Lynx User 1

CAMPUTERS
LYNX USER
The magazine for LYNX micro owners

INSIDE THIS ISSUE

JUNE 1983

96K LYNX SET FOR JULY LAUNCH

The first in a planned series of expansions for the Lynx microcomputer will become available from the beginning of July. Stocks of the 96K version of the machine will be on sale at Lasky's, Spectrum and reputable computer shops from that date.

Outwardly the 96K Lynx will be identical to the 48K machine, but the grey casing will sport smart new badging.

The main changes affect the machine's internal components. The addition of new RAM chips means that user workspace available to Basic in high resolution colour will rise to 37.5K. Considerably more memory can be accessed through machine code.

Printers

Software to drive both parallel and serial printers will be stored on ROM in the 96K Lynx, although it will still be necessary to buy a lead for use with serial printers and an interface pack to control parallel printers. The lead is scheduled to go on sale in July, priced at £3.99. The interface pack for use with parallel printers will be available in August with a price tag of £49.95.

The 96K Lynx will retail through good computer outlets for £299 including VAT.

Like its forerunner, the 48K Lynx, the new machine is designed to permit further expansion, ultimately to 192K. A 128K version of the machine is planned for an autumn launch (see story this page).

DISK DRIVE PLANS

Disk drive units dedicated for use with the Lynx are scheduled to appear in the shops on July 28.

The disk drives will run single-sided, double-density, 40 track 5¼ins 'floppies'. Double-sided disks will be introduced at a later date.

The purchase price of the Lynx disk drive unit with interface pack will be £343.85, including VAT.

There are also plans to introduce the following items, although launch dates have not yet been finalised: light pen, modem, Prestel interface and Network.

...AND UPGRADES FOR THE 48K LYNX

Owners of 48K Lynxes won't be neglected in the forthcoming launch of Camputers products.

You will be able to turn your 48K machines into 96K versions, simply by taking advantage of Camputers' Upgrade Service.

The process is simple. All you have to do is send your original 48K Lynx back to Camputers' Service Department via your local dealer. You must make sure that there are clear instructions on the packaging requesting the upgrade to 96K. You must also include a note of your name and address, and a cheque for £89.95. Please do not send your machines to our office address, since this will inevitably result in delays. The dealer will know the address of our Service Department.

The difference in price between the upgraded 48K machines and the boxed 96K versions covers postage and packing from Camputers back to you, the owner, as well as the man-hours involved in making the physical alterations and a thorough check on the machine before it is despatched.

Free software

All the 96K machines, whether they have been bought new or been upgraded, will have printer controls in ROM. But you will also be able to drive printers with a 48K machine. For serial printers, you will need a lead and software. The lead will be on sale in July at £3.99, and the software will be on a free cassette, available at the same time.

For parallel printers, you will require an interface pack, available in August for £49.95.

128K Lynx

Conceived for the serious business, scientific and educational user, the 128K Lynx — for which an Autumn launch is planned — is the top of Camputers' range for 1983.

The key feature of the 128K Lynx is that it runs CP/M, and thus opens the door to a wide range of professional software. The RS232-type serial port makes communication with other peripherals and hardware a simple matter.

Memory expansion

The memory expansion includes a more powerful screen driver, doubling the resolution and increasing the text display from 40 to 80 characters across the screen, allowing true word-processing capability.

CHILD'S PLAY: Thomas Bull, aged 2, gets to grips with his father's new 48K Lynx.

Dave Bull, a Southampton teacher, won the Lynx in a competition organised by Your Computer magazine. Readers were invited to complete the sentence. 'A Lynx would bring out the animal in me because...'.

Naturally, this brought down a deluge of dreadful puns on the heads of the magazine's editorial staff. 'I wynx and blynx, but the Lynx thynx', suggested E. Jupp; H. Howarth came up with 'It's the big-byte cat — great grrraphics and a purrfect purrocessor'.

Dave Bull's winning entry was 'It's the purrfect way to be an on-line feline'.

SOFTWARE
The story so far

CAMPUTERS HAVE been besieged with enquiries about software support for the Lynx. Many of you will no doubt be relieved to find that it is now appearing, both by mail order and through the shops.

Camputers have set up their own software house, Camsoft (see story below), and it has been their task to evaluate the games and educational programs that have been landing on the doorstep.

Naturally, Camsoft want to be certain that any piece of software issued on their own label is making correct use of the Lynx's far-ranging and individualistic talents. If that job is to be done thoroughly, it takes time — hence the delay in software reaching the market.

Numerons

However, the first two Camsoft tapes are already in the shops. The first is called Numerons, although it actually consists of three games on one tape.

The idea is to shoot down descending attackers, as in Space Invaders. The difference is that each of the attackers bears a number, and in order to destroy it you have to fire specific numbers at it. It relies partly on the player's arithmetical agility and dexterity.

The three games — Niners, Standard Numerons and Advanced Numerons — have different skill levels, the most difficult of which is positively fiendish. Numerons will give hours of fun, and costs £9.90 from reputable computer shops.

The second Camsoft tape is called Election Analyst. In a nutshell, it allows the user to employ the computer as a 'swingometer'. You program in the results of the previous election and then the results of the June election. The computer will give you an instant analysis of the swings.

Five more Camsoft tapes are also in the pipeline — Connect 4, Dambuster, Moonfall, Sultan's Maze and Monster Mine.

Gem Software have also developed games for the Lynx. They are currently selling five tapes, including two compendium tapes. Gem tapes all sell at £7.95 and can be ordered by mail from Gem Software, Unit D, The Maltings, Sawbridgeworth, Herts. For further details see the advertisement on page 23.

Adventure

Level 9 Computing, have now released three of their highly addictive adventures in Lynx versions. They are Colossal Adventure, Adventure Quest and Dungeon Adventure. Each game costs £9.90 and they are available from Level 9 Computing, Dept X, 229 Hughenden Road, High Wycombe, Bucks. for further details, see their advertisement on page 8

Those of you who are new to microcomputing may be interested to know that Teach Yourself Basic is available on a Lynx-dedicated cassette from branches of the Spectrum retail chain.

Camputers will shortly be launching an assembler on a ROM cartridge (£39.95) and cassette (£29.95). This should help those who like to devise their own games.

WANTED-SOFTWARE WRITERS

AS THE manufacturers of a new micro, we at Camputers want to give as much encouragement as possible to software writers developing programs for the Lynx. We have set up our own software house — Camsoft — which is always on the lookout for promising material.

We can offer writers good terms, either on a royalty or straight fee basis. If your program is selected for distribution we will undertake responsibility for production, printing, marketing and advertising support.

We are looking for software support for the Lynx in the following areas:

GAMES — action, adventures with strong graphic content, and serious games, such as chess.

EDUCATION — primary, secondary, further education, and programs designed for the handicapped.

HOME — economics, help in the kitchen, home doctor, diary.

BUSINESS — accounts control for specific types of business, stock control, invoicing, word processing.

Send us details of your software, preferably with a demo tape. Don't worry if the tape is designed for another kind of computer; we can run it here. The address to write to is:—

Camsoft Ltd.,
33a Bridge Street,
Cambridge.

We would like to assure all contributors that their copyright on material submitted to Camsoft will not be infringed.

CASSETTE CORNER

THERE ARE two aspects of the microcomputing life that can reduce newcomers to tears. The first occurs when you spend three hours unsuccessfully trying to make the computer load a program from a cassette tape. The second comes with the realisation that you have just lost a 100-line program because the cassette recorder failed to take it on board when you hit the 'SAVE' keys.

Problems of this nature usually stem from the recorder. The volume or tone controls may have been incorrectly set. Foreign bodies on the recording head may have blurred the signal to the extent where the computer cannot recognise the beginning of the program.

Or the fault may lie with the tape, particularly if you are using old audio stock. The coating on cassette tapes tends to fall off after a while. This creates 'drop-out' — blank sections on the tape. If you are listening to music on such a tape it doesn't really matter.

But computer program signals must be transmitted in full at a clear consistent rate. Even if you have a tiny amount of drop-out it will prevent the program from loading.

It is wiser to use cassette tapes that have been manufactured specifically for computing purposes. The coating is more even and therefore there is less likelihood of drop-out.

The solution to the recorder problem is more difficult to pin down. Some machines will save but not load; some will load but not save; some will save and load your own programs but will not load with bought-in software.

Cassette recorders were designed for audio use long before computers used them to record signals, and while some deliver an inverted phase signal to the recording head, others deliver an uninverted phase signal.

This means very little to a loudspeaker, but makes all the difference in the world to a computer. For this reason, tapes recorded on one recorder will not necessarily work on another.

As a case in point, we can recommend the Radio Shack CCR 81 and the Tandy Realistic CTR 60, which both work with the Lynx.

We are trying to build up a list of cassette recorders which are known to work both consistently and successfully with the Lynx.

If your recorder has performed perfectly for save and load operations, please write to the Editor of this newsletter at 33a Bridge Street, Cambridge CB2 1UW, giving details of the machine. We hope to be publish a list of tried and trusted machines in the next edition of Lynx User.

MORE FOR YOUR MANUAL

THERE ARE a number of errors in the manual. Some of them are obvious typing errors — a 'FOR... NEXT' instead of an 'IF... THEN' has achieved national fame in a WHAT MICRO? review!

Some are very frustrating errors in program examples, most notably in 'Watch this!' on page 65 which is somewhat less than spectacular because the semi colon at the end of

130 PRINT B$;

has been mislaid and a

90 CLS

wouldn't go amiss.

But there have been three rather embarrassing mistakes pointed out to us, which every Lynx owner should know about! The first two are very similar mistakes. As you may have read on page 20, the instructions for entering EDIT mode (page 36) read

[CONTROL]Q[RETURN]
instead of
[CONTROL]Q

and

[CONTROL]E[RETURN]
instead of
[CONTROL]E

Following the same desire for symmetry, the instructions for entering and exiting graphics mode (page 55) read

[CONTROL]1[RETURN]
instead of
[CONTROL]1

If you do put in [RETURN]s you can't type in a string of graphics characters because when you hit [RETURN] — to exit graphics mode — the line is entered into the computer before you've had a chance to close the string with (") and you're given a syntax error.

The third mistake is a conceptual one on my part. The cursor keys happen to have arrows on them. They have nothing to do with the arrow symbols in the character set — when you press an arrow key, you don't get an arrow on the screen, you see the cursor move.

I mixed them up. Page 30 claims that to move a token across the screen you would use

IF KEYN=123 THEN C=C+1

but 123 is the code for the (right arrow) symbol, not for the cursor movement the code for 'move cursor to the right' is 12 — see the list of Control Codes on page 62 — so to move the token to the right you would use:

IF KEYN=12 LET C=C+1

The codes for the cursor keys are:

CURSOR MOVEMENT CODES:

→	12	Although this is not implemented as a VDU command, it will work with KEYN and GETN
←	22	
↑	11	
↓	10	

The codes for the arrow symbols are:

ARROW SYMBOL CODES:

→	123	You can display these on the screen using VDU or ?CHR$( ).
←	124	
↑	125	
↓	126	

Omissions

Some commands were omitted from the manual because they were not finalised until after it went to print.

1. Arguments of VAL, ASC, LEN can be string expressions, for example:

VAL("7"+A$)
LEN(A$+B$)
ASC("7")

2. The STR$ function converts a numeric expression or constant into a string, so:

if A=7.93

then STR$(A)="7.93"

3. If you want a string array, A$, for example, with elements of length L and highest element H, then

DIM A$(L)(H)

(note the extra brackets)

A$(7) will give the 7th element. A$ and A$(0) will give the 0th element.

4. TEXT is a useful command which is equivalent to

PROTECT 0
INK GREEN
PAPER BLACK
CLS
PROTECT MAGENTA

By switching out two of the colour banks it allows the machine to print to the screen at about twice normal speed, though only in GREEN on BLACK. It is ideal for LISTing, etc., and programs which do not use graphics. To turn it off use PROTECT 0.

Technical Manual

Camputers are at present compiling a technical manual for the advanced programmer. As soon as it is ready we will publicise its availability.

Several publishers are currently preparing books about the Lynx, and notification of their publication dates will be posted in the newsletter. 'Lynx Computing' by Ian Sinclair is already in the shops. It's published by Granada at £6.95, and you should be able to buy it from most reputable computer shops as well as Laskys and Spectrum dealers.

Coming Shortly...

Two complex subjects had to be omitted from the manual: EXT and USER 0-3. Complete documentation will appear in the near future, but in the meantime, here are brief descriptions of them.

EXT allows you to add COMMANDS to the Basic. All your extensions take the prefix EXT. You can write your own routines for input and syntax checking and execution; alternatively, if your extension has the same format as an existing command you can use the ROM routine for that command.

If you try to use EXT and are given a NOT YET IMPLEMENTED message, this does not mean that the EXT routines are not in your ROM, it means that you haven't defined an extension command yet!

USER allows you to add functions. To use it you need to understand the mechanisms the Basic interpreter uses to fetch and process a function argument. USER takes a single floating point argument. Vectors to the routines are at 627C, 627F, 6282 and 6285 (see the list of System variables). The value is passed to WRA1 at end of function.

Finally, DISK (on page 88) will not work unless a disk drive has been attached. Meanwhile, don't use it, as the machine will not understand the DISK instruction.

STAR ROVER - Scrolling the Lynx 48K display

by George Kendall

THE LYNX display does not scroll normally because the chip used to control the video display — the **Cathode Ray Tube Controller (CRTC)** — works in units of four pixels vertically. The Lynx character set is 10 pixels high.

The Lynx's display can be scrolled horizontally or vertically, in either direction.

The video display is handled by a specialised chip — the 6845. This **CRTC** has a total of 18 **registers** — memory locations in the chip which are used to control its function.

Each register can store a binary number, comprising eight bits labelled 0 to 7 from right to left. Each bit can have a value of either 1 or 0 (set high or set low). So, for example,

```
  bit:76543210
value:00000100
```

bit 2 is set high, all the other bits are set low. The value of each of these bits is important — changing any one of them will alter the operation.

How to scroll

To send instructions to the CRTC — and scroll the display — you need to send instructions to two Z80 PORTS using the Basic OUT command.

The Z80 PORTS relevant to scrolling are 86 H (H = hexidecimal) and 87 H: 86 is used to select the CRTC register, 87 to send data to that register.

We are concerned with two registers, 12 and 13. Bits 0 to 4 of register 13 affect the horizontal displacement of the screen. If they are all set low the position is normal. To change the position, you need to change one or more of them to 1. The amount of displacement is determined by the value of those 5 bits, read as a 5 digit binary number.

So, for example, OUT &86,13 will select register 13, then OUT &87,2 will send the value 2 (binary 10) to register 13.

The vertical displacement is determined by the value of bits 5-7 of register 13 and bits 0-2 of register 12. Be careful not to set bits 3-5 of register 12 high, or the machine will crash!

If you put a value of 1 into register 13, the screen will scroll horizontally 8 pixels; if you put in a value of 32, it will scroll vertically 4 pixels.

Scrolling in Star-Rover

Star-Rover is a short program which demonstrates scrolling. It has no ending: when you run out of energy you are given an error message. You could try adding another procedure, ENDGAME (called in line 348) to give a more professional ending to the game. A table showing the function of each variable is given with the program listing.

Scrolling is handled like this: in the procedure JUMP, PORT 86H has already been set to 13 by the machine code routine called by line 130. Line 220 then uses PORT 87H to send the value of Z to register 13, which will then scroll the screen upwards.

After changing the bottom section of the screen and printing up the spaceship, the value of Z is increased by 32 so that when JUMP is called, the screen will be scrolled up again.

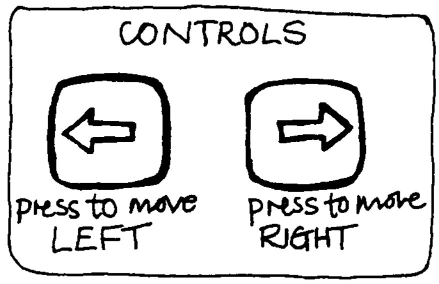
When the screen has been scrolled 16 times — using all four of bits 5-7 set to 1 — to scroll the screen on another unit all these bits must be set to 0 and bit 0 of register 12 set to 1. To do this in Basic would take four commands:

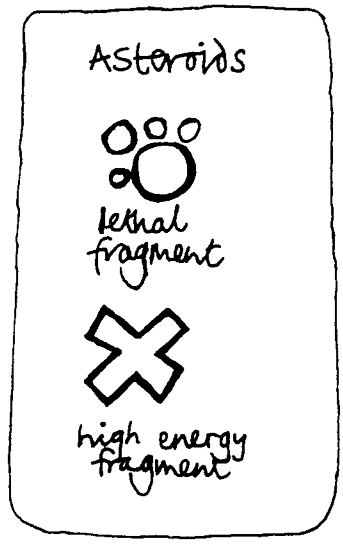
```
OUT &86,12
OUT &87,z
OUT &86,13
OUT &87,0
```

and the screen would flicker. To prevent this, the instructions have been put into machine code, and this routine is contained in line 1030.

The value of z is entered into the routine in line 120 by POKE L+6,z.

For other games, you could change the value sent to register 13 by a POKE L+13,Z.





VARIABLE TABLE

A$	**Character $ which makes up the space ship.**
C	**Parameter of PROC COLLIDE: gives the ASCII code of the object collided with.**
C( )	**Character of asteroid etc., in row specified by subscript.**
E	**Energy level of space ship.**
L	**Location of line 1030.**
M	**Vertical position in window of the 'bottom' of the screen display (in units of 4 pixels).**
P( )	**Horizontal position of asteroid, etc., in row specified by subscript.**
V **W**	**Counters for BEEP in PROC COLLISION and elsewhere.**
X	**Counter for PROC INITIAL and horizontal position of space ship.**
Y	**Vertical position of space ship.**
Z	**Value to be entered into register 13 for scrolling.**
z	**Value to be entered into register 12 for scrolling.**

STAR ROVER – THE PROGRAMME

You are the commander of a trans-galactic spaceship. With your energy levels seriously low and your engines shattered, you are lost in an asteroid field. Using the cursor keys, you can steer through the field, dodging asteroids. Luckily, some of the fragments have a high energy value and by intercepting these you can gain extra energy. Colliding with other asteroids, however, drains your energy via your force-shields. And, of course, when your energy level reaches zero...

```
10   TEXT
20   PROTECT YELLOW
25   INK BLUE
28   REM **** STARS
30   FOR V=1 TO 200 STEP 5
40     DOT RAND(V),RAND(256)
50   NEXT V
80   WINDOW 0,127,0,255
90   PROC INITIAL
110  REPEAT
120    POKE L+6,z
130    CALL L
140    REPEAT
150      PROC JUMP
160    UNTIL Z=256
170    LET Z=0,z=(z+1) MOD 8,
     Y=Y MOD 256
180  UNTIL FALSE
200  DEFPROC JUMP
220  OUT &0087,Z
225  PROC ASTEROID
230  IF INP(&0980)< > 255 THEN
     PROC ACTION
232  PROTECT CYAN
235  PRINT @ X,Y;A$;
240  LET Y=Y+4,M=(M+1) MOD
     64,Z=Z+32
250  ENDPROC
300  DEFPROC ACTION
310  IF INP(&0980)=251 THEN LET X
     =X-1
320  IF INP (&0980)=223 THEN LET
     X=X+1
330  IF X=2 THEN LET X=3
340  ELSE IF X=118 THEN LET
     X=117
342  REM **** ENERGY LOSS
345  BEEP E, 1000 DIV E,63
347  LET E=E-1
348  IF E < =0 THEN PROC
     ENDGAME
350  ENDPROC
400  DEFPROC ASTEROID
402  PROTECT BLUE
405  PRINT @ P(M), M*4;" ";
420  LET P(M)=RAND(126),C(M)=138
     +SGN(RAND(17)-8)*2
425  IF C(M) < > 138 THEN INK
     GREEN
428  PROTECT WHITE-INK
430  PRINT @ P(M),M*4;CHR$(C
     (M));
440  REM****CHECK COLLISION
450  IF (P((M-54) MOD 64) > =X
     AND P((M-54) MOD 64) <
     =X+4) THEN PROC COLLIDE
     (0)
455  ELSE IF (P((M-55) MOD 64) >
     =X AND P((M-55)MOD 64) <
     =X+4) THEN PROC COLLIDE(1)
460  ENDPROC
500  DEFPROC COLLIDE(C)
502  IF C((M-54-C) MOD 64)=138
     THEN LET C=20
505  FOR V=4 TO 1 STEP -0.5
510    FOR W=1+C TO 31
520      BEEP W*V,4,31*V+1-W
530    NEXT W
535  NEXT V
536  IF C=20 THEN LET E=E+20
538  ELSE LET E=E-50
539  IF E < 0 THEN PROC
     ENDGAME
540  ENDPROC
1000 DEFPROC INITIAL
1005 DPOKE GRAPHIC,LCTN(1090)
1006 DIM A$(9),P(63),C(63)
1008 LET Y=20,M=61,Z=0,z=0,L=
     LCTN(1030),E=150
1009 INK RED
1010 PROTECT CYAN
1011 REM****This FOR-NEXT loop
     reads the data in line 1020 and
     uses it to determine what
     characters and what control
     codes the spaceship (A$) will be
     printed up as.
1012 FOR X=1 TO 9
1014   READ V
1016   LET A$=A$+CHR$(V)
1018 NEXT X
1020 DATA
     1,2,128,130,22,22,10,132,134
1021 FOR X=0 TO 63
1022   LET P(X)=0
1023 NEXT X
1025 ENDPROC
1027 REM****************
     ***********
1028 REM Machine code program to
     set register 12 with the value in
     the 8th byte and register 13 with
     that in the 16th byte: I.E.: TO
     SCROLL
1030 CODE 01 86 0C ED 41 3E 00 D3
     87 04 ED 41 3E 00 D3 87 C9
1079 REM****************
     ************
1080 REM : 11 Code statements that
     define the even-numbered
     characters from 128 to 148 :-lines
     1090-1160 = spaceship;
     1170-1180 = asteroids etc.
1090 CODE 00 00 00 00 00 00 00 07
     07 03 00 00
1100 CODE 00 00 00 00 20 20 20 3C
     38 30 00 00
1110 CODE 01 01 09 09 0F 0B 01 01
     01 01 00 00
1120 CODE 30 30 30 28 28 28 30 20
     00 00 00 00
1170 CODE 2C 18 34 18 00 00 00 00
     00 00 00 00
1175 CODE 24 18 18 24 00 00 00 00
     00 00 00 00
1180 CODE 14 20 0C 20 00 00 00 00
     00 00 00 00
2000 REM A subroutine to normalise
     screen display.
2005 OUT &0086,12
2010 OUT &0087,0
2020 OUT &0086,13
2030 OUT &0087,0
2035 WINDOW 3,123,2,245
```

BASIC READING

There are many books available on Basic. When you are choosing a book, there are two important things to remember.

First, some books on Basic assume that you are using a mainframe computer and cover things which are not relevant to a micro.

Second, there is no 'standard' Basic on micros — not even Microsoft! — and Lynx Basic is very different from some of the other implementations available. So it is best to look for a book which is not only 'machine-independent' but which also stresses throughout the text where commands etc. are likely to vary from Basic to Basic, because example programs will almost always need to be translated.

Bearing these points in mind, here are just a few books you may find useful, with approximate prices.

Illustrating Basic by Donald Alcock (CUP), £3.50 for the spiral bound paperback version.

This is a clearly written book which is careful to examine different versions of the language. You will either love it or hate it because it is handwritten from cover to cover, and illustrated with witty little drawings.

Computer Programming in Basic by Peter Bishop (Nelson), £5.55 in paperback.

A clearly written and beautifully presented book. Every example program is accompanied by a flow diagram.

Elementary Basic: teach yourself Basic by solving the mysteries of Sherlock Holmes ('as chronicled by Dr John H. Watson, edited with commentaries by') Henry Ledgard and Andrew Singer (Fontana), £4.95 in paperback.

This book is a novelty. Each chapter begins with a short Sherlock Holmes story, written in a fairly convincing style, which poses a problem. The chapter on Arrays, for example, is called 'The Ciphered Letter'; it sets out an algorithm for solving the problem, then develops a program. As an added bonus, it contains some of Sidney Paget's original Sherlock Holmes illustrations. Program is spelt 'programme'!

Finally, two books aimed at the more advanced user:

Problem Solving With Basic by Richard Dillman (Holt, Rinehart and Winston), £16.95 in paperback.

Problem Solving and Structured Programming in Basic by Elliot B. Koffman and Frank L. Friedman (Addison-Wesley), £7.95 in paperback.

BASIC HINTS

Thank you to everyone who has helped to bring bugs in Lynx Basic to light!

Mr Thomasson, besides several other people, has pointed out that the values of ARCSIN(1) and ARCCOS(1) are the wrong way round.

You can run into problems with REM statements if you include an odd number of "s in them. For example:

```
10 REM "[RETURN]
LIST [RETURN]
```

can have spectacular results. But you can EDIT line 10 — you may have to reset INK and PAPER colours by using TEXT — to remove the offending quotes, and continue as normal.

In addition, the Lynx has a couple of idiosyncrasies which can't really be called bugs, but which may cause you some problems.

If you have a line like this:

```
10 IF Y=T OR Y=P THEN GOTO 100
```

you will get a syntax error message. Mr Walton, of Manchester, rightly pointed out that it was because the Basic sees the line as

```
IF Y= TO R.........
```

The problem occurs because the Lynx allows you to leave accidental spaces in commands. You can type DEF PROC or GO TO or even GO TO. Because TO is a possible whole word, it should only happen with T and O. For example,

```
IF Y=G OR.....
```

will not be mistaken for a mis-spelt GOTO because when it reaches the 'R', the Basic interpreter will know it is not GOTO or GOSUB and will look for some other construction.

A similar problem arises because the Lynx allows commands to be abbreviated — F. instead of FOR, and so on.

The problem is that if you try to multiply, for example, 10 by .2:

```
10*.2
```

the computer sees * as an abbreviation of ** — 'raised to the power of' — and gives an answer of 100! You can solve this problem easily, of course, by typing

```
10*0.2
```

Finally VDU 14 and 15 have been causing problems because they are not adequately explained in the manual — ie nobody explained them to the manual writer!

Only part of the Lynx screen driver — the routine in the ROM which controls the screen display — is used by the Basic. The cursor is not used by the Basic and when you power up it is turned off. The "cursor" you see — the flashing block — is not really a cursor, just a marker to show your position.

Try this:

```
10 VDU 14 [RETURN]
20 PAUSE 2 0000 [RETURN] RUN
```

An underline will appear at the end of the next line. This is the real cursor, the one which is switched on and off by the VDU command. It is automatically switched off at the end of the program.

If you want to get rid of the flashing block, just redefine it, using CCHAR, as two spaces (ASCII code 32).

RESETTING AFTER A CRASH

Generally speaking, to regain control of the Lynx after it has crashed, you need to disconnect it from the mains — USE THE MAINS SOCKET SWITCH OR PLUG; AVOID DISCONNECTING THE POWER SUPPLY LEAD FROM THE MACHINE.

But there is a command which will reset the Lynx — if the crash is not too bad — and which is worth trying, even if it doesn't appear on the screen as you type it in.

Type:

```
CALL 0 [RETURN]
```

This action, unfortunately, will lose Basic programs and any machine code lines contained in them. Machine code programs will also be lost.

PRINTING WITH THE LYNX

Serial Printers

Many of you are already aware that the Lynx needs additional software which will be available in July on cassette or listing, with a lead, specifically for the Seikosha GP250X or faster printers.

Please accept our apologies for this inconvenience. Those 48K Lynx owners who wish to drive serial printers should order the lead and software through their stockist. Approximate price is £3.99 (for cassette and lead).

Parallel Printers

These will be driven via an adaptor to be plugged into the expansion bus socket. This will also be available in August and should be ordered through your local stockist. Approximate price is £50.00.

INTERRUPTS ON THE LYNX 48K

It has been suggested that the hardware cursor interrupt be used to access the display during the frame blanking period by enabling interrupts, which are vectored to 38H from which control is passed to the user's own code.

The recommended procedure is that interrupts are to be enabled and the halt instruction immediately executed. The halt instruction must be located at an address with either A7 Low or A6 High, hence the cursor interrupt is suitable for writing some games software requiring a fast display.

SAVING VARIABLES

Many people have asked whether it is possible to save a program complete with variable values on the Lynx. Here is a short routine which can be added to any program in a CODE line, and which will save variables, PROTECT levels and INK and PAPER colours.

```
1000 CODE EB 21 F6 61 E5 2A 1F 62 E5
     21 00 00 C3 FF 3E     [RETURN]
1010 REM "program name" [RETURN]
1020 CALL LCTN(1000), LCTN(1010)
     [RETURN]
1030 REM program will run from here!
     [RETURN]
```

Save your program by using

See page 19 LU 2

```
RUN 1020
```

(the program name is contained in line 1010). To LOAD the program use

```
MLOAD "program name"
```

the program will automatically run from the line following that containing the CALL LCTN.

TROUBLE WITH YOUR INTRO TAPE?

We've heard from a few people who've had difficulty loading the intro tape supplied with their Lynx, despite being able to SAVE and LOAD their own programs satisfactorily with the same tape recorder.

We've now changed the recording of the tapes to overcome this problem, but if you're still having problems with yours we'll be happy to replace it. Just send us the old tape, preferably in a Jiffybag — and don't forget to include your name and address, together with a note of the serial number of your Lynx.

Is there life after NEW?

Mr Walton of Manchester points out that by using POKE 26957, 192 the program previously NEW'd will be restored with its first line as the number 1. Similarly, POKE 26957, 128 has the same effect as typing NEW.

Mr Walton goes on to ask if we are interested in educational programs: we certainly are, and they should be sent to us for evaluation on cassette, accompanied by full details and a post paid envelope if they are to be returned.

We will be pleased to negotiate terms with those creating first class software (see page 2).

REMOTE CONTROL

If you find the remote control on your Lynx doesn't control your tape recorder when you're saving and loading, you may have a polarity problem. The circuitry of the Lynx is designed to control the polarity normally found in cassette recorder remote circuits, but won't handle circuits with the opposite polarity because it includes a transistor.

The quickest solution is to remove the 'remote' jack from the Lynx cassette lead and reconnect it with the two wires transposed.

JOYSTICK CONTROL

Avid games fans who feel their style is cramped by the lack of joystick control on the Lynx will shortly have an answer to their prayers.

Camputers are scheduled to launch an interface pack which allows the use of Atari-compatible Discwasher PointMaster joysticks in conjunction with the Lynx. The interface face will be priced at £14.95 and will be on sale from the beginning of August.

CALLING THE MONITOR

Some Lynx owners have been disturbed to find that the numbers displayed on the screen when they call up the Monitor are different from those shown in the diagram on page 72 of the Lynx manual.

Don't despair — there is nothing wrong with your Lynx!

The numbers displayed are the values that happen to be stored in the Z80's registers — internal memory — at the moment you call up the monitor. They will differ depending on what the processor was doing when you called it up.

VIDEO PROJECTION

It should be possible for the Lynx to drive wide screen video projection units by using the composite video signal plus ground, from the light pen socket.

FIRST DIMENSION	SECOND DIMENSION	"BOX NO."	EXAMPLE BASIC (PRINT) STATEMENT:
1	1	Ø	PRINT A (1,1)
	2	1	PRINT A (1,2)
	3	2	PRINT A (1,3)
2	1	3	PRINT A (2,1)
	2	4	PRINT A (2,2)
	3	5	PRINT A (2,3)

MULTI-DIMENSIONAL ARRAYS

Hitherto the means of handling these has been incorrectly described by some sources. For this reason, it is explained here in greater depth.

If you wanted an array of 2 by 3, most machines would require the statement: DIM A (2,3) the result of which could be illustrated as shown in the diagram.

Following this, in most machines the statement PRINT A (2,1) would print the contents of 'box 3' illustrated in the diagram above as computed by the machine from the DIM statement.

It is possible to compute the 'box number' yourself, and in the Lynx this is not only necessary but makes it possible to have as many dimensions as you like.

To explain this let's say that the variable F contains the first dimension, and the variable S the second dimension. If F contained 2 and S contained 3, then the statement: DIM A(F*S) will set up the 'boxes' shown in the diagram above.

To print from the appropriate 'box', instead of saying PRINT A (2,1) you would need to say PRINT A(2*S−1); referring to 'box 3'. In other words *S− is used in place of the comma(,).

Note that the above formula doesn't compute contiguous boxes as shown in the diagram, but this makes no difference in run-time or accuracy. It is this formula which has been previously mis-printed.

Third, fourth, fifth, etc, dimensions can be set up by the above means, though these are of course much more unwieldy and to be left only to the most dedicated with plenty of time to experiment.

An example of two dimensional arrays follows to re-inforce the above, showing the conventional and the Lynx method of setting up and using the array:

CONVENTIONAL METHOD	LYNX METHOD
DIM A(1Ø,15)	DIM A(1Ø*15)
...	...
PRINT A(5,3)	PRINT A(5*15-3)
PRINT A(4,2)	PRINT A(4*15-2)
PRINT A(8,11)	PRINT A(8*15-11)
(etc..)	(etc..)

The UNDEFINED VARIABLE message can occur with dimensional arrays, since the Lynx won't let you read from an array item until you have previously written something into it.

For example the statement: PRINT A(1Ø) will cause an UNDEFINED VARIABLE message if A(1Ø) hasn't previously had a value stored in it.

The same applies to string arrays and the following coding would initialise both types:

```
DIM A(1Ø), A$ (3Ø) (1Ø)
FOR X = 1 TO 1Ø
LET A(X) = Ø, A$ (X) = ""
NEXT X
```

The above caters only for subscripts greater than Ø, i.e. in the range A(1,1) through A(1Ø,15) in the example shown. This saves memory when zero subscripts are not in use.

To cater for subscripts ranging between A(Ø,Ø) and A(1Ø,15) it is necessary to modify the above as follows:

```
DIM A(11*16)
...
PRINT A(5*16+3)
(etc...)
```

Notice that 1 (one) has been added to each dimension and the expression now replacing the comma(,) is:

*S+

(where S is the second dimension plus 1).

Extra dimensions can be accommodated by extending this formula; memory capacity is the only limit.

For example, three dimensions (1Ø,15,3) would be:

```
DIM A (11*16*4)
...
PRINT A (5*16*4+3*4+Ø)
```

which is the equivalent to the conventional

```
PRINT A(5,3,Ø)
```

PROGRAMS

Lynx programmers — this is your page! We're offering a FREE Camsoft cassette to the Lynx user who comes up with the best program for the next issue of this newsletter. The Editor of Lynx User will be looking for tightly written programs that demonstrate the full range of the Lynx's capabilities. Send your entries to the Editor, Lynx User, 33a Bridge Street, Cambridge, CB2 1UW.

SOUND

Don Thomasson

This program demonstrates how the Lynx can be made to play a tune.

When the question 'key?' appears on the screen, press any number to continue.

Lines 120-160 set up powers of the twelfth root of two in B(), these defining semitone intervals in terms of frequency. Lines 170-240 then set up a table which picks out the particular semitones used in a given key. At line 250, a given key is determined by input of a number, which must be an integer (though rounding down occurs if it is not). The tune defined by lines 350-430 is then played in the chosen key. Line 430 is a standard terminator.

There is room for considerable enhancement. Input of a letter for a given key could be translated simply enough, but adjustment of the BEEP factors might be necessary. Alternative tunes could be provided.

```
100 REM Set up musical scales
110 CLS
120 LET A=2**(1/12)
130 DIM B(48),C(32)
140 FOR X=0 TO 47
150   LET B(X)=A**X
160 NEXT X
170 FOR X=1 to 32
180   READ C(X)
190 NEXT X
200 DATA 1,3,5,6,8,10,12,13
210 DATA 15,17,18,20,22,24,25
220 DATA 27,29,30,32,34,36,37
230 DATA 39,41,43,44,46,48,49
240 DATA 51,53,55
250 INPUT "KEY";K
260 RESTORE 350
270 REPEAT
280   READ T
290   READ D
300   LET E=C(D)+K
310   LET F=1/B(E)
320   BEEP 1000*F,T*50/F,63
330 UNTIL T=0
340 GOTO 250
350 DATA 1,8,1,8,1,8,1,9,1,8,1,8,2,5
360 DATA 1,6,1,5,1,6,1,7,2,8,2,8
370 DATA 1,8,1,8,1,8,1,9,1,8,1,8,2,5
380 DATA 1,6,1,5,1,6,1,7,2,8,2,8
390 DATA 1,12,1,11,1,10,1,9,1,10,1,9,2,8
400 DATA 1,6,1,5,1,6,1,7,2,8,2,8
410 DATA 1,5,1,5,1,6,1,7,1,8,1,8,2,9
420 DATA 1,12,1,11,1,10,1,9,2,8,2,11,
4,8
430 DATA 0,1
```

AIR RAID UPGRADE

Thomas Griffiths

Add these few short lines to the program for Air Raid on the Lynx Introductory Tape, and the computer will give you a score of 20 points for every block of buildings you destroy. If you reach the ground but still hit a building you get a 1000 point bonus. If you land safely you get a bonus of between 2000 and 9998 points.

```
305 LET h=3000,i=0,S=0,X=0,H=0
332 IF H=180 AND X=108 THEN GOTO
    340
334 S=0
336 i=1
542 IF H=180 AND X=108 THEN S=S+
    (RAND(4000)+1000)*2
544 IF H=180 AND X<108 THEN S=S+
    1000
735 S=S+20
1115 PRINT @ 10,10;"High Score:";h
1061 IF i=0 OR (H=180 AND X=108)
     THEN GOTO 1069
1063 PRINT @ 40,110; "Your score:";S
1065 IF S>h THEN PRINT TAB 8;CHR$
     (18);"You've got the High Score!!"
     +CHR$(18)
1067 IF S>H THEN h=S
1068 PAUSE 20000
1069 CLS
```

SPLASHDOWN

Don Thomasson

A short graphics programe that demonstrates the Lynx's special screen colour abilities.

```
100 DIM H(255), L(255)
110 LET B=-SIN(0.4), C=COS (0.4)
120 FOR N=0 TO 255
130 LET H(N) =0
135 L(N)=0
140 NEXT N
150 FOR Y=200 TO -200 STEP -10
160 FOR X=-120 TO 120
170 LET R=SQR(X*X+y*Y)
180 IF R= 0 THEN GOTO 200
190 LET Z=SIN(R/10)*1000/R
200 INK INT(R/50)+2
210 LET U=128+X, V=82+INT
(B*Y+C *Z)
220 IF V>=L(U) THEN GOTO 250
230 DOT U,240-V
240 LET L(U)=V
250 IF V<=H(U) THEN GOTO 280
260 DOT U,240-V
270 LET H(U)=V
280 NEXT X
290 NEXT Y
300 STOP
```

RISING MOON

Don Thomasson

A simple demonstration of painting in a circle, which also illustrates the use of inverse trig functions.

```
100 CLS
110 INK 5
120 LET X=128,Y=120, R=50
130 FOR N=1 to R
140 LET A=N-R
150 LET B=ARCCOS(A/R)
160 LET C=SIN(B)*R*1.2
170 MOVE X+A,Y-C
180 DRAW X+A,Y+C
190 MOVE X-A,Y-C
200 DRAW X-A,Y+C
210 NEXT N
220 VDU 24
230 PRINT @ 35,10; "THE MOON IS
    ON HIGH"
240 VDU 25
250 INK WHITE
260 GOTO 260
```

DOWN THE TUBE

A short graphics program that relies on an unusual procedure to draw circles. The circle commands can be lifted out quite easily, and all you need to add are the X and Y co-ordinates.

```
1 PROC CIRCLE
2 VDU 4,21,25
3 PROC BRAIN
4 FOR X = 0 TO 190 STEP 10
5 INK X + 1
6 FOR A = 0 TO 360
7 PLOT 4,30 + (M(A) + X), 30 + (N(A)
  + X)
8 PLOT 4,30 + (M(A) + (190-X)), 30
  + (N(A) + X)
9 NEXT A
10 NEXT X
11 PROC DUNNIT
12 G = GETN
13 END
14 DEFPROC CIRCLE
15 R = 25, r = 25
16 DIM M (360)
17 DIM N (360)
18 CLS
19 VDU 24
20 PRINT @ 40,20; "I'M THINKING"
21 PROC BRAIN
22 FOR A = 0 TO 360
23 B = A*PI/180
24 C = R*COS (B)
25 D = r*SIN (B)
26 M(A) = C
27 N(A) = D
28 NEXT A
29 ENDPROC
30 DEFPROC BRAIN
31 FOR F = 0 TO 100
32 BEEP RAND (300) + 1,3,63
33 NEXT F
34 ENDPROC
35 DEFPROC DUNNIT
36 FOR J = 0 TO 1
37 FOR S = 100 TO 0 STEP - 10
38 FOR U = 0 TO 100 STEP 10
39 BEEP U + S,10,63
40 NEXT U
41 NEXT S
42 NEXT J
43 ENDPROC
```

ROLL OVER, BEETHOVEN

Chris Saffin

This table gives frequencies and wavelengths for musical scales. The actual notes produced are not, strictly speaking, musically accurate — Middle C, for example, should have a wavelength of 415. But each note is proportionally correct.

NOTE	FREQ	W/LEN
A	110	909
A#	116.6	857.3
B	123.5	809.7
C	130.8	764.5
C#	138.6	721.5
D	146.8	681.2
D#	155.6	642.7
E	164.8	606.8
F	174.6	572.7
F#	185	540.5
G	196	510.2
G#	207.7	481.5
A	220	454.5
A#	233.3	428.6
B	246.9	405
MID C	261.1	383
C	277.2	360.7
D	293.7	340.5
D#	311.1	321.5
E	329.6	303.4
F	349.2	286.4
F#	370	270.3
G	392	255
G#	415.3	240.8
A	440	227.3
A#	466.2	214.5
B	493.9	202.5
C	523.2	191
C#	554.4	180.3
D	587.3	170.3
D#	622.3	160.7
E	659.3	151.7
F	698.5	143.2
F#	740	135
G	784	127.5
G#	830.6	120.4
A	880	113.6
A#	932.3	107.3
B	987.8	101.3
C	1046.5	95.5
C#	1108.7	90.2
D	1174.7	85.3
D#	1244.5	80.3
E	1318.5	75.8
F	1396.9	71.6
F#	1480	67.5
G	1568	63.8
G#	1661.2	60.2
A	1760	56.8
A#	1864.6	53.6
B	1975.5	50.6
C	2093	47.8
C#	2217.5	45
D	2349.3	42.6
D#	2489	40.2
E	2637	37.9
F	2793.9	35.8
F#	2959.6	33.8
G	3135.9	31.9
G#	3322.4	30

INSIDE THE 6845 CRTC

REGISTER NO.	REGISTER NAME
AR	ADDRESS REGISTER
RØ	HORIZONTAL TOTAL
R1	HORIZONTAL DISPLAYED
R2	HORIZONTAL SYNC POSITION
R3	SYNC WIDTH
R4	VERTICAL TOTAL
R5	VERTICAL TOTAL ADJUST
R6	VERTICAL DISPLAYED
R7	VERTICAL SYNC POSITION
R8	INTERFACE + SKEW
R9	MAXIMUM RASTER ADDRESS
R10	CURSOR START RASTER
R11	CURSOR END RASTER
R12	START ADDRESS(H)
R13	START ADDRESS(L)
R14	CURSOR(H)
R15	CURSOR (L)
R16	LIGHT PEN (H)
R17	LIGHT PEN (L)

The following information is intended as a guide to the working of the Lynx's CRTC (Cathode Ray Tube Controller) the chip which controls the video display. It has been — severely — summarised from the Motorola Semiconductor data book. If you intend to program the CRTC extensively, you will need a complete data sheet.

Note that the CRTC works in character blocks of 8*4, while Lynx software-defined character blocks are 6*1Ø.

You can use the cursor control in conjunction with RESTART 38 to generate interrupts.

Address Register (AR)
This is a 5-bit register used to select the 18 internal control registers (RØ-R17). You can access registers Ø-17 by writing the appropriate address into this register.

Horizontal Total Register (RØ)
Used to program total number of characters per line including the retrace period. The data is 8-bit and its value will vary according to the display format required. If the total number of characters is M, program (M – 1); when using interlace mode, M must be even.

Horizontal Displayed Register (R1)
Used to program the number of horizontal characters per line. Data is 8-bit and you can use any number smaller than the horizontal total characters.

Horizontal Sync Position Register (R2)
Used to program the horizontal sync position as multiples of the character block period. Data is 8-bit and you can use any value lower than the horizontal total number. If H is the character number of the horizontal sync position, use (H-1). If you increase the value the display is shifted to the left; if you decrease it, the display is shifted to the right.

Sync Width Register (R3)
Used to program the horizontal sync pulse width. The horizontal sync pulse is programmed in the lower 4-bit as multiples of the character clock period — you cannot use Ø.

Vertical Total Register (R4)
Used to program the total number of lines per frame including the vertical retrace period Data is 7-bit. and varies according to the CRT involved. If N is the total number of lines, use (N-1).

Vertical Total Adjust Register (R5)
Used to adjust the total number of rasters per field.

Vertical Displayed Register (R6)
Used to program the number of displayed character rows on the screen. Data is 7-bit and you can use any value smaller than the vertical total characters.

Vertical Sync Position Register (R7)
Used to program the vertical sync position on the screen as multiples of the horizontal character line period. Data is 7-bit and you can use any value equal to or less than the vertical total characters. If V is the character number of the vertical sync position, use (V-1). If you increase the value, the display is moved down.

Interface and Skew Register (R8)
Used to program the raster scan mode and skew (delay) of CUDISP signal and DISPTMG signal.

Maximum Raster Address Register (R9)
Used to program the maximum raster address in 5-bit. Defines the total number of rasters per character including space. If N is the total number of lines, use (N-1).

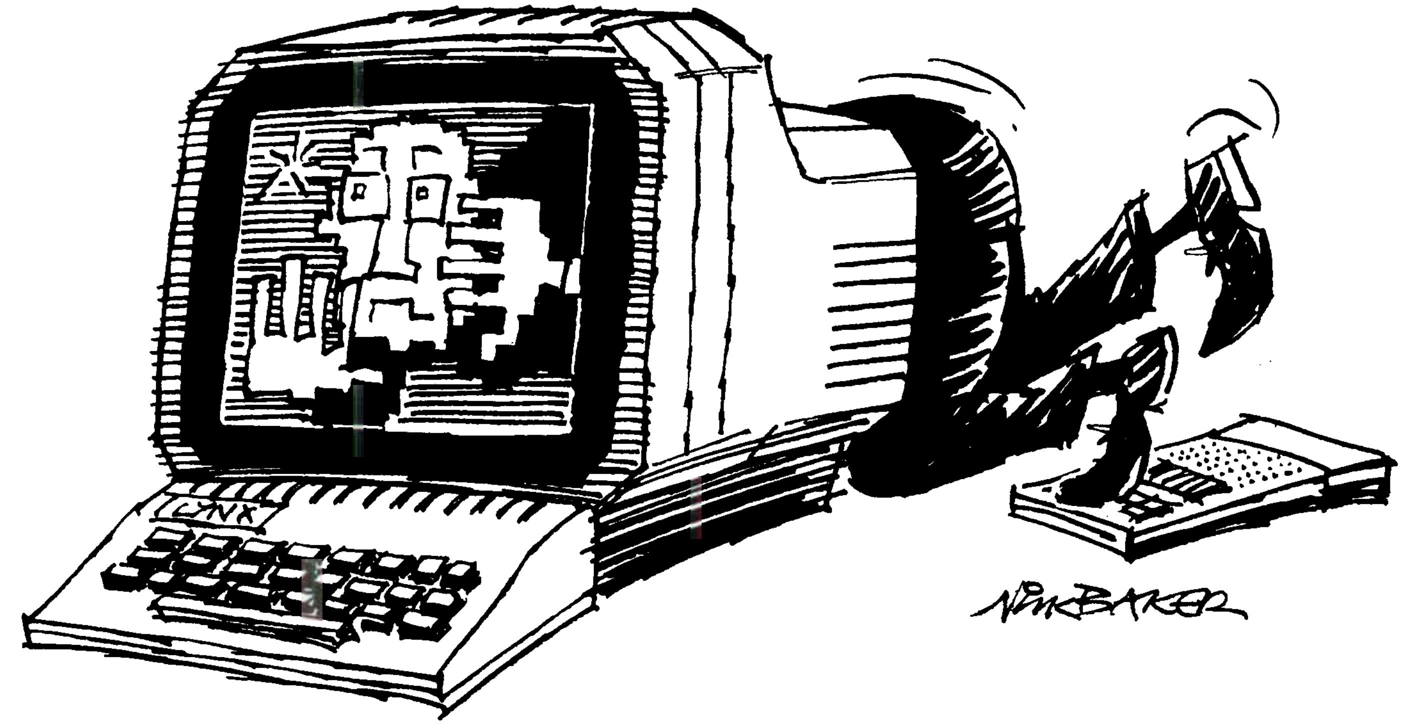
Cursor Start Register (R1Ø)
Used to program the cursor start raster address — with the lower 5-bit — and the cursor display mode — with the higher 2-bit.

Cursor End Raster Register (R11)
Used to program the cursor end raster address.

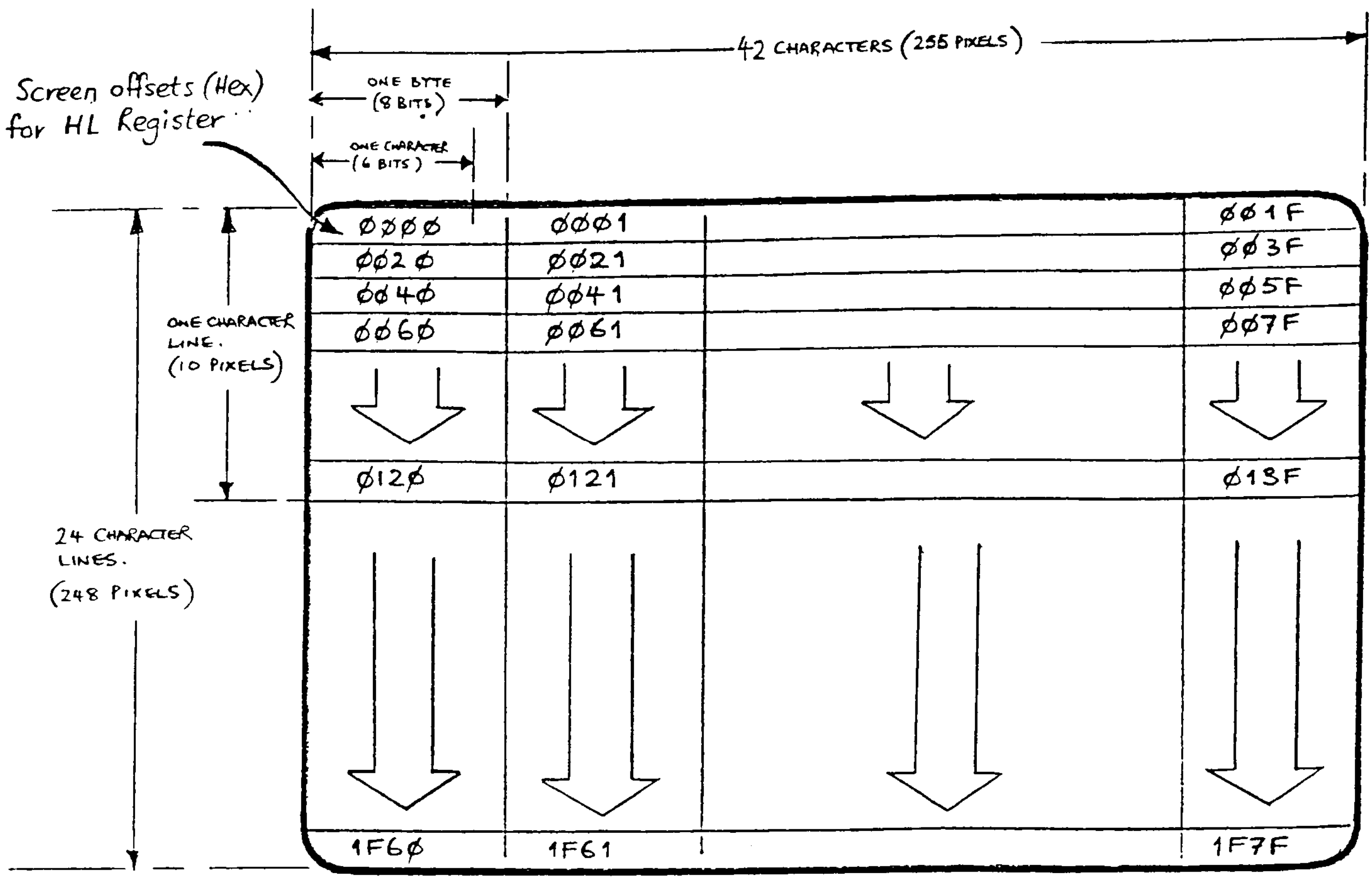
Start Address Register (R12,R13)
Used to program the first address of refresh memory to read out. You can scroll and page easily using these. (See Star-Rover program on page 4.)

Cursor Register (R14,R15)
Read/write registers which store the cursor location.

Light Pen Register (R16,R17)
Read-only registers, used to catch the detection address of the light pen. The value needs to be corrected by software because there is a time delay between detection and response.



SCREEN DISPLAY MAP



Note 1: The address in HL Register must be inverted — e.g. ØØ4Ø becomes 4ØØØ (See Z8Ø manual)
Note 2: The Lynx Basic of necessity does not use the far left and far right bytes. Neither does it use the six bits across the top and bottom of the screen. Thus the Basic print statements use 24 rows and 40 columns

BANK SWITCHING ON THE 48K LYNX

by

DAVID BARNES

64K [illegible] expanded Lynx: [illegible] the [illegible]

THE LYNX'S Basic graphic commands are very simple and comprehensive, but its screen handling is comparatively slow. To write faster graphics programs — in machine code — you need to know something about the internal workings of the machine, and in particular its system of **bank switching**.

This article assumes that you are familiar with machine code programming. Remember that it is just possible that by mishandling the bank switching you could damage your Lynx through **bus contention** — sending more than one electrical current down the same path at the same time. The easiest way to handle the screen is by using the ROM routines described at the end of the article.

The Z80 microprocessor is designed to address ('talk to') a maximum of 64K of memory, but with bank switching it can be made to address more. It's rather like having a set of notebooks; each one has only 64 pages but you can switch between notebooks.

Bank Switching and the Screen or, why you can't PEEK and POKE video memory

On the 48K Lynx, the 32K of video memory is divided into 4 8K blocks: one for each of the primary colours, RED, GREEN and BLUE, plus an extra GREEN block — details of how to use this 'alternative green' block will appear in a later issue.

The colour display is **bit-mapped**. Bit-mapped means that each individual dot or 'picture cell' (**pixel**) on the display corresponds to a bit in the video memory.

To produce a continuous stream of information — to make up the screen display — the video circuitry has to scan its video memory at a constant rate. But the Z80 takes different times to execute different instructions and cannot synchronise with the video scanning easily. (The two systems are 'asynchronous'). However, the Z80 *can* take control of the video memory without interfering with the normal screen display providing it does so during **line blanking** and **frame blanking** periods.

The line blanking period lasts 22μs (microseconds) and allows the electron beam in the television tube to move from the end of one line to the start of the next. The frame blanking period lasts 3.4 ms (milliseconds) and allows it to move from the bottom of the picture to the top of the next picture. Line blanking happens once every 64 μs, frame blanking every 20 ms. The Lynx hardware is designed to allow you to access these periods.

If you want to access the screen at other times, or to transfer a large amount of picture information, you can do so outside these periods, but the display will be degraded — some pixels will be 'blacked out' temporarily and it will therefore be impossible to read them.

CONTROLLING BANK SWITCHING

The following information is for the 48K Lynx; the High Resolution version will be different, and details of it will appear in a later issue.

Lynx hardware provides the decoding for up to 5 banks. The address of the controlling port is:

```
A15        A0
**1*******111111
```

where * = 'don't care' i.e. it can be either a 0 or a 1).

This port is **write only** and the output byte controls the banks as shown in diagram 1.

So, for example, if you set bit 3 high, then bank 4 is **write enabled**; if you set bit 5 high, bank 1 is **read disabled.** The banks are:

D7	D6	D5	D4	D3	D2	D1	D0
RDEN4	RDEN2 or RDEN3	$\overline{\text{RDEN1}}$	$\overline{\text{RDEN0}}$	WREN4	WREN3	WREN2	$\overline{\text{WREN1}}$

BANK 0 (ROM):
0 — 3FFF H is the BASIC
4000 — 5FFF H will be BASIC EXTENSION
E000 — FFFF H any EXTERNAL ROM

Bank 0 is a special case: it can only be read, and can be enabled in conjunction with reads from any one other bank. But it will mask out any reads from addresses 0 to 5FFF; and E000 to FFFF if external ROM is present.

BANK 1: 16K USER RAM, READ and WRITE.

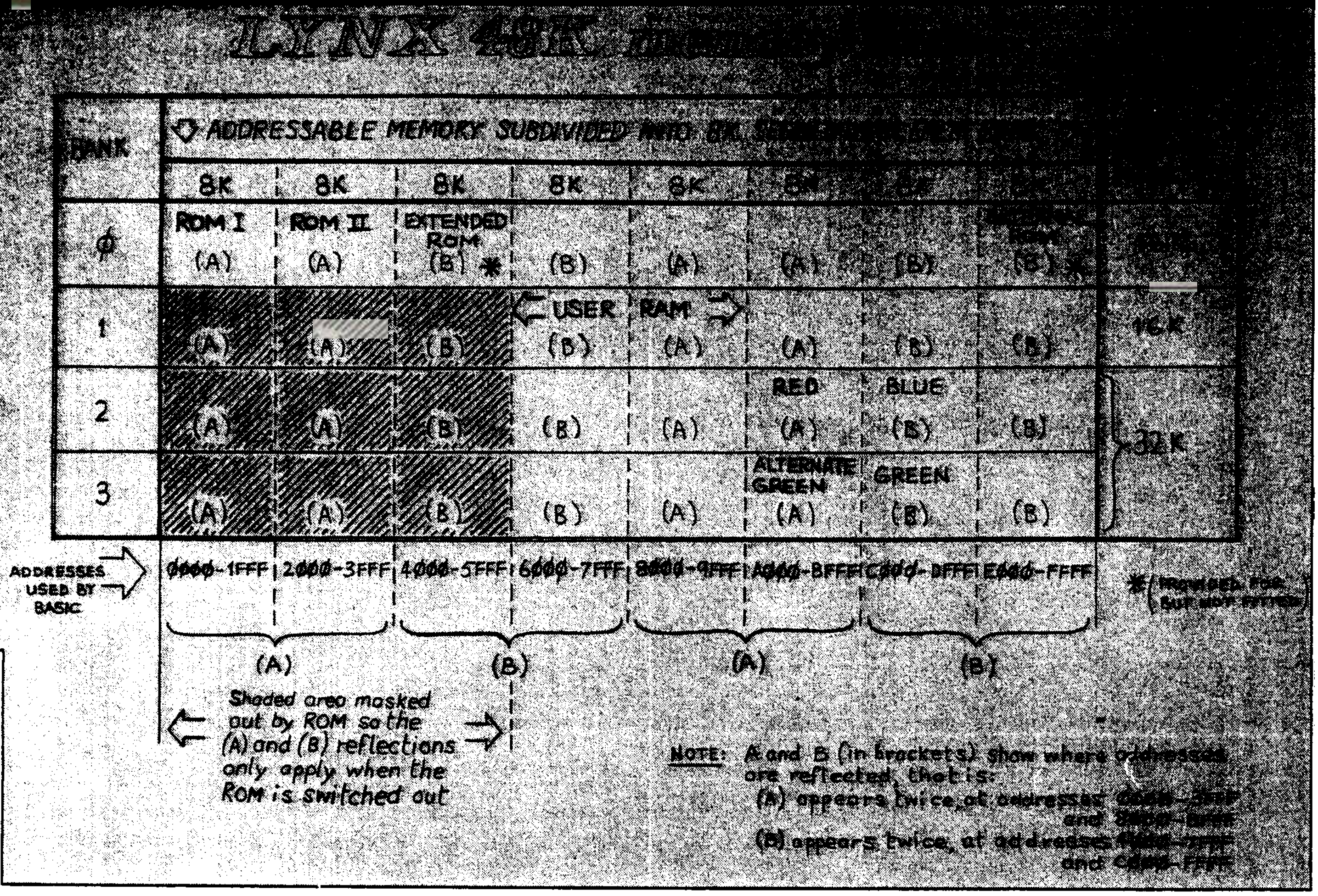
BANK 2: WRITE 16K VIDEO RAM RED & BLUE

BANK 2: READ 16K VIDEO RAM RED & BLUE or GREEN & ALTERNATIVE GREEN, depending on PORT 80 (see below).

BANK 3: WRITE 16K VIDEO RAM GREEN & ALTERNATIVE GREEN.

BANK 4: is provided for off-board expansion.

Any number of banks can be enabled simultaneously for writing, but only one bank at a time, other than with bank 0, can be enabled for reading.



D7	D6	D5	D4	D3	D2	D1	D0
Ø FRAME BLANKING	LINE BLANKING MONO-STABLE	CPU ACCESS	Ø=GREEN 1=ALT. GREEN	BANK3 CASEN	BANK2 CASEN	Ø	Ø

This example of switching the banks is taken from the Basic interpreter. It is part of the routine for clearing the screen.

A contains Bank switch data, HL contains the start address of the screen.

```
08CF  017FF    SCLSS  LD   BC,OFF7FH  ;BANK SWITCH
08D2  ED79            OUT  (C),A      ;
08D4  010040          LD   BC,4000H   ;SCREEN LENGTH 4000H TO
                                      ALLOW FOR HI RES VERSION
08D7  72       SCLS1  LD   (HL),D     ;D CONTAINS DATA TO BE
                                      DISPLAYED
08D8  0B              DEC  BC
08D9  23              INC  HL         ;MOVE TO NEXT BYTE
08DA  78              LD   A,B
08DB  B1              OR   C
08DC  20F9            JR   NZ,SCLS1   ;DO UNTIL BC=0
08DE  AF              XOR  A
08DF  017FFF          LD   BC,OFF7FH  ;SWITCH BANKS
08E2  ED79            OUT  (C),A      ;
08E4  C9              RET
```

To clear red bank, for example

```
LD A,23H
CALL SCLSS
```

See page 19 Lynx User 2.

PORT 80

Access to the video RAM is controlled by PORT 80. The address of PORT 80 is:

A15 A0
********10***00*

and the data sent to it controls the video according to diagram 3.

- D0, D1 — used by cassette and loudspeaker.
- D2 — when high makes reading and writing to the RED & BLUE banks impossible, and RED & BLUE are not shown.
- D3 — similar to D2, but affects GREEN & ALTERNATIVE GREEN.
- D4 — GREEN/ALTERNATIVE GREEN selector. The Basic will return this to GREEN after a screen access.
- D5 — CPU access when set high, disables the normal video scanning and allows access to the video RAM.
- D6 — can be used to synchronise the Z80 to the video line blanking — if it is set high, the Z80 will be 'frozen' until the next line blanking period, when it will start up again. Then the screen can be accessed without any picture deterioration.
- D7 — must be kept zeroed.

Accessing the screen during frame blanking – strictly for experts!

Accessing the screen during frame blanking allows you much more time for access. The Z8Ø does not know where the ray is in the television tube – it has to be synchronised by hardware. You can do this by using the hardware cursor facility on the 6845 as a (maskable) interrupt. The Basic has already placed the cursor at the bottom right hand corner of the screen, immediately before the ray flies back.

By enabling interupts, and jumping to a special interrupt routine, you could access the screen during frame blanking.

You would need a good knowledge of machine code, a 6845 data sheet, and a Z8Ø handbook to manipulate the bank switching.

NB: Writing to Ports FFFF and ØØ8Ø halts the video display system, i.e. the Z8Ø MPU assumes immediate control instead of waiting for a blanking period. You must therefore write to these ports from within your interrupt routine to avoid this. For more details on interrupts, see page 6.

ADDRESS	NAME	FUNCTION	
69H	INBLUE	Reads byte in red/blue bank at HL into L, and H=Ø	READ
70H	INGREEN	Reads byte at green/alt. green bank at HL into L, and H=Ø	READ
		SYSTEM VARIABLE:	
626C	OUTB	Jumps to routine which outputs byte with data in A, mask in C and relative displacement from top left of screen in HL. Uses colour selected by INK/PAPER + PROTECT.	WRITE
Note that previous information about these routines was incorrect. * * *			

ROM ROUTINES

Three routines have been included in the ROM to allow you to read and write to the screen.
For example, to put a dot on the screen using OUTB, put the data into A.

If you are writing to bit 0:

A contains 00000001 gives a dot in INK colour.

or

A contains 00000000 gives a dot in PAPER colour.

(these colours *are* affected by PROTECT levels.)

The other bits 1-7 are 'masked' by a value of 1 in the corresponding bit of C: 11111110

If you were writing to 4 bits A and C would look like this:

A – 00001010

C – 11110000

NB – HL must contain the relative offset from the top left of the screen.

This example of switching the banks is taken from the Basic interpreter. It is part of the routine for clearing the screen.

A contains Bank switch data, HL contains the start address of the screen.

NB – HL must contain the relative offset from the top left of the screen.

BLOCKS AND SWITCHING: AN ANALOGY

IN COMPUTER parlance the term 'blocks' is usually referred to as 'phantom reflections' in memory. Nothing to do with ghosts or mirrors, what this really means is that several addresses are considered to be the **same** address, when the address bus is not fully decoded. This is rather like ignoring the forenames of people and assuming those with like surnames are one and the same person.

In the Lynx 48K machine, bits A15 and A13 of the address bus are not decoded (ie ignored) and this means that addresses which are unique only by the setting of these bits are in fact looked upon as the same address.

The article on Bank Switching by David Barnes describes these similar addresses as 'blocks' for the sake of convenience and this is further illustrated in diagram 5.

If we looked at banks as teams of people and blocks as the surnames of people in each team, we would have the following comparison:

OUR ANALOGY		COMPUTER EQUIVALENT	
TEAM 1	**TEAM 2**	**BANK 1**	**BANK 2**
(John) SMITH	(Carol) SMITH	(8ØØØ) BLOCK A	(AØØØ) BLOCK A
(Fred) JONES	(Annie) JONES	(6ØØØ) BLOCK B	(CØØØ) BLOCK B

Thus, by ignoring the forenames we would consider the Smith of each team as the same person; the same goes for the Joneses. Similarly, by ignoring bits A15 and A13 the computer looks upon Block A as the same address, and likewise for Block B.

By comparing the analogy defined above with its computer equivalent, it should be easier to appreciate the means of manipulating the Lynx 48K memory.

Writing to a memory bank

If I were to tell Smith to memorise something, the Smith in each team would do so. If I wanted only the Smith in team 2 to hear me, I would need to cover the ears of team 1's Smith.

In computer terms, this means I would have to use the Bank Switch at FFFF to write disable Bank 1, so I could write specifically to Block A of Bank 2. Otherwise it would be written into Block A of **both** Banks (or in our

ND BANK G—)GY

by
TED KEATING

analogy, heard by the Smiths in each team).

Reading from a memory bank

If I were to ask a question of Smith,the Smiths in both teams would answer me at the same time in their own, way, making it impossible for me to understand either of them. Therefore if I wanted the answer from Smith in team 2, I would have to 'gag' team 1 before posing the question.

In computer terms, I would need to use the Bank Switch at FFFF to read disable ('gag') Bank 1, before trying to read from Block A of Bank 2.

In fact, this point is very important since the Lynx can be physically damaged by being forced to read from more than one bank at a time (excluding bank Ø which **can** be read at the same time as any other bank), as a result of the bus contention it would cause. So, machine code programmers BEWARE!!

Also, the computer differs slightly from our analogy when reading from banks, since it is essential that a copy of the coding which performs the read, must exist in each bank from which a read is likely to take place. That is, a separate copy of the machine code must exist in Bank 1 *and* Bank 2 if a read from these banks is needed.

Our analogy would depict this requirement as each team needing its own interviewer.

Finally, reading from Bank 2 (Video RAM) requires the use of Port 8Ø to select the appropriate set of colours, as described earlier and in the Bank Switching article by David Barnes.

Our analogy is only capable of explaining port 8Ø if we introduce another team 2, so that port 8Ø might be said to identify which team 2 we want to "listen" to.

DIAGRAM 5.

HOW IT LOOKS IN BITS :

ADDRESS (IN HEX)	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	AØ	MNEMONIC :
ØØØØ	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	(BLOCK A)
2ØØØ	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	(BLOCK A)
8ØØØ	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	BLOCK A
AØØØ	1	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	BLOCK A
4ØØØ	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	(BLOCK B)
6ØØØ	Ø	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	BLOCK B
CØØØ	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	BLOCK B
EØØØ	1	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	BLOCK B

SHADED BITS ARE IGNORED (NOT DECODED) BY THE LYNX 48K

NOTE: ROM MASKS OUT ADDRESSES 0000 THROUGH 5FFF, SO THESE ARE UNIQUE (EFFECTIVELY DECODED) UNLESS THE ROM IS SWITCHED OUT VIA THE BANK SWITCH. THIS DOESN'T MEAN EXTRA MEMORY BECOMES AVAILABLE TO THE USER, ONLY THAT THE CONTENT OF BLOCKS A & B WILL BE REFLECTED IN THESE ADDRESSES.

Above: Address decoding in the Lynx 48K micro-computer

Reading from Video RAM

Looking more closely at the Video RAM, each specific colour is selected for reading and writing by setting port FFFF (Bank Switch) and port 8Ø in conjunction with the addresses shown in diagram 6.

As described earlier, because before reading from Video RAM, it is essential to read disable the User RAM because the CPU can't listen to more than one bank at a time. However, now the User RAM is disabled the CPU can't execute the user code because it can't read it! This is exactly why the machine code which performs the read has to exist in the bank being read.

This is very easy for the ROM Routines which read the display, since ROM is 'reflected' in every bank by virtue of the fact that it masks out ('lives in') addresses from ØØØØ to 5FFF of each bank.

It is therefore impractical for the user to have copies of his machine code in each bank. Apart from anything else, the coding itself would appear on the display!

Finally, a little more explanation of the values shown in diagram 6, and bank switching in general.

When writing to Bank 2 (red/blue) you will generally want to read from Bank 1 and write to either Banks 2 or 3. Therefore Port FFFF must contain Ø3 (READ Ø,READ 1, WRITE 2) when writing to Bank 2, and Ø5 (READ Ø,READ 1, WRITE 3) when writing to Bank 3.

When writing to video memory the $\overline{\text{CAS}}$ must be disabled on either Bank 2 or 3. It is not enough to simply write-enable one of the banks using Port FFFF. This is because the other bank — the one you haven't write-enabled — will think it is being read from and will therefore cause bus contention, unless one of the $\overline{\text{CAS}}$es is turned off. This is very important since bus contention must be avoided at all times.

When reading green/alternative green (in Bank 3) it is actually Bank 2 which must be read-enabled. No signal exists specifically to read-enable Bank 3.

A CLOSER LOOK AT VIDEO RAM

DIAGRAM 6

ADDRESS	PART OF RAM	BANK	PORT CONTENTS FFFF	PORT CONTENTS 0080	GIVES ACCESS TO ...
C000-DFFF	BLOCK B	2	60	28	READ RED
A000-BFFF	BLOCK A	2	60	28	READ BLUE
C000-DFFF	BLOCK B	3	60	24	READ GREEN
A000-BFFF	BLOCK A	3	60	24	READ ALT. GREEN
C000-DFFF	BLOCK B	2	Ø3	28	WRITE RED
A000-BFFF	BLOCK A	2	Ø3	28	WRITE BLUE
C000-DFFF	BLOCK B	3	Ø5	24	WRITE GREEN
A000-BFFF	BLOCK A	3	Ø5	24	WRITE ALT. GREEN

THESE ARE EXAMPLES; VARIATIONS CAN BE APPLIED AS DESCRIBED IN THE BANK SWITCHING ARTICLE BY DAVID BARNES.

ROM ROUTINES

The following lists of ROM routines and System Variables will be amplified in later issues.

RESTARTS

RST 8H	Display routine
RST 10H	Checks if the character at the address pointed to by DE is " "; if so, increments DE until it is not a space.
RST 18H	Evaluates the reverse polish expression pointed to by DE as a binary integer in HL.
RST 20H	Checks if the byte after RST is the same as A; if it is not, displays a syntax error.
RST 28H	Evaluates reverse polish expression pointed to by DE in floating point in WRA1 (working register area 1).
RST 30H	Jumps to monitor TRAP routine.
RST 38H	Jumps to RAM RSTRAM (see RAM variables).

ADDRESSES

33H	Has POP HL JP (HL)
69H INBLUE	Reads byte in red/blue bank at HL into L, and H = 0.
70H INGREEN	Reads byte in green/alternative green bank at HL into L, and H=0.
CEH FONT	Gives address of bit map of character in A.
9BDH KEYDVR	Keyboard driver. Returns code of key pressed in A. Without single key entries.
0B65H RSYNC	Reads sync from cassette.
0B85H RBYTE	Reads byte from cassette into A.
0B93H WSYNC	Writes sync to cassette.
0CF2H MOTON	Turns cassette motor on.
0CFBHMOTOFF	Turns cassette motor off. NB must follow MOTON.
105EH PRTSTR	Displays string pointed to by HL and terminated by a CR.
1B98H RDN	Reads number pointed to by DE into WRA1 as floating point. (ASCII)
1C9AH PHL	Displays binary number in HL without leading zeros.
1D59H PN	Displays number in WRA1.
1ED0H FRNDS	Generates next random number in RNDNO.
1F05H ESC	Sets Z flag if ESC key is currently pressed.
202FH KEY	Calls to keyboard driver in KEYB
2132H ETEXT	Same as Basic TEXT command.
2234H PHEXHL	Prints binary number in HL.
2568H CHLDE	Compares HL, DE flags set as HL-DE.
25E2H RP	Reads ASCII pointed to by DE into reverse polish pointed to by HL.
2B1FH FNDLN	Finds line with number in WRA1. IX points to line and z flag set if found, else IX points to next highest line.
3497H FPINT	Floating point in WRA1 to integer in HL.
34C4H INTFP	Integer in HL to floating point in WRA1.
350DH ZWRA1	Clears WRA1 to 0s.
3539H PLINE	Display line pointed to by HL and terminated by 0.
3542H SWAP	Swaps contents of WRA1 and WRA2.
3561H CMP	Compares WRA2, WRA1. Flags set as (WRA2) - (WRA1).
35AEH LZERO	Loads WRA1 with floating point 0.
35B1H LNUM1	Loads number pointed to by HL into WRA1.
35BAH LNUM2	Loads number pointed to by HL into WRA2.
35BFH LONE	Loads WRA1 with floating point 1.
366AH SBT	Subtracts (WRA2) from (WRA1), stores result in WRA1.
366DH AD	Adds (WRA1) and (WRA2), stores result in WRA1.
36C8H MLT	Multiplies (WRA1) and (WRA2), stores answer in WRA1.
37B0H DIV	WRA1 = WRA1/WRA2

SYSTEM VARIABLES

INPBUF	**6ØØØ**		Buffer used for text input from keyboard.

AREA USED FOR MONITOR STACK AND REGISTER STORE

STACK	**61EE**	(2)	Stores address of Basic stack pointer (STACK) + 2 = HIMEM.
RNDNO	**61FØ**	(4)	31 bit random number seed.
HLSTORE	**61F4**	(2)	Used to store HL after a CALL.
CURRLP	**61F6**	(2)	Pointer to first byte of line Basic is currently executing.
DATAP	**61F8**	(2)	Pointer used by READ, DATA, points to end of last entry read.
OTYPE	**62Ø6**	(1)	Current output type and LINK status.
LASTDSP	**62Ø7**	(1)	Last character output to screen (used for VDU 1,n VDU 2,n).
VTYPE	**62Ø8**	(1)	Current variable type.
CONTLP	**62Ø9**	(2)	Pointer to CONTinue line. Ø if cannot continue.
RSTACK	**62ØB**	(2)	Pointer to return stack.
RSP	**62ØD**	(2)	Return stack pointer.
VTBL	**62ØF**	(2)	Pointer to variables A-Z, a-z.
ATBL	**6211**	(2)	Pointer to array variables A-Z, a-z.
STBL	**6213**	(2)	Pointer to string variables.
FTBL	**6215**	(2)	Pointer to function evaluation table.
TTBL	**6217**	(2)	Pointer to non-command token table.
XTTBL	**6219**	(2)	Pointer to command token table.
XITBL	**621B**	(2)	Pointer to Input syntax checking table.
XETBL	**621D**	(2)	Pointer to command execution table.
TOV	**621F**	(2)	Top of variables (strings and arrays).
IEXT	**6221**	(3)	Jump to EXT syntax checking.
EEXT	**6224**	(3)	Jump to EXT execution.
INPLSUB	**6227**	(3)	RAM call from line input routine.
EXECSUB	**622A**	(3)	Called before execution of every line.
ZFLAG	**622D**	(1)	Number flag gives TRAIL and ROUND status.
AUTOFL	**622E**	(1)	Auto line numbering on/off flag.
CRST	**622F**	(2)	Pointer to message used to generate CR (ØDH) in display driver.
SOB	**61FA**	(2)	Points to start of Basic.
EOB	**61FC**	(2)	Points to end of Basic.
POLBUF	**61FE**	(2)	Pointer to buffer used for conversion to internal language.
PRINTD	**62ØØ**	(2)	Address of display driver.
LPRNTD	**62Ø2**	(2)	Address of printer driver.
KEYB	**62Ø4**	(2)	Address of keyboard driver.
KSMS	**6231**	(2)	Single key entry.
RPTDLY	**6233**	(2)	Repeat delay on keyboard.
LASTK	**6235**	(1)	Last character from keyboard.
STATUS	**6236**	(1)	Used by keyboard driver.
RPT	**6237**	(2)	Length of time between key repeats.
SHLKT	**6239**	(1)	Used by shift lock.
KTBL	**623A**	(26)	26 bytes storing tokens for shorthand. Only command tokens allowed. In order A-Z.
CURSORX	**6254**	(1)	X component of cursor (Ø-126).
CURSORY	**6255**	(1)	Y component of cursor (Ø-247).
WINDST	**6256**	(4)	Window size.
CURSTAT	**625A**	(1)	Cursor on/off.
INKST	**625B**	(1)	Stores ink colour.
PAPST	**625C**	(1)	Stores paper colour.
FLASH	**625D**	(2)	Cursor flash rate.
INPCUR	**625F**	(2)	Cursor characters.
EXFLAG	**6261**	(1)	TRACE, SPEED on/off store.
GLINE	**6262**	(3)	Jump to line draw routine.
GOLDX	**6265**	(2)	Graphic cursor position X.
GOLDY	**6267**	(1)	Y co-ordinate of graphics cursor.
GNEWX	**6268**	(2)	Stores new X co-ordinate before LINE draw.

GNEWY	626A	(1)	Stores new Y co-ordinate before LINE draw.
PROTST	626B	(1)	Protect store.
OUTB	626C	(3)	Jump to routine which outputs byte with data in A, mask in C and relative displacement from top left in HL. Uses colour selected by INK and PAPER.
CHRTBL	626F	(2)	Pointer used to generate characters 32-127.
GPHTBL	6271	(2)	Pointer used to generate characters 128-255.
BPERL	6273	(2)	Bytes per line: 20H normally, 40H in double height mode.
MASK	6275	(1)	Mask used by display driver: change with VDU 20/21.
FLIGHT	6276	(3)	Jump to lightpen function.
FJOY	6279	(3)	Jump to joystick function.
FUSERØ	627C	(3)	Jump to USERØ function.
FUSER1	627F	(3)	Jump to USER1 function.
FUSER2	6282	(3)	Jump to USER2 function.
FUSER3	6285	(3)	Jump to USER3 function.
ERRAM	6288	(3)	Called during error.
SLFRAM	628B	(3)	Called when line feed occurs.
BLUBNK	628E	(2)	Pointer to BLUE bank.
REDBNK	629Ø	(2)	Pointer to RED bank.
GRNBNK	6292	(2)	Pointer to GREEN bank.
NMIRAM	6294	(3)	Jump here on NMI.
RSTRAM	6297	(3)	Jumps here on RST 38H.
ERRTBL	629A	(2)	Pointer to error messages.
CASLEV	629C	(1)	Cassette input threshold level.
COARSE	629D	(1)	Coarse adjustment on cassette speed.
FINE	629E	(1)	Fine adjustment on cassette speed.
SYNCLEN	629F	(1)	Length of sync.
PLYLEV	62A0	(1)	Voltage reference for comparator. Used by LOAD instruction.
CASTBL	62A1	(2)	Pointer to table used to generate wave.
RBIT	62A3	(3)	Jumps to read bit routine.
WBYTE	62A6	(3)	Jumps to write byte routine.
PP	62A9	(3)	Jump to print a polish expression.
NTP	62AC	(3)	Jump to normal to internal language conversion routine.
EE	62AF	(3)	Jump to evaluate expression.
DBASLN	62B2	(3)	Jump to display basic line.
DEBUG	62B5	(3)	Jump to SPEED and TRACE routines.
VIDEO	62B8	(3)	Jump to routine which outputs character in A to current cursor position.
LINP	62BB	(3)	Jump to line input routine.
VRPTS	62BE	(3)	Jump to string variable pointer routine.
SPEEDST	62C1	(1)	Used by SPEED.
STORE	62C2	(2)	General purpose store.
OLDESC	62C4	(1)	ESC key roll over.
OLDKEY	62C5	(1Ø)	Rest of keyboard roll over.
WRA4	62CF	(5)	WORKING REGISTER AREA 4
WRA3	62D4	(9)	WORKING REGISTER AREA 3
WRAE	62DD	(3)	WORKING REGISTER AREA 2 EXTENDED.
WRA2	62EØ	(5)	WORKING REGISTER AREA 2
WRA1	62E5	(5)	WORKING REGISTER AREA 1
SYNL	62EA	(1)	
SYNP	62EB	(1)	
SYNC	62EC	(1)	
NPM	62ED	(1)	All used by NTP.
SBC	62EE	(1)	Space before command on line listing (used for indentation).
PROCST	62EF	(2)	Pointer to first DEFPROC.
EDST	62F1	(2)	EDIT buffer driver.
AUTOST	62F3	(1Ø)	Auto line number, increment.
PRTBUF	62FD	(1Ø)	Used during printing number.
POLBUFS	63Ø7	(256)	Internal language buffer.
STRBUF	64Ø7	(256)	String calculation buffer.
CSTACK	65Ø7	(256)	Calculator stack.
RSTACKS	66Ø7	(256)	Return stack.
VTBLS	67Ø7	(53*5)	Variable table.
ATBLS	681Ø	(53*4)	Array table.
STBLS	68E4	(1Ø4)	String table.
DFTMRK	694C.	(1)	Carriage return used in READ, DATA.
SOBS	694D	(1)	START OF BASIC PROGRAM.

LOOPING THE LOOP ON THE LYNX

I have experienced a couple of problems with my Lynx.

On three separate days I have programmed in a simple game from a book called 'Computing is Easy', written for schools. The program includes this sequence:—

```
40  FOR Q=1 TO 9 STEP 1
50  LET A(Q)-INT(RND*X)+1
60  IF Q+1 THEN GOTO 100
70  FOR R=1 TO Q STEP-1
80  IF A(Q)=A(R) THEN
    GOTO 50
90  NEXT R
100 NEXT Q
```

Three times out of five when the program is RUN the error message 'NEXT without FOR in line 100' comes up. The other times the program runs properly. This seems to indicate that some connection is not being made correctly.

From the book 'Basic BASIC' by Donald Monro I typed in a program to display Pascal's Triangle. Omitting 'REM's the program reads:—

```
FOR N=0 to 9
LET C=1
?TAB(20-3*N);C;
FOR R=1 TO N
LET C=C*(N-R+1)/R
?TAB(20-3*N+6*R);C;
NEXT R
?
NEXT N
```

This program runs correctly — except for the first row. This should be simply '1' in the top centre of the screen. But with the Lynx it appears as '1 0'. I cannot understand why the '0' appears, and I would like to know how to eliminate it. Have you any suggestions?
R White
Potters Bar
Hertfordshire

These program problems illustrate the difficulties you can face in translating from one implementation of Basic to another.

In the case of the first program, you are jumping out of a FOR... NEXT loop. Some computers will allow you to do this — although their internal workspace becomes cluttered and this can cause problems later in the program — but the Lynx is very strict: if you exit a FOR... NEXT loop without passing through the NEXT it tells you: 'FOR without NEXT in line...'. It doesn't happen every time because of the RND in line 50; sometimes you jump out of the loop, sometimes you don't.

In the second program, you have a loop which runs from 0 to 9. On some Basics if N=0 the loop will not be

CURSOR DEFINITION SCROLLING

Last week I bought a Lynx and I have a couple of queries.

If you get the cursor backspacing (continually), how can you stop it, without switching the machine off and losing any program typed in?

Is there any equivalent to SCROLL as per Spectrum?
B Moody
Whitley Bay
Tyne and Wear

If you have defined the cursor as a backspace ignore what is happening on the screen — the computer will still respond to whatever you type.

Press

[RETURN]

then type

CCHAR &205F [RETURN]

The cursor should return as a ____.

If by SCROLL on the Spectrum you mean halting the display during a LIST, you can do this by holding down a [SHIFT] key: it will continue once you release the key.

If you want to scroll the display, read George Kendall's article on page 4.

LET THE LYNX DO THE SELLING

As salesmen rarely know much about Basic and are often very busy, particularly on Saturday mornings, how about a sales demo program to be used in the shop: 'Hello I am the Lynx — would you like to see what I can do — choose one or all from my menu.' Then a demonstration of maths, graphics, simple business application, bar charts, graphs, pie charts, etc. — then looping back to the

more letters on next page

FEEDBACK

It would be very silly of us to ignore the wealth of information which users will gather from their experience on the Lynx. We will all benefit from hearing it.

While it is not always possible to respond with personalised technical detail, all your comments are gratefully considered for our Newsletter.

■ DO YOU HAVE ANY TIPS WE CAN PASS ON IN OUR NEXT NEWSLETTER?

■

start. I think it might help to make it sell.

So far I am pleased with mine, but as a very mature, but very humble beginner in Basic my greatest error at the moment is Syntax Error!

I wish that

[CONTROL]Q[RETURN]

would work for me. The same goes for

[CONTROL]E[RETURN]

It's frustrating to have to start the whole line again!

Still it's a good machine, well made, and it's British.

M Welfare
Durham

We like your program idea.

On the subject of editing, thank you for drawing our attention to our blunder in the manual. The section on page 36 should read:

[CONTROL]Q

and

[CONTROL]E

that is, no return!

You may also be having two other problems. First, (CONTROL) and Q or E must be pressed simultaneously, i.e. the [CONTROL]key must be held down while typing Q or E.

Second, when a computer checks lines for syntax errors as you enter them, it does not accept faulty lines into its memory. The Lynx — unlike many other machines — allows you a second chance. It puts the line into a temporary storage area, so that you can call it back with [CONTROL]Q; but if you type anything other than that command, the temporary copy is destroyed. So if you make a syntax error, don't notice it, and carry on typing new lines, then, although it still appears on the screen, the line will not be stored in the computer and cannot be recalled for editing.

UNLOCKING THE LYNX'S GRAPHICS

I'd like to take this opportunity to say what an excellent machine you have produced. Can you clear up a few queries for me?

The 'L' key has a 'G' printed on it; is there some way this can be corrected without having to send the machine away?

The '\@' key when shifted produces a '£' not a ' i'; is this as it should be?

Defining my own characters as shown on page 64 of the manual seems to corrupt characters 224-249; can this be avoided?

M Haas
Wool
Dorset

The example shown on page 64 of the manual shows how to reset the pointer GRAPHIC to point to extra self-defined graphics characters. It is not mentioned that having done this I can no longer access the standard graphics characters. How is GRAPHIC set back to the original place? This was brought to my attention because the cursor was redefined after running the example to three large horizontal bars.

Also the values displayed when using functions GRAPHIC and ALPHA do not change after POKEing new values in, although the character set is changed — please explain.

G Jenkins
Shepperton
Middlesex

The \@ keytop was a design error. It should show £@.

In addition, a few Lynxes have the wrong keytop on them. You can either contact your dealer or write directly to Camputers, stating which key you need. You can swap the keytops easily.

On the subject of User Defined Graphics: when you define graphics characters, you don't corrupt the pre-programmed characters — unless you redefine characters 224-249 — you just shift the pointer to them, so the computer can't find them. If, when you power the machine up, you

DPEEK (GRAPHIC)

and make a note of the value, you can later reset the pointer with

DPOKE GRAPHIC,value

Similarly, because the cursor character is a graphics block it changes too — the pointer now points to a different character. If you change the cursor to one of the standard ASCII characters, 33-127, it will be unaffected. That's the main reason for the CCHAR command.

GRAPHIC is a pointer to a location; when you POKE graphic, the value goes into the location, so the value of GRAPHIC is not changed.

LOWDOWN ON MONITORS – AND GUARANTEES

Congratulations on an excellent (and different) personal micro. A TV set cannot do justice to the graphics and I intend to purchase an RGB monitor — do you have any recommendations?

Is the guarantee invalidated if the power supply cover is removed and ventilated or fitted into a larger box?

Alan White
Walsall
Staffordshire

By all means purchase an RGB monitor to do justice to Lynx graphics. But don't forget that you will need a special lead to connect the monitor to the computer. Reputable computer shops should be able to supply you with the lead, if you explain what you need it for.

Camputers intend to market their own RGB monitor later in the year, which will be available — with a lead — for a competitive price.

We do not recommend that you attempt to remove the cover of your Lynx's power supply, or try to ventilate it. For a start it is dangerous, and secondly, it will invalidate your guarantee.

USER GROUPS

EVER SINCE the public first became aware of computers, the prophets of doom and despondency have been predicting that artificial intelligence would outstrip mere human ingenuity and that we would all become slaves to our computers.

It didn't take much imagination to conjure up the image of the wild-eyed computer freak whose only interface with the world was his terminal.

But, as newcomers to the micro world have found out, there is no substitute for traditional human communication — or talking, as it is usually called.

So that is why we're keen to encourage the growth of Lynx user groups — circles of people who live near each other and who want to delve deeper into the labyrinths of machine code and hexidecimal numbering systems. As the owner of a Lynx you have that world literally at your fingertips, and the voyage of discovery will be far more revealing if you have fellow travellers.

If you are interested in joining a Lynx user group, please write to: The Editor, Lynx User, 33a Bridge Street, Cambridge CB2 1UW. We will then put you in touch with other Lynx users in your area.

The following people have already written in saying that they are interested in forming user groups:

Colin Baxter, 4 Low Ash Avenue, Wrose, Shipley, West Yorkshire (tel: Bradford 596239).

Bob Jones, 209 Kenton Lane, Kenton, Harrow, Middlesex (tel: 01-907 3406).

Richard Lamude, 78 Warren Drive, Hornchurch, Essex.

LYNX 48K MANUAL INDEX

Many Lynx owners have pointed out — quite correctly — that the machine's manual does not have a general index. To rectify the situation, cut this page out and keep it with your manual.

Many thanks to Mr R. A. Wilson of Morecambe for compiling this index.

LYNX 48K I/O MAPS

LYNX INTERNAL INPUT/OUTPUT MAP

DESCRIPTION	DATA								I/O ADDRESS				DATA DIRECT
	D7	D6	D5	D4	D3	D2	D1	D0	A15-A12	A11-A8	A7-A4	A3-A0	
BANK SWITCH (Refer to article on Bank Switching)	RDEN4	RDEN2	$\overline{\text{RDEN1}}$	$\overline{\text{RDEN0}}$	WREN4	WREN3	WREN2	$\overline{\text{WREN1}}$	**1* OR IN 64K VIDEO CASE ****	**** ****	*111 *111	1111 1111	WRITE
KEYBOARD PORT	SHIFT	ESCAPE	↓	↑	SHFT LK			1	****	0000	10**	*00*	READ
			C	D	X	E	4	3	****	0001	10**	*00*	READ
		CONTROL	A	S	Z	W	Q	2	****	0010	10**	*00*	READ
			F	G	V	T	R	5	****	0011	10**	*00*	READ
			B	N	SPACE	H	Y	6	****	0100	10**	*00*	READ
			J		M	U	8	7	****	0101	10**	*00*	READ
			K		,	O	I	9	****	0110	10**	*00*	READ
			;		.	L	P	0	****	0111	10**	*00*	READ
			:		/	I	@	-	****	1000	10**	*00*	READ
			→		RETURN	←	]	DELETE	****	1001	10**	*00*	READ
AUDIO VISUAL CONTROL PORT (Refer to article on Bank Switching)	0 (must be)	Line Blanking Access Control	CPU Access	0=Green 1=Alt Green	Bank 3 $\overline{\text{CASEN}}$	Bank 2 $\overline{\text{CASEN}}$	CASSETTE MOTOR 1=ON 0=OFF	SPEAKER 1=ON 0=OFF	****	****	10**	*00*	WRITE
SERIAL PORT (UART)	PINS 5,33,19	PINS 6,32,22	PINS 7,31	PINS 8,30	PINS 9,29	PINS 10,28,13	PINS 11,27,15	PINS 12,26	PIN 4	18	10**	*01*	READ
	Note also following pins connected to +5V: Pins 1,34,35,37,38,39 And following to ground 36,3 STANDARD BAUD RATE 2,400								PIN 16		10**	*10*	READ
									PIN 23		10**	*01*	WRITE
6845 VIDEO DISPLAY GENERATOR (Refer to article Inside the 6845 CRTC)	Register select 0 to 11 (Hex) — Consult 6845 Data Sheet								****	****	10**	*110	WRITE
	Register Data								****	****	10**	*111	WRITE
6 BIT D/A OUTPUT (LOUDSPEAKER)	0 (must be)	NOT USED	MSB B5	B4	B3	B2	B1	LSB B0	****	****	10**	*10*	WRITE

NOTE: * = 'don't care'. The 'OUT (BC)' instruction must be used for the bank switch and IN(BC) for the keyboard. The 6 BIT D/A output is available on pin 4 of the cassette socket.

This diagram shows connections to the various key internal ports on the Lynx. On the serial port we have shown the pin connections, so that, with the aid of a data sheet, the machine code programmer will be able to see how the UART is being utilised.

LYNX EXTERNAL INPUT/OUTPUT MAP

DESCRIPTION	DATA								I/O ADDRESS								DATA DIRECTION
	D7	D6	D5	D4	D3	D2	D1	D0	A7	A6	A5	A4	A3	A2	A1	A0	
DISK	1793 R0 — R3								*	1	0	1	0	0	00 to 11		READ
	1793 R0 — R3								*	1	0	1	0	1	00 to 11		WRITE
	1=125ns 0=250ns	1=No PRECOMP	NC	EPROM ENABLE	MOTOR	SLIDE	SEL 1 — 4		*	1	0	1	1	0	0	0	WRITE
JOYSTICK	SPARE	SPARE	FIRE	SPARE	RIGHT	LEFT	DOWN	UP	*	1	1	1	1	0	1	0	READ
	SPARE	SPARE	FIRE	SPARE	RIGHT	LEFT	DOWN	UP	*	1	1	1	1	0	1	1	READ
PARALLEL PRINTER INTERFACE	PRINTER STATUS WORD					SEL	PE	BUSY	*	1	1	1	1	1	0	0	READ
	PRINTER INITIALISE SIGNAL								*	1	1	1	1	1	0	1	R/W
	DATA 8	DATA 7	DATA 6	DATA 5	DATA 4	DATA 3	DATA 2	DATA 1	*	1	1	1	1	1	1	0	WRITE

* = 'Don't care'

NOTE: * = 'Don't care'. When a port is specified as write, then a read from that port address will also effect a write, and vice versa, a write to a read port results in data being put onto the bus.

This diagram shows the ports used to connect peripherals to the Lynx. The information shown is provisional and subject to alteration. Note: Additional circuitry is required before the Lynx can be hooked up to add-on hardware. The disk drive, for example, will house its own controller.

HELP US TO HELP YOU . . .

WIN A FREE DISK DRIVE

THE MORE we know about what Lynx users want, the better we'll be able to help you. So will you help us by answering the questions listed here and returning the form to us as soon as possible? **Don't forget to fill in your name and address, because everyone who returns a completed questionnaire will be entered in a prize draw for a FREE LYNX DISK DRIVE.**

Please tick which of the following items you'd buy to go with your Lynx first, which second and so on:

	1st purchase	2nd purchase	3rd purchase	Eventual purchase
Serial printer interface				
Parallel printer interface				
Mini printer (no graphics)				
Mini printer (more expensive but *with* graphics)				
Single disk drive				
Twin disk drives				
Light pen				
Modem				
Joysticks				
Other languages (specify):				
Extended Basic				
Utilities: Assembler				
Disassembler				
Colour monitor				
Mono monitor				

We are currently developing an Extended Basic ROM pack. What features would you like it to include?

Which of these types of software are you most likely to buy? (please tick as appropriate).

	Very likely to buy	Quite likely to buy	Would consider buying it at some time	Quite unlikely to buy	Very unlikely to buy
Primary education programs					
Secondary education programs					
Further education programs					
Educational games					
Adventure games					
Arcade games					
Home management programs					
Small business programs					
Word processing					

Is the Lynx the first computer you have owned?

Yes
No

If no, which computer(s) have you owned before?

____

____

NAME: ______________________

ADDRESS: ______________________

____

____

____

Who else uses your Lynx?

AGE: ____________ **SEX:** ____________