should connect to the bridge rectifier AC input, not to the positive output of the rectifier as shown.



Secondly, a link is missing from the vero layout. This is the link connecting pin 24 of the 2708 socket to the +5v supply. One may be fitted in a similar way to that on the 2716 socket.

Thirdly, a series of minor errors occurred in the listing. There was a superfluous right-hand bracket is assembler line number 144; it should read

```
144             LD (ROMFLG), A
```

Line 150 was omitted and line 149 was wrong; this section should read

```
149             JR PROMPT
150    TYP2K    LD A, 08H          ; FLAG FOR 2K EPROM
151             LD (ROMFLG), A
```

An incorrect address appeared in line 167; the correct address is 0A52H. In line 216 the label TEXT7 was given as TEXT17. In line 293 a superfluous WAIT2 occurred at the end of the line; this of course is the label for the delay loop on line 294

Finally, note 1 on page 20 of the last article could be expressed better as follows:-

1) The layout is shown from the copper side. All components including links are mounted on the other side of the board, except the links mentioned in Note 5 and the two diodes in Note 6.

+ - + - + - + - + - + - +

PLANNING AND WRITING A PROGRAM

by Viktor

INTRODUCTION

Program Power asked me if I had time to write a program to deal with the administration of their royalty payments. I didn't have , of course. Nor did I have time to write this article. However, like most computer enthusiasts, once a program idea has got implanted in my tiny mind it tends to act like the proverbial cuckoo and creates its own time and space. Here, therefore, follows the first results of my deliberations.

THE PROBLEM

Program Power sells programs which in the main have been written by individuals not employed by the company, these individuals being remunerated by way of 20% royalties, calculated on the prices at which the programs are sold.

Including those for certain micros whose names may not be printed in this magazine, there are t present approx. 100 programs being offered for sale, written by say 50 authors, though the number per author varies from one up to sixteen. Each Quarter the royalties payable are calculated according to number sold at the company's selling price price.

If there were only one price for each program, then it could easily be argued that the job could be handled by a competent clerk with a calculator. However, in recent months, due to the sale of programs to trade customers, at discounts varying with quantities sold, together with special offers to direct customers, the number of different prices has grown enormously. A computerised solution is therefore being sought before the Job gets into a tangle or consumes too much valuable time.

OUTPUTS

When writing a program, I always look first at the 'outputs' of the task, both in terms of which pieces of paper have to be produced, and also the detail which will appear on them. In this case there is firstly a statement to each author, giving his name and address, the names of his programs ,the number sold at each price, the royalty calculated and the grand total payable. These, of course, are printed on single sheets. The second output is a summary for each computer, listing the same information without the author's addresses, and giving the overall total for that computer. A continuous list is acceptable for this part of the Job.

INPUTS

The next stage is to look at inputs and to establish how they are to be made available to the computer. The main inputs are the authors' names and addresses,

nascom MEANS SOLUTIONS nascom MEANS PERFORMANCE

Nascom have come a long way since their acquisition by Lucas. With the knowledge of over 30,000 units already in the field you can buy with confidence from NASCOM.

PRODUCTS:
We have kits, built and tested boards, and our fully assembled and tested NASCOM 3 system with a full choice of configuration either cassette or disc based. Alternative operating systems include NAS DOS and CP/M.

SOFTWARE:
We have a team of programmers who are writing software and courseware especially for UK educational business and domestic users.

FREE ADVICE:
We have appointed experts to advise on the specialist use of micro computers in U.K. schools, homes or businesses.

BACK-UP:
We have a nationwide dealer network giving full sales back-up and after sales service. From our head office we have a service line to sort out any problems.

SYSTEM EXPANSION:
NASCOM machines are designed to grow with users. Easily and simply NASCOM systems can be expanded by adding extra modules to the basic system.

LUCAS LOGIC LIMITED
NASCOM MICROCOMPUTERS DIVISION,
Welton Road, Wedgnock Industrial Estate,
Warwick CV34 5PZ, England.

Learn more about NASCOM now. Complete the coupon for further information and a full list of dealers.

Have you a NASCOM programme? If so send for an application form for inclusion in our programme book

Dealer Enquiries Welcome

£1,548 + VAT
A typical system as shown, consisting of a Nascom 3 (48K), dual disks, monitor, and printer is available from your dealer now.

To Lucas Logic Ltd., Nascom Microcomputers Division, Welton Road, Wedgnock Industrial Estate, Warwick CV34 5PZ, England

Please send:
Literature ☐ Dealer List ☐ Prog. Book Form ☐

Name .....
Position .....
Establishment .....
Address .....
.....
Tel. No .....

PP2

Lucas Logic

the program names, the program prices and the numbers sold at each price. The first two are essentially 'standing' or file information. We will therefore need some means of building up a file of these, together with routines to enter new data and amend or delete existing data. My solution was to make use of Program Power's 'Basic File Handler' program, which gives amongst other things the ability to save and load string data.

The numbers of each program sold must obviously be entered each quarter. This leaves the price information. I established that, although there were perhaps only 10 basic prices for programs, the variations discussed earlier could easily multiply this by a factor of at least 10. Thus I thought it best to deal with this in the same way as numbers sold.

STRUCTURE & HANDLING

Name and Address File - this can fairly simply be held as follows:
N$(I), A$(I), B$(I), C$(I), D$(I) - where N is the name; A, B & C are a three line address; and D is the telephone number, (strictly speaking not required for this application). The subscript 'I' will be used as a key effectively to the file of authors.

As regards the programs, I decided to go for a two dimensional string array, as this permits ease of handling. Thus, for a file of 30 authors (covering just one of the micros), we have the array P$(I,J), where J is the key to the programs of each author. Since there are very few authors who have more than two programs, this is very wasteful of memory space, but as it is not a problem in this particular case, the benefits of ease of handling will be allowed to take preference.

I mentioned earlier that the majority of the information has to be output twice. This means that the price and numbers sold data must be held between one print run and the next. Alternatively, it could be saved as a temporary file, but this solution was not thought to be very elegant. My first solution to this problem was to create two three- dimensional numeric arrays viz. P(I,J,K) to hold the prices and S(I,J,K) to hold the relevant numbers sold. A quick calculation of 2(arrays) x 30(authors) x 20(programs) x 10 (prices) x 6 bytes per variable = 72000bytes persuaded me I had better think again.

I then thought of the way in which the information will be used. The main part of the program will start with the first author, deal with his programs in turn and then move on down the list of authors until they have all been looked at. A solution much less wasteful of space would therefore be to create two single tier arrays viz. K(X) for prices and L(X) for numbers sold, with a variable element deliberately left

equal to zero to signify the end of the prices for any one program. The routine to enter the prices would skip an element, and the routine to calculate and print the royalties would search for those elements equal to zero, and then move onto the next program. The calculation is then [2(arr ays) x say 75(programs)] + 75(vacant elements) = 225. Multiplying by the 6 bytes per element then gives 1350 bytes. Phew!!

On reflection, this method might also be adopted in handling the program name information. But since the program was written first and it works, we'll save that up our sleeves, in case we need the space in some future enhancement.

We do not of course have to write one large program containing all the options. The tasks to be done can conveniently be split into 'file maintainence' i.e. create, amend or delete the standing information, and the 'operating' program to enter the prices and numbers sold and print the letters and summary. However, since space did not prove a problem, the program has been left as one complete entity. The version which now follows has been more or less de-bugged, although some improvements, for example, in the print layout, remain to be made.

In the next issue I will cover some sections of the program in greater detail, and discuss the operation of the Basic File Handler.

. . . . . .

```
5 DOKE4271,-20482:DOKE4100,-14336:TC$="RESET":TR=USR(0)
10 CLEAR10000:DIMN$(30),A$(30),B$(30),C$(30)
15 DIMD$(30),E$(30),P$(30,20),K(100),L(100)
50 CLS:PRINTAB(17);"ROYALTIES PROGRAM"
55 SCREEN8,3:PRINT"Options:-
60 SCREEN5,5:PRINT"1)    New Authors
65 SCREEN5,6:PRINT"2)    New Programs
70 SCREEN5,7:PRINT"3)    Amend/Delete Authors/Programs
80 SCREEN5,8:PRINT"4)    Quarterly Sales/Prices
85 SCREEN5,9:PRINT"5)    Print Authors' Letters
90 SCREEN5,10:PRINT"6)    Print Summary
93 SCREEN5,11:PRINT"7)    Read Program File
96 SCREEN5,12:PRINT"8)    Write Program File
100 PRINT:SCREEN5,14:INPUTC
110 ONCGOTO200,400,600,1000,1200,1400,1600,1800
120 IFC<1ORC>8THEN50
200 FORI=1TO30
205 IFN$(I)=""THENZ=I:I=30:GOTO240
210 NEXT
220 PRINT"No Space for New Author !"
230 FORA=1TO3000:NEXT:GOTO50
240 CLS:I=Z:INPUT"Author's Name";N$(I)
245 IFLEN(N$(I))=<13THEN250
247 PRINT:PRINT"Max. 13 chrs.":FORC=1TO3000:NEXT:GOTO240
250 INPUT"Correct";L$
255 IFL$<>"Y"THEN240
260 INPUT"ADDR 1";A$(I)
```

```
265 IFLEN(A$(I))>20THENPRINT“Max. 20 chrs.”:GOTO260
267 INPUT“Correct”;L$
270 IFL$<>“Y”THEN260
275 INPUT“ADDR 2”;B$(I)
277 IFLEN(B$(I))>20THENPRINT“Max. 20 chrs.”:GOTO275
280 INPUT“Correct”;L$
285 IFL$<>“Y”THEN275
290 INPUT“ADDR 3”;C$(I)
292 IFLEN(C$(I))>20THENPRINT“Max. 20 chrs.”:GOTO290
295 INPUT“Correct”;L$
300 IFL$<>“Y”THEN290
320 INPUT“Tel. No.”;D$(I)
322 IFLEN(D$(I))>20THENPRINT“Max. 20 chrs.”:GOTO320
325 INPUT“Correct”;L$
330 IFL$<>“Y”THEN320
335 GOTO50
400 CLS:INPUT“Enter Author’s Name”;Y$
402 IFY$=“”THEN400
405 FORI=1TO30
410 IFN$(I)=Y$THENZ=I:I=30:GOTO440
415 NEXT
420 PRINT:INPUT“Name not Found – another try”:L$
430 IFL$=“Y”THEN400
435 GOTO50
440 I=Z
450 FORJ=1TO20
455 IFP$(I,J)=“”THENZ=J:J=20:GOTO470
460 NEXT
465 PRINT“No Space for New Program!”
467 FORA=1TO3000:NEXT:GOTO50
470 CLS:J=Z:INPUT“PROG. NAME”;P$(I,J)
472 IFP$(I,J)=“”THEN470
475 PRINT:INPUT“Correct”;L$
480 IFL$<>“Y”THEN470
485 PRINT:INPUT“Any More Progs”;L$
490 IFL$=“Y”THENZ=Z+1:GOTO470
495 GOTO 50
600 CLS:FORI=1TO30
605 PRINTI;“ ”;N$(I);:I=I+1
610 PRINTTAB(16);I“ ”;N$(I);:I=I+1
615 PRINTTAB(32);I“ ”;N$(I):NEXT
620 PRINT:INPUT“Which Author”;I:IFI<1ORI>30THEN600
622 IFN$(I)=“”THEN600
625 CLS:PRINTN$(I):PRINT:INPUT“Correct Author”;L$
630 IFL$<>“Y”THEN600
635 PRINT:PRINT:PRINT“Options”;TAB(20)“1) Programs”
640 PRINTTAB(20)“2) Names & addresses”
645 PRINT:PRINTTAB(20):INPUTC:IFC<1ORC>2THEN645
650 ONCGOTO700,800
700 CLS:PRINTN$(I)“:”:FORJ=1TO20
710 PRINTJ“ ”P$(I,J);:J=J+1
720 PRINTTAB(24);J;P$(I,J):NEXT
725 INPUT“Amend/Delete Y/N”;L$
730 IFL$=“N”THEN50
735 INPUT“Enter Program No.”;J
737 IFJ=999THEN50
740 IFJ<1ORJ>20THEN735
```

```
742 IFP$(I,J)=“”THENPRINT“Old data only!”:GOTO735
745 PRINTP$(I,J)
750 INPUT“Enter new data”;P$(I,J)
752 IFP$(I,J)<>“”THEN756
753 INPUT“Delete Program”;L$
754 IFL$=“Y”THEN775
755 GOTO750
756 INPUT“Correct”;L$:IFL$<>“Y”THEN750
760 INPUT“More changes this author”;L$
765 IFL$=“Y”THEN700
770 GOTO50
775 J=J+1:IFP$(I,J)=“”THEN700
780 P$(I,J-1)=P$(I,J):P$(I,J)=“”:GOTO775
800 CLS:PRINTN$(I):PRINT
805 PRINTA$(I):PRINTB$(I):PRINTC$(I):PRINT:PRINTD$(I):PRINT
810 INPUT“Amend/Delete Y/N”;L$
815 IFL$=“N”THEN50
820 PRINT“Name  “;N$(I)”   “;:INPUT”New Data  Y/N”;L$
825 IFL$=“Y”THENINPUTN$(I)
826 IFN$(I)<>“”THEN830
827 INPUT“Delete Author”;L$
828 IFL$=“Y”THENGOTO900
829 GOTO820
830 PRINT“ADDR1  ”;A$(I)“   ”;:INPUT“New Data  Y/N”;L$
835 IFL$=“Y”THENINPUTA$(I)
840 PRINT“ADDR2  ”;B$(I);“   ”:INPUT“New Data  Y/N”;L$
845 IFL$=“Y”THENINPUTB$(I)
850 PRINT“ADDR3  ”;C$(I);“  ”;:INPUT“New Data   Y/N”;L$
855 IFL$=“Y”THENINPUTC$(I)
860 PRINT“Tel.  No.   ”;D$(I); “   ”;:INPUT“New Data   Y/N”;L$
865 IFL$=“Y”THENINPUTD$(I)
875 GOTO50
900 I=I+1
905 IFN$(I)=“”THEN600
910 N$(I-1)=N$(I):N$(I)=“”
915 A$(I-1)=A$(I):A$(I)=“”
920 B$(I-1)=B$(I):B$(I)=“”
925 C$(I-1)=C$(I):C$(I)=“”
930 D$(I-1)=D$(I):D$(I)=“”
935 GOTO900
1000 CLS:X=0:FORI=1TO30:PRINTB$(I);
1002 IFN$(I)=“”THENI=30:GOTO1060
1005 FORJ=1TO20:PRINTTAB(20);P$(I,J)
1007 IFP$(I,J)=“”THENJ=20:GOTO1055
1010 INPUT“Enter Price”;K(X)
1015 INPUT“Correct”;L$
1020 IFL$<>“Y”THEN1010
1025 INPUT“No.  Sold”;L(X)
1030 INPUT“Correct”;L$
1035 IFL$<>“Y”THEN1025
1040 X=X+1:INPUT“More Prices this Program”;L$
1045 IFL$=“Y”THEN1010
1050 K(X)=0:L(X)=0:X=X+1:NEXTJ
1055 NEXTI
1060 GOTO50
1200 DOKE3187,1918:IFPEEK(1910)=0THENDOKE3187,1912
1205 CLS:WIDTH80:INPUT“Enter Date”;M$
1210 PRINT:INPUT“Correct”;L$:IFL$<>”Y”THEN1205
```

```
1215 X=0:FORI=1TO30
1216 IFN$(I)=“”THENI=30:GOTO1290
1217 F=0:PRINT:PRINT:PRINTN$(I)
1220 PRINTA$(I):PRINTB$(I):PRINTC$(I):PRINTD$(I):GOSUB1350
1230 FORJ=1TO20:PRINTP$(I,J);:E=0
1235 IFP$(I,J)=“”THENJ=20:GOTO1270
1240 PRINTTAB(20):K(X);
1245 D=INT(100*((K(X)*.2)+.005))/100
1250 PRINTTAB(30);D;TAB(40);L(X):TAB(50):D*L(X)
1255 E=E+D*L(X)
1260 X=X+1:IFK(X)<>0THEN1240
1265 PRINT:PRINTTAB(60):E:PRINT:F=F+E:NEXTJ
1270 PRINT:PRINT:PRINTTAB(50);“Total £    ”;F
1275 INPUT“Change paper for New Author”;L$
1280 IFL$<>“GO”THEN1275
1285 NEXTI
1290 DOKE3187,1919:IFPEEK(1910)=0THENDOKE3187,1913
1295 GOTO50
1350 PRINTTAB(25):CHR$(14);“ROYALTIES PAYMENTS”
1355 PRINTTAB(15);“for the Quarter ended ”;M$
1360 PRINTTAB(5)“PROGRAM”TAB(17)”PRICE”;
1365 PRINTTAB(27)“ROYALTY”TAB(37)”NO.  SOLD”
1370 PRINT:RETURN
1400 DOKE3187,1918:IFPEEK(1910)=0THENDOKE3187,1912
1405 CLS:WIDTH80:INPUT“Enter Date”;M$
1410 PRINT:INPUT“Correct”;L$:IFL$<>“Y”THEN1405
1415 X=0:G=0:FOR1TO30
1420 IFN$(I)=“”THENI=30:GOTO1475
1425 GOSUB1350:F=0:PRINT:PRINT:PRINTN$(I)
1430 FORJ=1TO20:PRINTP$(I,J);:E=0
1435 IFP$(I,J)=“”THENJ=2-:GOTO1470
1440 PRINTTAB(20):K(X);
1445 D=INT(100*((K(X)*.2)+.005))/100
1450 PRINTTAB(30);D;TAB(40);L(X);TAB(50);D*L(X)
1455 E=E+D*L(X)
1460 X=X+1:IFK(X)<>0THEN1440
1465 PRINT:PRINTTAB(60);E:PRINT:F=F+E:NEXTJ
1470 G=G+F:NEXTI
1475 PRINT:PRINT:PRINT:PRINTTAB(30)“Total this micro £   ”;G
1480 DOKE3187,1919:IFPEEK(1910)=0THENDOKE3187,1913
1485 GOTO50
1600 INPUT“Start Tape & Press ENTER”;L$
1605 TC$=“OPEN: IN(‘AUTH,1,6395)”:TR=USR(0)
1610 FORI=1TO30
1615 TC$=“READ(N$(I),A$(I),B$(I),C$(I),D$(I))”
1620 TR=USR(0):NEXT
1625 FORI=1TO30:FORJ=1TO20
1630 TC$=“READ(P$(I,J))”:TR=USR(0):NEXTJ,I
1635 TC$=“CLOSE: IN”:TR=USR(0):GOTO50
1800 INPUT“Start Tape & Press ENTER”;L$
1805 TC$=“OPEN: OUT(‘AUTH’,1,6395)”:TR=USR(0)
1810 FORI=1TO30
1815 TC$=“WRITE(N$(I),A$(I),B$(I),C$(I),D$(I))”
1820 TR=USR(0):NEXT
1825 FORI=1TO30:FORJ=1TO20
1830 TC$=“WRITE(P$(I,J))”:TR=USR(0)
1835 NEXTJ,I
1840 TC$=“CLOSE:OUT”:TR=USR(0):GOTO50
```

EXPANDING THE KEYBOARD OF THE NASCOM

USING NORMAL KEYSWITCHES

by J. M. H. Hill

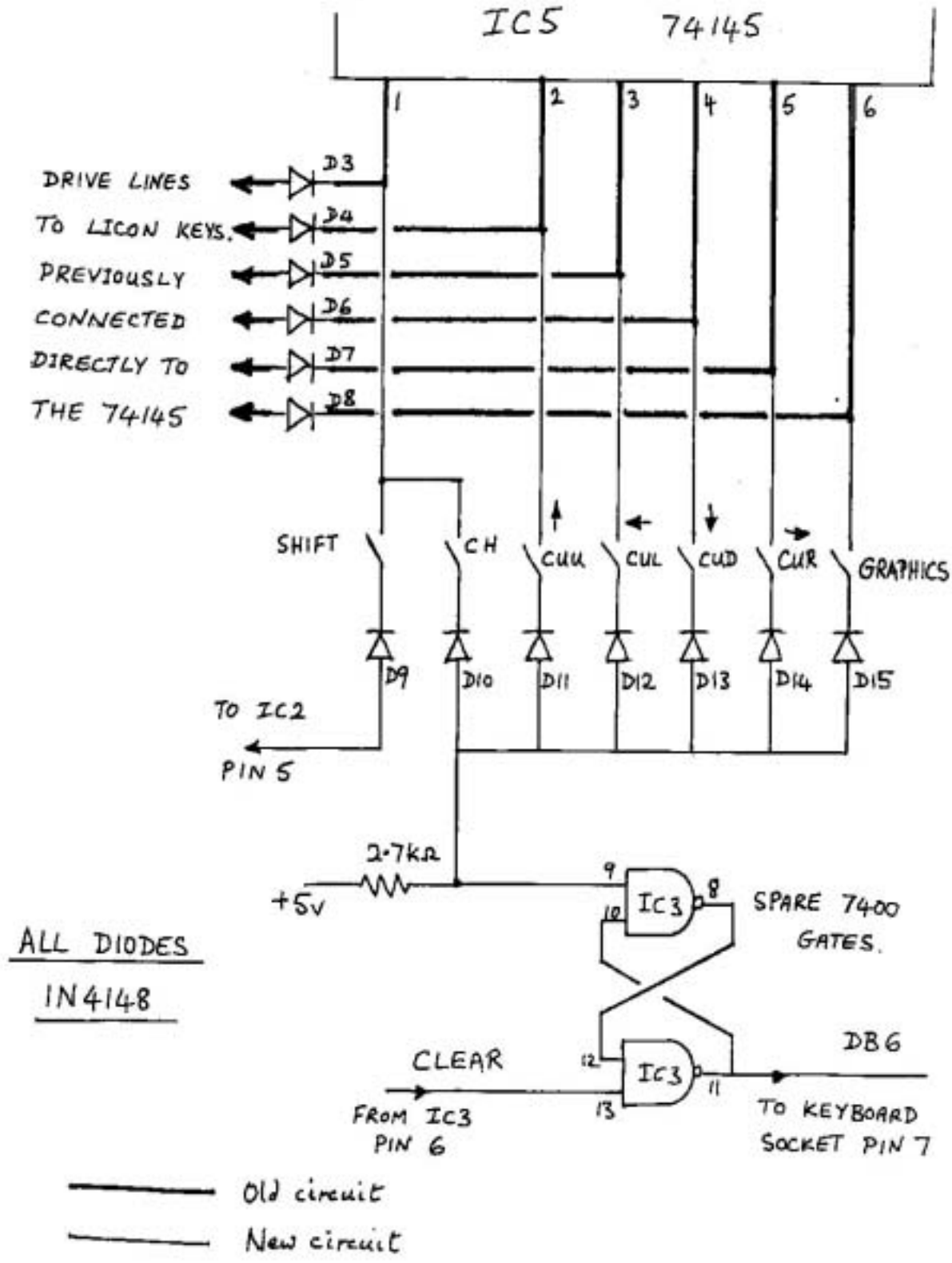
An article was published in issue number 2 of Micropower on the expansion of the Nascom 1 keyboard. Now although the keyboard uses special Licon keys, it is possible to expand it using ordinary push-to-make keyswitches. I have been using such an expanded keyboard for about a year without any problems, and there seems to be no reason why the principle should not be extended further if desired to cover a separate numerical keypad.

The only modifications to the existing keyboard conductors is the cutting of the tracks running between the open collector outputs of IC5 and the original key matrix, to allow the insertion of the isolating diodes D3 - D8. The presence of these siodes does not affect the operation of the original keys. Their purpose is to prevent the voltages of the inactive drive lines from being affected via the Licon key windings by others which are active. A more ambitious expansion would also involve the other two outputs from IC5 on pins 7 and 9, which would also need to be fitted with isolating diodes.

To duplicate any existing key, as in the case of the SHIFT key shown in the circuit, it is necessary to connect the right drive line output from IC5 via a keyswitch and diode (D9) to the input of the appropriate RS flip-flop. The connections are shown for the SHIFT key, but others can be worked out by examining the Nascom 1 keyboard circuit diagram. A further locking keyswitch could be connected in parallel with the SHIFT key to give a shift lock if required. Such a shift lock could be added to the standard Nascom 2 keyboard to provide the facility requested by Mr. R. C. Taylor in issue 2.

Duplication of the standard keys will work with any monitor, but the remaining keys shown in the diagram are for use with Nas-Sys 1 or 3. They give most of the facilities of the Nascom 2 keyboard, inluding single key cursor movement. The GRAPHICS key will of course only work if a suitable graphics unit has been fitted.

The standard Nascom 1 keyboard only uses the lowest six bits of port 0. To handle the extra keys a sense line using bit 6 is needed, plus an extra flip-flop. Fortunately, there are two unused gates in IC3 which can be used for this purpose, as shown in the diagram. An extra wire will be needed to connect the output from the new flip-flop to pin 7 of the keyboard socket on the main board.



NASCOM 1. KEYBOARD EXPANSION.

Nascom & Gemini USERS

NEW 32K C.M.O.S. BATTERY BACKED RAM BOARD

FEATURES:

EFFECTIVELY REPLACES EPROMS.
Does away with the inconvenience of EPROM programming and the compromise of assigning valuable address space to ROM.

ON BOARD RE-CHARGEABLE Ni-Cad BATTERY RETAINS MEMORY FOR OVER 1000 Hrs.
Battery is automatically charged during power-up periods.

HIGH SPEED OPERATION up to 6 MHz WITHOUT WAIT-STATES.

FULLY NASBUS[1] and GEMINI-80 BUS[2] COMPATIBLE.

PAGE MODE SCHEME SUPPORTED.
The board can be configured to provide one 32k byte page or two completely independent 16k byte pages.
Complete pages of 64k bytes are simply implemented by adding more boards on to the bus.

SOFTWARE and/or HARDWARE READ/WRITE PROTECTION.
4K blocks in either page are link selectable to be aligned on any 4K boundary.

FULLY BUFFERED ADDRESS, DATA AND CONTROL SIGNALS.

MEMORY I.C. SOCKETS ARE LINK SELECTABLE TO SUPPORT ANY 24 PIN 2k byte MEMORY I.C.s.
Thus the board can support up to 32k bytes of any mixture of cmos, nmos rams or 2716/2516 eproms.
All options are link selectable using wire links plugged into gold-plated socket pins, avoiding the risk of damage and the inconvenience caused by soldering links directly to the board.
The printed circuit board is manufactured to the high quality demanded by industrial users and conforms to B.S.9000.
The board comes assembled and tested and is supplied with a minimum of 2k bytes of low-power cmos ram. Fully documented.

AVAILABLE NOW!

PRICES:

Board with 2k bytes	£80.00
Board with 16k bytes	£120.00
Board with 32k bytes	£170.00
Bare Boards (circuit diagram supplied)	£17.50
HM6116-LP/3 Very low power 2k cmos memory I.C.	£6.50

Please add 95p P&P and VAT @ 15%

[1]nasbus is a trademark of nascom microcomputers a division of LUCAS LOGIC
[2]trademark of GEMINI MICROCOMPUTERS LIMITED

cheques and P.O.s to:

Processes 41a MOOR LANE, CLITHEROE, LANCS. BB7 1BE. For further information Phone (0200) 27896

Please supply me with the following:

Price

Total enclosed £

Address:

POST CODE

io research ltd.

"PLUTO" COLOUR GRAPHICS PROCESSOR

Pluto is a self-contained colour display processor on an 8" x 8" NASBUS and 80-BUS compatible card featuring:
- Own 16 bit microprocessor
- Up to 192 Kbytes of dual-ported display memory for fast flicker-free screen updates. (Outside of the host address space).
- 640(H) x 288(V) x 3 planes (8 colours) – 2 screenfulls OR 640(H) x 576(V) x 3 planes (interlaced display)
- Fast parallel I/O interface usable with ALMOST ANY MICRO. Only single +5v supply required.

Pluto executes on-board firmware providing high level functions such as:
- Fast vector draw – over 100,000 pixels/sec. Lines can be drawn using REPLACE, XOR, AND, OR functions
- User-definable characters or symbols
- Spare display memory with memory management facilities for allocating symbol storage space or workspace
- Rectangle Fill and copy using REPLACE, XOR, AND, OR plus 5 other functions
- Fast access to single pixels
- Write protect memory planes during copy
- Double-buffered screen memory for animated displays
- Complex polygon colour fill

Pluto is expandable. An expansion board will be available later this year to give Pluto up to 8 memory planes with no loss of resolution.

AVAILABLE NOW. Please contact us for further details

6 Laleham Avenue, Mill Hill,
London NW7 3HL
Tel: 01-959 0106

io research ltd.

A/D BOARD FOR NASCOM
- 8 input channels
- 30 microsec conversion
- Over voltage protection
- Prototyping area
- 8 bit resolution
- Sample and hold
- Full list/interrupt control
- NASBUS compatible

Price £120 + 15% VAT (post free)

GRAPHICS BOARD FOR NASCOM
- 384(H) x 256(V) high resolution graphics display
- Fully bit mapped
- Full software control
- Mixed text and graphics
- NASCOM 2 or 4MHz NASCOM 1
- Graphics software supplied

Price £55 + 15% VAT (post free)

EPROM PROGRAMMER
- Programs 3 rail: 2708/2716
  Single rail: 2508/2758, 2516/2716, 2532/2732
- Software supplied for Read/Program/Verify
- Can be used with other machines with 2 parallel ports

Price £63 + 15% VAT (post free)

DUNCAN
- Fast real time interpreter/control language for NASCOM 1 or 2 (please specify)

Price £12 + 15% VAT (post free)

MEMORIES
- 4116-150ns **95p each + 15% VAT (min order 8)**
- 64K-200ns **£6 each + 15% VAT**

MONITORS
- BMC 12" green phosphor – 18MHz

Price £150 + 15% VAT (carriage paid)

6 Laleham Avenue, Mill Hill,
London NW7 3HL
Tel: 01-959 0106

64K RAM ON A NASCOM RAM B BOARD

by Douglas M. Barr

This article is aimed at the owner of a Nascom RAM-B board who wants to increase the available RAM to the full 64K that the Z-80 microprocessor can address. There may well be something of interest for readers who are considering the modification of other boards populated with 4116 dynamic RAM chips; be warned, however, that although the modified board will work on a Nascom 2 or a Gemini multiboard system, it will NOT work on a Nascom 1 as described. The modification uses the MEXT line, whih is essential to the correct operation of a Nascom 1. That said, there is no reason why some enterprising Nascom 1 owner should not circumvent this problem and find some suitable alternative way of routing what we shall call the 'RAM BLK 3' signal on the board

One of the very attractive features of Nascom products so far has been the literature that the firm has published with their hardware, and I am assuming that you have the circuit diagrams which came with the board when you bought it. In particular, I shall be referring to figures 10, 11 and 12, which are the circuit diagrams issued with the manufacturer's instructions for the board, but it should be possible to follow the gist of the theory behind the modification from this article alone. If you are interested only in the practical aspects of 'how to do it', then the article should be self explanatory.

The modification should not be beyond the skill of the average Nascom owner, but I am well aware that for some who are new to the game, what I am proposing may sound a bit daunting. For their benefit I shall take things rather slowly, and I ask any old hands to bear with me if some of what I say is 'old hat' to them. After you have completed the modifications you will have a board which will support 64K of RAM at 4 Mhz without wait states, will retain the ability to 'write protect' the banks of 4116s, and will also retain the Nascom page mode of operation. For several months I have been running such a modified board on my Nascom 2, and more recently I have used it also on a Gemini Multiboard system. What is interesting is that I used 200 ns RAMs, apparently without any adverse effects, but if you add up all the gate and delay-line times, you will realise that there is probably a certain element of luck in getting the board to run at 4 Mhz, and that the RAMs may be running a bit faster than spec. Those of a weak disposition may consider the use of 150 ns RAMs if they cannot bear the thought of the memory dumping at an inopportune moment.

The modification involves soldering sockets or chips on top of other delicate chips - with all the risks that entails. Many argue that the best way to do this is to wrap a couple of layers of cooking foil around a spare piece of polystyrene tile and plug the chip into the foil. This does two things; it 'commons' all the pins of the chip, and it helps conduct away heat and thereby reduces the risk of damage to delicate connectors inside the DIL package. If you are short of space inside your Nascom and there is not too much space between the boards, you may have to solder the

upper chips directly on top of the lower chips, but otherwise you would be well advised to solder DIL sockets for the upper layer onto the chips of the lower layer, as only one chip is put at hazard, and you will retain the option to remove the top chip easily in future. Before soldering on the top chip/socket, wrap it in a couple of layers of foil as well, with the pins protruding through the foil. Again, the foil acts as an insulator and heat sink.

One unfortunate by product of this 'recommended' method is that when you have finished you have to dig bits of cooking foil from between the pins of your newly mated chips, and you may prefer to make yourself a Jig with a spare DIL socket which has all its pins commoned in some scrap veroboard. One last thought on piggybacking ICs; it is a good idea to make sure that pin 1 is clearly recognisable from below on the bottom IC, otherwise just as you finish a beautifully neat soldering Job, you realise that you can't remember having checked the orientation of the bottom chip. If the end of the 4116 is not clearly notched, lightly scratch the underside of the IC next to pin 1, or mark it with a small dot of marker dye; and do this before you start any soldering.

Now for a quick look at the theory behind the modification. Fig. 12 of the instructions which accompany the RAM-B board show the memory ICs arranged in three banks of eight, with addresses commoned from the right of the diagram, and data commoned vertically. On the left there are essentially three lines to each bank; a column address strobe (CAS) which is common to all three banks, a unique row address strobe (RASx) to each bank, and a write strobe (WRx), again unique to each bank. The logic behind the latter two runs something like this. Each bank MUST be individually addressed and if you want to create another bank then you MUST create an additional RASx signal from somewhere. Since the existing banks are numbered 0 - 2, with their associated RAS0 - RAS2 signals, it makes sense to number the new bank 3, and call the new signal RAS3. The third signal is the WRx signal, which must be taken low when whenever you (the CPU really!) want to write to the respective bank. If the WRx signal is held high, bank x will become write protected, and if the relevant pins of the upper bank are soldered to those of the lower bank, the upper bank will take the same write protect status as the lower. At this stage, let me state that it is possible to arrange for the new bank of RAMs to have independent write protect , but to be frank, although I have run my board in this condition, I have never found the need for separate write protect on the new bank of RAMs, and in the end I Just removed the additional wiring as it was a bit of an eyesore. For this reason, I shall not cover this aspect of the modification, but if any reader is interested in the details, please contact me through this magazine.

From what I have said, it should be obvious that all 16 pins of the upper and lower 4116s can be commoned (is there really such a word in English?) except for pins 4, which must be separated between banks but comoned along the new bank.

We are now in a position to start piggybacking the RAM chips. Pin 4 of every socket should be bent out at right angles to the socket. The remaining pins should be soldered using the smallest bit available. As you complete each 4116, it is a good idea to put it back into the board and test it. (If you are soldering chips directly onto chips, you must take pin 4 of the top IC to +5 volts when testing. lf the pin is allowed to float, neither IC will work correctly, and the memory test will fail.) When all eight chips have been piggybacked all the pn 4s should be commoned by connecting them to a wire link running along the bank. As a further check the top ICs can now be inserted and if the linking wire is pulled to +5 volts through a suitable resistor (I found that 330 ohms was O.K.) the board should function normally as a 48K board at this stage.

Now that we have the fourth bank of chips mounted successfully on top of one of the other banks, and the board is working normally, we can start looking for the extra logic signals that are needed. If you look at fig. 10 of the RAM-B instructions you will see SK1 where the DIL header plug (4K BLOCK DECODE) is inserted at the top left of the diagram. Pin 20 is labelled NAS1 SEL and goes direct to MEXT on line 11 of Nasbus. If you are working with Nascom 2 or Multiboard, this signal is not required, and pin 20 of SK1 provides an ideal point for the connection of the RAM BLK3 signal. Cut the MEXT line on the solder side of the board where it runs from the plated through hole opposite pin 11 of IC43 to the hole opposite C52. I strongly advise that you check that you have the correct line by using a multimeter or some other form of LOW VOLTAGE continuity tester before you do any cutting.

You will see from diagram 10 that RAM BLK0/1/2 signals go to pins 4/2/1 of IC35. We need to connect pin 20 of SK1 to pin 5 of IC35. This pin is connected to +5 volts by a wide track which is hidden by the socket, so the simplest way to make the connection is to remove the IC from the socket and bend pin 5 at right angles so that it remains clear of the socket when reinserted. A wire should be soldered on this pin and taken to intercept the line from pin 20 of SK1. Look at the area on the component side of the board between IC35, IC36, RP5 and IC42; there are two through plated holes in this area, and the one nearer to IC35 is on the MEXT line and connected to pin 20 of IC35. The wire from pin 5 of IC35 should be connected to this point. Once again, check carefully that you have the correct point.

The RAM BLK signal lines are pulled up to +5 volts by 1Kohm resistors R7, R8 and R9. RAM bLK3 must be pulled up by a similar resistor. A convenient place to do this is beside SK1, where there are two plated through holes next to the silk-screened numbers 7 and 9; the one near the 7 connects to pin 20 of SK1, and the one near the 9 is at +5 volts. The MEXT line is already pulled up to +5 volts by a 4K7, so to achieve a pull-up of 1 Kohm the resistor used should be 1.2 Kohms.

The RAM BLK 0 - 2 signals are taken to pins 2, 5 and 10 of IC36, where they are ANDed with the RFSHB signal, taken to pins 1, 4 and 9. Another gate is required to support the RAM BLK 3 signal. Two spare gates are available on IC41, which is also a 74LS00. Pins 12 and 13 of IC41 are connected to +5 volts by a wide track. After double checking, carefully cut this track to isolate these pins. Join pin 13 of IC41 to the new RAM BLK 3 line using light hookup wire, or Verowire; the best point to join the wire is at pin 10 of RP5. The RFSHB signal can be tapped from the plated through hole beside pins 10 and 11 of IC36; it should be connected to pin 12 of IC41.

If you hold the board up to a strong light, and look at a point halfwasy between pins 8 and 9 of IC35 and pins 6 and 7 of IC36 you will see that there is quite a large area where there are no tracks on either side of the board. Mark the centre of the clear area and, after rechecking, drill a hole through the board. The hole only needs to be large enough to permit hookup wire to pass through. One end of this wire should be soldered to pin 11 of IC41, and the other end of the wire is passed through the hole to the component side of the board where it passes between IC36 and C61 and is soldered to a point vertically above pin 1 of IC37.

We now have to find a new OR gate to generate a RAS signal for the exta memory bank, and we do this by piggybacking a new 74LS32 onto IC37. The only signals needed from IC37 for the new IC are +5 volts (pin 14), 0 volts (pin 7), and RAS, which is at pins 2, 5 and 10. For purey mechanical reasons, pin 2 is used, as it spaces the mounting pins evenly.

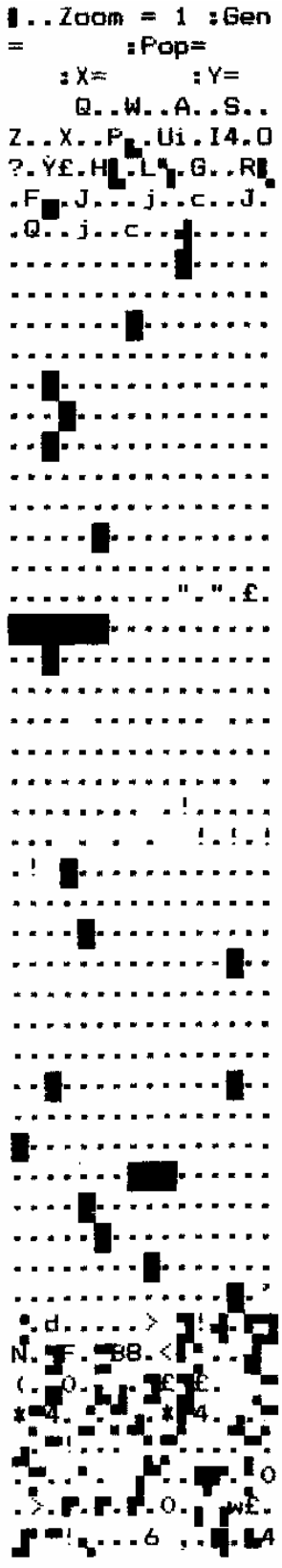
Now make up a DIL socket for the new IC. This requires some ‘nice’ soldering, and I would strongly advise against soldering the new IC directly on top of IC37. Take a 14 pin DIL socket and break off pins 6, 8 and 11. Solder pins 2, 7 and 14 to the 74LS32; a touch of instant glue between the IC and socket is recommended. Carefully bend out pins 1, 3, 4, 5, 9, 10, 12 and 13. Pins 4 and 5 should be taken low by linking them to pin 7. Similarly, pins 9, 10, 12 and 13 should be taken to +5 volts by linking them to pin 14. Insert the joined up IC and DIL socket, and link the wire from pin 11 of IC41 to pin 1 of the socket. Next soldedr a hookup wire to pin 3 of the socket, and connect this through a 33 Ohm resistor to the wire which links the pin 4s on the new memory bank. Plug a 74LS32 into the socket, check the board thoroughly, and when satisfied insert the board in the Nascom and check for normal operation. If all is well, you can rewire the DIL header plug that fits in SK1; the only difference from the addressing shown in the RAM-B instructions is that pin 20 is now used to decode the new memory bank. You should now have a RAM-B board which will run 64K of dynamic RAM at 4 Mhz.

COORDINATE LIFE

By P. Whittaker

The program plays ‘Life’ on a 10000x1000 array, running under Nas-Sys 1 or 3. Start by E1000. I shall give a brief description of the program in the next issue of the magazine.

```
1000  C3 03 15 5A 6F 6F 6D 20 3D 20 31 20 3A 47 65 6E
1010  3D 20 20 20 20 20 20 3A 50 6F 70 3D 20 20 20 20
1020  20 20 20 3A 58 3D 20 20 20 20 20 3A 59 3D 20 20
1030  20 20 20 20 51 00 02 57 01 02 41 00 01 53 01 01
1040  5A 00 00 58 01 00 50 E6 15 55 69 19 49 34 1A 4F
1050  3F 1A 59 23 19 48 E7 16 4C F1 16 47 09 17 52 E3
1060  18 46 BC 18 4A 03 15 11 6A 15 12 63 15 13 4A 15
1070  14 51 15 15 6A 15 16 63 15 00 AE 16 00 00 01 00
1080  01 02 03 01 04 00 05 00 06 00 FF 01 03 02 01 02
1090  03 03 02 03 03 12 08 12 09 12 0A 13 08 13 0B 14
10A0  08 15 08 16 09 16 0B FF 00 0F 0B 00 0B 01 01 0E
10B0  01 10 0C 01 0C 02 02 0E 02 11 0D 00 03 0F 03 11
10C0  04 10 FF 00 00 00 01 00 02 01 02 02 02 03 00 03
10D0  01 05 00 FF 00 02 00 03 01 03 02 00 02 01 02 02
10E0  03 00 FF 00 00 00 01 00 02 00 06 00 07 00 08 01
10F0  00 01 03 01 05 01 08 02 00 02 08 03 00 03 08 04
1100  01 04 03 04 05 04 07 06 04 07 03 07 04 07 05 08
1110  03 08 04 08 05 FF 00 01 00 02 01 01 04 00 05 01
1120  06 00 07 01 08 00 08 01 09 02 19 02 19 04 1A 02
1130  1A 05 1B 02 1B 04 1C 03 1D 03 22 03 22 04 23 03
1140  FF FF FF FF FF FF 00 02 01 02 02 02 03 00 03 01
1150  04 00 FF 00 00 00 01 00 10 00 11 00 12 01 00 01
1160  01 01 0E 01 12 02 0D 02 12 03 0D 03 11 05 0D 05
1170  11 05 1F 05 20 06 0D 06 12 06 1F 06 20 07 00 07
1180  01 07 0E 07 12 08 00 08 01 08 10 08 11 08 12 0E
1190  19 0E 1F 0F 18 0F 19 0F 1A 0F 1E 0F 1F 0F 20 10
11A0  17 10 18 10 1A 10 1E 10 20 10 21 13 1A 13 1E 14
11B0  1A 14 1E 20 18 20 19 20 1F 20 20 21 18 21 19 21
11C0  1F 21 20 FF 00 03 00 04 01 02 01 05 02 01 02 06
11D0  03 00 03 07 04 00 04 07 05 01 05 06 06 02 06 05
11E0  07 03 07 04 FF 00 01 01 01 02 00 02 02 03 01 04
11F0  01 05 01 06 01 07 00 07 02 08 01 09 01 FF 00 02
1200  00 03 01 02 01 03 08 03 09 02 09 03 09 04 0A 01
1210  0A 02 0A 03 0A 04 0A 05 0B 00 0B 01 0B 05 0B 06
1220  0F 02 0F 03 0F 04 10 02 10 03 10 04 13 02 13 03
1230  14 02 FF 00 01 01 00 01 01 01 02 02 02 FF 00 01
1240  00 02 00 03 01 03 02 02 02 03 03 00 03 01 03 02
1250  FF 00 05 01 04 01 06 03 03 03 06 04 03 04 04 05
1260  03 1A 00 1A 01 1B 00 FF FF FF 00 00 01 00 01 02
1270  02 00 02 01 FF 00 01 01 00 02 00 03 00 03 03 04
1280  00 04 01 04 02 FF 00 01 01 00 02 00 03 00 04 00
1290  04 03 05 00 05 01 05 02 FF 00 01 01 00 02 00 03
12A0  00 04 00 05 00 05 03 06 00 06 01 06 02 FF 10 27
12B0  E8 03 64 00 0A 00 01 00 3E 20 FD 21 AE 12 A7 FD
12C0  4E 00 FD 46 01 ED 42 38 04 3C C3 C5 12 09 FE 20
12D0  28 02 F6 30 12 E6 F0 1C FD 23 FD 23 0D 20 E0 C9
12E0  2A CD 34 11 EC 0B CD B8 12 2A CF 34 11 F4 0B CD
12F0  B8 12 C9 21 00 04 0E C0 C5 CD 09 13 06 06 CD 0B
1300  13 CD 09 13 C1 0C 20 F0 C9 06 05 AF CB 09 C5 30
1310  02 3E 0F CB 09 CB 09 CB 09 30 02 C6 F0 77 23 10
1320  FC C1 C9 21 E2 0B 06 07 36 20 2C 10 FB 11 D3 34
```



```
1330  2A D1 34 23 A7 ED 52 44 4D 19 EB ED B0 E5 21 D3
1340  34 36 FF 22 D1 34 E1 C9 21 E8 0B 7E 3D 77 FE 30
1350  38 06 C0 2D A6 2C 77 C9 36 39 2D 18 EE 21 E8 0B
1360  18 03 21 DC 0B 7E F6 30 3C 77 FE 3A C0 36 30 2D
1370  18 F3 21 01 0C CB 66 28 03 87 87 87 21 C8 34 4E
1380  0D 28 03 87 18 FA 4F C9 CD E0 12 21 F5 31 0E 03
1390  3A 50 1A 06 F0 77 23 10 FC 0D 20 F7 3E 7F 32 4F
13A0  1A 21 D1 34 23 23 7E A7 F2 A4 13 57 23 5E 23 E5
13B0  D5 ED 5B CF 34 2A 53 1A 3A C8 34 29 3D 20 FC EB
13C0  A7 ED 52 CB FC D1 A7 EB ED 52 30 04 E1 C3 A4 13
13D0  3A C8 34 3D 28 07 CB 1C CB 1D A7 18 F6 E5 2A 53
13E0  1A 29 29 EB E1 4D ED 52 E1 38 3B 3A 50 1A FE 20
13F0  20 1D 3A 4E 1A D6 7E 28 16 06 00 50 58 17 CB 10
1400  17 CB 10 17 CB 10 17 CB 10 4F 21 F5 29 ED B0 21
1410  F5 31 11 0A 08 06 00 3E 0F 0E 30 ED B0 0E 10 EB
1420  09 EB 3D 20 F4 C9 7E A7 FA AB 13 57 23 5E 23 E5
1430  D5 ED 5B CD 34 2A 51 1A 3A C8 34 29 3D 20 FC EB
1440  A7 ED 52 D1 A7 EB ED 52 30 04 E1 C3 26 14 3A C8
1450  34 3D 28 07 CB 1C CB 1D A7 18 F6 E5 2A 51 1A 29
1460  29 EB E1 45 A7 ED 52 38 04 E1 C3 A6 13 3A 50 1A
1470  FE 20 28 2B 21 C5 34 11 D0 FF 79 3D 19 D6 03 30
1480  FB 1E 04 CB 03 3C 20 FB 78 1F 38 06 CB 0B CB 0B
1490  CB 0B 85 6F 7C CE 00 67 7E B3 77 E1 C3 26 14 19
14A0  C5 E5 AF 47 79 0E 30 21 C5 34 ED 42 D6 10 30 FA
14B0  ED 44 3D 5F C1 79 E6 07 57 CB 18 CB 19 CB 39 CB
14C0  39 09 7E D6 80 30 21 3A 4F 1A 3C 28 31 32 4F 1A
14D0  77 D6 80 6F AF 67 29 29 29 29 01 04 2A 09 06 0F
14E0  77 2B 10 FC 77 C3 F3 14 6F AF 67 29 29 29 29 01
14F0  F5 29 09 42 57 19 3E 80 04 07 10 FD B6 77 C1 E1
1500  C3 26 14 EF 0C 00 AF 32 C5 34 32 C6 34 3C 32 C8
1510  34 3E C0 32 50 1A 21 0B 00 22 53 1A 21 18 00 22
1520  51 1A 3E FF 21 D3 34 77 22 D1 34 21 88 13 22 CD
1530  34 22 CF 34 21 03 10 11 CA 0B 01 30 00 ED B0 CD
1540  62 13 CD E0 12 CD F3 12 18 57 3E FF 1E 02 47 18
1550  06 3E 01 1E 04 06 00 CD 72 13 2A CF 34 09 22 CF
1560  34 18 17 3E FF 1E 05 47 18 06 3E 01 1E 03 06 00
1570  CD 72 13 2A CD 34 09 22 CD 34 16 0C D5 CD 88 13
1580  06 07 FF 10 FD E1 DF 62 CB 76 28 15 3E 0A 32 81
1590  15 7D D6 02 28 B4 3D 28 D1 3D 28 B5 18 C5 CD 88
15A0  13 21 E2 09 4E 16 5F 3E 50 32 81 15 7E 72 57 06
15B0  3C DF 62 00 38 05 FF 10 F8 18 F1 71 32 79 10 21
15C0  32 10 23 23 BE 23 20 FA 5E 23 7E FE 03 38 03 57
15D0  EB E9 4F 16 00 42 2A CD 34 19 EB 2A CF 34 09 44
15E0  4D CD 18 16 18 B8 AF 32 C7 34 CF FE 11 38 FB FE
15F0  15 30 1A E6 07 47 0E 03 B9 30 02 0E 0C 3E 80 07
1600  10 FD 47 3A C7 34 A1 B0 32 C7 34 18 DD E6 0F 32
1610  C5 34 32 C6 34 C3 A1 15 CB F8 21 D2 34 78 3D 18
1620  01 23 23 BE 30 FB 3C BE 38 08 79 23 BE 28 28 30
1630  EC 2B C5 D5 EB 2A D1 34 A7 ED 52 44 4D 03 19 EB
1640  21 04 00 19 22 D1 34 EB ED B8 D1 C1 23 70 23 71
1650  23 72 23 73 C3 5D 13 7A 3D 18 01 23 23 BE 30 FB
1660  3C BE 38 08 7B 23 BE 28 1D 30 EC 2B D5 EB 2A D1
1670  34 A7 ED 52 44 4D 03 19 EB 21 02 00 19 22 D1 34
1680  EB ED B8 D1 18 CA 23 7E E5 2B 2B E6 80 54 5D 28
1690  06 2B 2B A6 28 01 EB 2A D1 34 EB E3 EB ED 52 44
16A0  4D 03 E1 EB ED B0 1B ED 53 D1 34 C3 48 13 CD B4
16B0  16 C3 9E 15 1B 1B 1A 21 7C 10 D6 61 D8 28 0B FE
```

```
16C0  1B D0 47 7E 23 3C 20 FB 10 F9 7E FE FF C8 5F 16
16D0  00 23 4E 23 42 E5 2A CD 34 19 EB 2A CF 34 09 44
16E0  4D CD 18 16 E1 18 E3 3E 20 21 3C 00 11 60 00 18
16F0  0B CD F3 12 21 0B 00 11 18 00 3E C0 32 50 1A ED
1700  53 51 1A 22 53 1A C3 9E 15 DD 21 D3 34 FD 21 D3
1710  34 21 00 00 D9 FD 46 00 FD 4E 01 0B 2A D1 34 22
1720  C9 34 21 CB 34 71 23 70 21 D3 34 E5 FD E5 ED 4B
1730  CB 34 0B 1E 00 DD 7E 00 B8 20 0B DD 7E 01 B9 20
1740  05 1C DD 23 DD 23 03 7E B8 20 09 23 7E 2B B9 20
1750  03 1C 23 23 03 FD 7E 00 B8 20 0C FD 7E 01 B9 20
1760  06 FD 23 FD 23 18 13 7B A7 20 0F FD 7E 00 FE FF
1770  28 61 47 FD 4E 01 FD 23 FD 23 ED 43 CB 34 E5 0B
1780  2A C9 34 23 70 23 71 22 C9 34 03 E1 DD 46 00 DD
1790  4E 01 7E B8 28 15 30 04 47 23 4E 2B FD 7E 00 B8
17A0  28 11 30 15 47 FD 4E 01 C3 B9 17 23 7E B9 38 EA
17B0  C3 9B 17 FD 7E 01 B9 38 EC 78 07 D2 58 18 2A C9
17C0  34 2B CB 7E 28 04 2B 22 C9 34 3C 28 06 E1 DD E1
17D0  C3 2B 17 2A C9 34 ED 5B D1 34 A7 ED 52 28 61 44
17E0  4D EB 23 11 D3 34 ED B0 EB 36 FF 22 D1 34 E1 E1
17F0  CD 8B 13 CD 62 13 D9 11 E4 0B CD B8 12 21 08 0C
1800  DF 62 CB 66 C2 A1 15 3A C6 34 A7 CA 09 17 3D 32
1810  C6 34 C2 09 17 3A C5 34 32 C6 34 3A C7 34 2A CD
1820  34 0F 30 01 2B 0F 30 01 23 22 CD 34 2A CF 34 0F
1830  30 01 23 0F 30 01 2B 22 CF 34 CD E0 12 C3 09 17
1840  21 4F 18 11 5E 09 01 09 00 ED B0 CF C3 03 15 44
1850  49 45 53 20 4F 55 54 21 AF 57 08 1E 00 79 DD BE
1860  01 20 0C 78 DD BE 00 20 05 DD 23 DD 23 1C 79 23
1870  BE 2B 20 0A 78 BE 20 05 23 23 1C CB FB 79 FD BE
1880  01 20 0B 78 FD BE 00 20 05 FD 23 FD 23 1C 08 82
1890  83 CA 8C 17 E6 0F FE 03 28 0E FE 04 28 06 7A 53
18A0  03 C3 5A 18 CB 7A 28 F6 E5 2A C9 34 23 0B 70 23
18B0  71 22 C9 34 E1 03 D9 23 D9 C3 9E 18 CD 23 13 7E
18C0  CB 7F 28 0A FE FF CA 9E 15 47 23 4E 23 7E 57 23
18D0  5E 23 E5 C5 2A CD 34 29 A7 ED 52 EB CD 18 16 C1
18E0  E1 18 DC CD 23 13 7E CB 7F 28 1C FE FF CA 9E 15
18F0  E6 7F 57 23 5E 23 E5 2A CF 34 EB A7 ED 52 ED 5B
1900  CD 34 19 EB E1 18 DF 47 23 4E 23 E5 D5 ED 5B CD
1910  34 2A CF 34 A7 ED 42 19 44 4D D1 D5 CD 18 16 D1
1920  E1 18 C3 2A CF 34 11 05 00 ED 52 44 4D 2A CD 34
1930  ED 52 EB 26 0A D5 2E 0A 3E FF CD 56 19 FE 14 30
1940  09 E5 D5 C5 CD 18 16 C1 D1 E1 13 2D 20 EA D1 03
1950  25 20 E2 C3 9E 15 C5 E5 21 CB 34 47 ED 5F 86 CE
1960  FF 90 30 FD 80 3C E1 C1 C9 CF FE 43 CA 0A 1A FE
1970  57 CA 13 1A FE 52 CA 24 1A FE 53 28 41 D6 30 DA
1980  A1 15 FE 0A D2 A1 15 3C 47 21 54 1A 23 E5 DD E1
1990  DD 66 05 DD 6E 04 11 D3 34 A7 ED 52 EB DD E5 E1
19A0  19 11 06 00 19 7E FE FF C2 A1 15 10 DF DD E5 D1
19B0  ED 52 44 4D 03 21 CD 34 EB ED B0 C3 9E 15 06 0A
19C0  21 54 1A 23 E5 DD E1 DD 66 05 DD 6E 04 11 D3 34
19D0  A7 ED 52 EB DD E5 E1 19 11 06 00 19 7E FE FF C2
19E0  E7 19 10 DF C3 A1 15 2A D1 34 11 CC 34 A7 ED 52
19F0  44 4D DD E5 D1 62 6B 09 D5 11 F5 29 ED 52 D1 D2
1A00  A1 15 21 CD 34 ED B0 C3 03 15 21 00 00 22 59 1A
1A10  C3 A1 15 21 55 1A 22 0C 0C 21 F5 29 22 0E 0C DF
1A20  57 C3 A1 15 21 00 00 22 0C 0C 3E 52 32 2B 0C DF
1A30  52 C3 A1 15 3A C8 34 3C FE 06 38 09 C3 9E 15 3A
1A40  C8 34 3D 28 F7 32 C8 34 F6 30 32 D1 0B 18 ED 44
```

PROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAM

"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 right 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 prove 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 the other magical beings. Beware also the Vampire Bats who will sap your strength requiring you to find a life-giving Elixir, and the SPIDER GODS whose attentions are usually fatal !

****

The program requires 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 & again. (Nascom BASIC/Graphics).

Send NOW for this excellent program. Only £8.95

Please add 55p/order P & P + VAT @ 15%. Large (15½p) SAE for FULL CATALOGUE.

PROGRAM POWER

PROGRAM POWER
5, Wensley Road,
Leeds LS7 2LX.

PROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAM

PROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAM

NASCOM 1&2 OWNERS!

ELDORADO GOLD

MIN. 16 K/G.

INTRODUCING A GREAT NEW M/C – CODE ADVENTURE FROM PROGRAM POWER

Welcome to a past age, when cowboys ruled the roost and Indians terrorised the new settlers.

In ELDORADO COUNTY many fortunes were made – and nearly as many lost! Legend has it that Old Bill McCluskey, who met a rather sudden death, had built up a vast treasure somewhere in the nearby territory.

Can you end up a rich man where many have failed? If the marauding Apaches or Big Jake's gang don't get you, you may die of thirst in the desert or starve to death in jail.

Begin your Western Adventure NOW – ONLY £7.95

Please add 55p order P & P + VAT at 15%

Send SAE for our Full Catalogue

PROGRAM POWER
5 Wensley Road
Leeds LS7 2LX
Tel. (0532) 683186

PROGRAM POWER

PROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAM

PROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAMPOWER

NASCOM 1&2

PROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAMPOWER

NASPRINT 80
2K extension to NASPEN. New functions incl. Pagination, page no. & title output, title centering & text re-formatting with embedded control codes-change line length, spacing & margins; centre line between margins;-new page; output printer control codes. Contains parallel printer routine for MX80. Use with serial or other parallel catered for. Special routines facilitate operation with ZEAP, NAS-DIS, DEBUG, NAS-SYS & ROM BASIC. In Eproms £14.95

NASCOM 1
Cottis Blandford cassette interface for N2 format, reliability & fast load £14.90

AY-3-8910 SOUND CHIP
INVASION EARTH with INCREDIBLE SOUND EFFECTS (MC/G) £10.95
SOUND CHIP—Program up to 3 independent channels, with music & sound effects! Data sheet incl. £6.45
SOUND CHIP INTERFACE BOARD—Designed to interface between the PIO & the chip. Ready built—plugs straight onto PIO. Nascom 1 connectors available. Sound generation illustrated in MC. & Basic. (chip not incl.) £13.50

DEMO PROGRAM (MC) 1st mode—direct entry to chip register making experimentation simple. 2nd mode—turns keyboard into 7 octave 'piano', giving state of registers & notes played £5.95
DATA MANUAL (80 pages) No VAT £2.25

BASIC FILE HANDLER (MC)
for cassette-based systems.
PAYROLL, SALES & PURCHASE LEDGERS, PRICE LISTS etc.—NOW you can write them! Save complex data files on cassette—any combination of strings, string variables, string arrays, constants, expressions, variables or arrays. Definable block size. At 2400 BAUD using 1K blocks, 1000 numbers can be stored/accessed in less than 1 min. Comprehensive manual & circuit for optional automatic cassette drive control supplied £17.50

"MICRO-POWER"-Magazine
Series Article on hardware & software news from clubs, letters point of views, questions & answers. Now available issues 1-4 £3.95 (Europe £6.60) & overseas £7.90. 1982 Subscription (six issues £5.95), (Europe £9.90, overseas £11.80).

THE KEYS OF KRAAL (24K/B/G)
Superb adventure game PLUS exciting graphics. Fight the monsters & demons in real time. Swords flash, arrows fly & spells home-in. Endless hours of enjoyment. Save on tape £8.95

THE NEW SPACE ERA
MOON RAIDER (MC/G)—'Scramble' type game with superb SOUND effects through Keyboard Port. Raid enemy dumps, blast asteroids & defending ram ships. Limited fuel, 4 skill levels £8.95
STARSHIP COMMAND (16K/B/G)—3D galaxy (8x8x8). 6 levels from Commodore to Space Marshal. Excellent combination of galactic search and M/C code battles £9.95
JAILBREAK IN SPACE (16K/MC/G)—Fast arcade game. Beat off the alien rescue ships attacking your jail. Hold onto your high security prisoners as long as possible £8.95
NEW FASE (16K/MC/G)—Action packed invaders prog. 4 different enemy fleets, individual motion, direct & 'truly' angled missiles £7.95

GRAPHIC GOLF (16K/B/G)
18 hole course, driver, 9 irons & putter, variable wind speed & direction, swing control, super graphics. Beat the par 72 £7.95

8K RAM required unless otherwise stated.
Please state if Nascom TAPE Basic required.
ALL PROGRAMS SUPPLIED ON CASSETTE IN CUTS/KANSAS CITY FORMAT.

Please add 55p/order P & P + VAT @ 15%. Large (15½p) SAE for FULL CATALOGUE.

PROGRAM POWER
5, Wensley Road,
Leeds LS7 2LX.

PROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAMPOWERPROGRAM