NILUG NEWS

NATIONAL INDEPENDENT LYNX USER GROUP

Magazine for **LYNX** Users

Volume 1. Issue 5.

NILUG NEWS

VOLUME 1. ISSUE 5.

CONTENTS

EDITORIAL

I stated in my review of ZEN that there is no APPEND command. This is not true. The LOAD command actually appends the cassette file onto whatever is there. Consequently an APPEND command isn't necessary. Apologies to Laurie Shields for my mistake.

Another couple of members would like to contact local users.
John Batty, 72 Latimer Road, Cropston Leicester, LE7 7GN.
C.E. (Bob) Dinsbach, Rossinihof 18, 2402 EG, Alphen A/D Rijn, Netherlands. Home 01720 20977, Work 070 614971

WARNING
48K to 96K RAM UPGRADE

In Issue 4 there was an article on upgrading the user RAM of a 48K Lynx from 16K to 64K. It stated that 4164 RAM chips are needed. It has been pointed out to me that there are two types of 4164 RAMs. There are 7-bit and 8-bit refresh versions and there is no way to tell which is which by looking at the chips. The Lynx requires chips with a 7-bit refresh. I have been told that the TEXAS versions are 8-bit refresh but there may be others. If any one has any more information about which makes work (or which don't) please let me know.

The upgrade article provoked a reaction from Camputers. They would like to point out that your guarantee will become null and void if you upgrade your Lynx.

SUBSCRIPTIONS (6 ISSUES)
UK £9.00 OVERSEAS £12.00
BACK ISSUES £1.50 EACH
Available from:-
NILUG
53 Kingswood Avenue, Sanderstead, South Croydon, CR2 9DQ

BACK ISSUES

LETTERS

I have had several letters which contain quite useful pieces of information or ask sensible questions but for one reason or another could not be published without some re-touching. Colin Clayman wrote a letter which was mainly about Cursor Interrupts. Along with the letter he gave a rather useful routine which displays instructions at any time (even if BASIC is doing something else). I feel that its a very good procedure but that it needed a little modification to make it safer and an explanation of how it works.

It works by printing instructions in the alternate green video RAM. You can then display this by disabling the red, blue and normal green banks and enabling the alternate green bank. This is done by an interrupt routine held in a code line. As I explained in the article on Using the BREAK Key (N.N. Issue 3) you have to prevent interrupts during video access. This is done by two further routines held in code lines (20 and 30) which disable interrupts for the display driver and the line draw routine. IT IS VERY IMPORTANT TO NOTE THAT ONCE YOU HAVE ENABLED THESE ROUTINES (lines 10120,10130 & 10140) YOU MUST NOT EDIT LINES 10, 20 or 30 OR PUT ANY MORE CODE BETWEEN THESE LINES. The reason for this is that if you do you will move the location of these new routines and the Lynx will 'fall over'. As an example if a line 15 were inserted the display driver routine (line 20) would be 'lost' and the Lynx would crash.

```
10 CODE  F5 3E 14 D3 80 AF
D3 84 F1 ED 56 FB C9
20 CODE  F3 CD A4 06 FB C9
30 CODE  F3 CD BE OF FB C9
40 REM DISPLAY INSTRUCTIONS
50 PROC HELP
60 REM THE MAIN PROGRAM (dummy)
100 FOR I=1 TO 20
120   PRINT I
140 NEXT I
160 END
10000 DEFPROC HELP
10040 TEXT
10050 DPOKE &6292,&8000
10060 CLS
10070 PROC INSTRUCTIONS
10080 DPOKE &6292,&C000
10090 OUT &0086,14
10100 OUT &0087,&003F
10110 DPOKE &6294,&45ED
10120 DPOKE &6298,LCTN(10)
10130 DPOKE &6200,LCTN(20)
10140 DPOKE &6263,LCTN(30)
10150 CALL LCTN(10)
10160 LET K=GETN
10170 PROTECT BLACK
10180 INK WHITE
10190 ENDPROC
10200 DEFPROC INSTRUCTIONS
10210 PRINT "The first line
of the instructions"
10220 PRINT "The second line
..... etc"
10230 ENDPROC
```

I have had several letters asking the following questions:-

Will software written for the 48K Lynx run on a machine upgraded to 96K? The 48K to 96K RAM upgrade will make no difference to how the software will work. It is the new EPROMS (BASIC version 2) which could cause problems. If software makes direct calls to routines in the EPROMS and the location of these routines change from version to version then the chances are that the software will not run. If the routines are in the same location then there is no problem. If the software makes indirect calls to routines via the RAM vectors then there will be no problem because even if the routine location changes the vector will also change.

To conclude then, it all hangs upon whether software makes direct calls to routines in the EPROMs and whether the locations of the routines have changed. Consequently I would expect most software to run on either machine.

Several people seem to be under the impression that the 'INTERFACE' on the back of the Lynx is a 'port'. Consequently they ask how do you send data to it. The INTERFACE is not a port and the pins can't be used directly. If you don't understand the pin designations my advice is to leave well alone. You need some extra hardware (joystick interface, parallel printer interface etc) to be able to use the socket.

It seems that some of you don't know how to read and make use of the I/O maps published in Lynx User Issue 1 (page 22). There are four main columns (or sets of columns) DESCRIPTION, DATA, I/O ADDRESS and DATA DIRECTION. The DESCRIPTION should need no explanation. The DATA field explains how the external hardware is connected to the ports. Since there are eight bits (binary digits) per port the table gives the connections for each bit. As an example the first line of the keyboard port describes how some of the keys are connected. The SHIFT key is connected to bit 7 (D7), the ESCAPE key is connected to bit 6 (D6), etc. The I/O ADDRESS column contains the 16-bit address for the port. It is slightly complicated by the fact that not all the address lines are decoded (ie used). Consequently some of the bits may be set as either 0 or 1. These are flagged by asterisks (*). The easiest way of dealing with these is to consider them as zeros. To read the address is quite straightforward as long as you take it in 4-bit nibbles. The first address is '**** **** *111 1111'. If you change the asterisks to zeros you get '0010 0000 0111 1111'. Now convert each of the 4-bit nibbles into hex (see Machine Code for Beginners - Part 1 if req). The address then becomes 207FH.

The joystick interface is now available for the Lynx. The 48K Lynx does not have a JOYSTK command. Can you use joysticks on the 48K machine or do you have to upgrade to the 96K machine? The joysticks are connected to ports 7AH and 7BH. You can read the status of the ports with INP(&7A) and INP(&7B). Now that you have learnt how to read the I/O Maps you will see that the FIRE button is connected to bit 5 and RIGHT, LEFT, DOWN and UP are connected to bits 3 to 0 respectively. All input bits are high (ie set to 1) if there is no input. Hence the following table gives the input values associated with the various operations.

OPERATION	INP(&7A) or INP(&7B)		
	BINARY	HEX	DECIMAL
No operation	1111 1111	FF	255
FIRE	1101 1111	DF	223
UP	1111 1110	FE	254
DOWN	1111 1101	FD	253
LEFT	1111 1011	FB	251
RIGHT	1111 0111	F7	247
UP & FIRE	1101 1110	DE	222
DOWN & FIRE	1101 1101	DD	221
LEFT & FIRE	1101 1011	DB	219
RIGHT & FIRE	1101 0111	D7	215
UP & LEFT	1111 1010	FA	250
UP & RIGHT	1111 0110	F6	246
DOWN & LEFT	1111 1001	F9	249
DOWN & RIGHT	1111 0101	F5	245
UP,LEFT & FIRE	1101 1010	DA	218
UP,RIGHT & FIRE	1101 0110	D6	214
DOWN,LEFT & FIRE	1101 1001	D9	217
DOWN,RIGHT & FIRE	1101 0101	D5	213

Dear Mr Poate,

Thank you very much for Issue 3 of Nilug News. It makes very interesting reading and I look forward to future issues.

My main purpose in writing is to give you a list of Lynx Software that I have compiled from various sources. This is in response to the request in your Editorial for such information. I hope I am not too late in supplying it.

The list is as complete as I can make it and although you will no doubt know most of the items some of the software may have escaped your notice. There are more tapes available than one might expect but I cannot affirm that it is all still available. Nor can I guarantee that the list is a complete one.

Yours sincerely, A. Rendall.

LYNX SOFTWARE

SUPPLIER	TYPE	TITLE	PRICE	COMMENTS
Abersoft	G	Mazeman	£4.95	
Acme Software	G	Connect Four	?	Which Micro Nov 83
Active Software	U	Character Generator Aid	£6.00	Which Micro Jul 83
	G	Execution	£6.00	
	G	Fruit Machine	£6.00	
Albasoft	E/U?	Music Master	£4.95	Which Micro Oct 83
G.L.Banks	G	Nine Games	£3.50	Small adds Personal computer news 15/4/83
Basic Concepts	G	Happy Landings	£6.95	Personal Computer News 29/4/83
	U	Teach Yourself BASIC	£6.95	" " " "
Colin Baxter	U	Lynx Disassembler	£5.00	Small adds Popular Computing Weekly 12/5/83
Bond Systems	U	Disassembler	£5.00	Popular Computing Weekly 12/5/83
Bus-Tech	G	Jumping John	£7.00	
	G	Maze of Doom	£6.00	
	G	Roborun	£6.00	
	G	Ynxvaders	£7.00	
Camsoft	G	Adventure Quest	£9.90	
	U	Assembler	£29.95	
	G	Colossal Adventure	£9.90	
	G	Connect Four	£4.95	
	G	Dam Busters	£6.95	
	G	Dungeon Adventure	£9.90	
	U	Election Analysis	£7.90	
	G	Games Pack Three	£7.95	
	G	Games Pack Four	£7.95	
	G	Gobble De Spook	£9.90	
	G	Golf	£7.95	
	G	Hangman	£7.95	
	G	Lynx Invaders	£9.90	
	G	Mined Out	£7.95	
	G	Monster Mine	£7.95	
	G	Moonfall	£7.95	
	E/U?	Music Master	£7.95	
	E/G?	Numerons	£9.90	
	G	Power Blaster	£9.90	
	G	Protector	£6.95	
	G	Racer	£6.95	
	G	Spannerman	£9.90	
	G	Sultan's Maze	£7.95	
	G	3D Monster Craze	£9.90	
Clive Carter	G	Tape 1	£5.00	Popular Computing Weekly 31/3/83
	G	Tape 2	£5.00	
Cascade Software	G	50 Games	£9.95	
A.Cowell	G	Space Crabs	£4.25	Small adds Popular Computing Weekly 7/7/83
D.T.Microsystems	U	Disassembler	£8.95	Small adds Your Computer Dec 83
Daletek	U	Keytek	£5.50	Teaches touch typing; Your Computer Dec 83
Durell Software	G	Dambusters	£6.95	
F.L.Software	U	Coder	£7.50	
	G	Roader	£5.95	
	G	Toeder	£5.95	
Fiddament	G+U	Computer Attack	£5.00	
Gem Software	G	Games Pack One	£7.95	
	G	Games Pack Three	£7.95	
	G	Gempack Four	£7.95	

SUPPLIER	TYPE	TITLE	PRICE	COMMENTS
	G	Golf	£7.95	
	G	Monster Mine	£7.95	
	G	Oh Mummy	£7.95	
	G	Pontoon	£7.95	
	G	Spannerman	£7.95	
	G	Sultan's Maze	£7.95	
Harvest Software	G	King Solomon's Mines	£5.95	Small adds Your Computer Dec 83
Hilton Comp.Services	U	Personal Banking	?	"soon to be available" Home Computer Weekly 8/3/83
Imperial	G	Computer Attack	£5.00	Your Computer 8/83; Same name & price as Fiddament
M.Lawson	E	Englands History	£5.00	Small Adds Personal Computer News 1/12/83
	E	French	£5.00	
Level 9 Computing	G	Adventure Quest	£9.90	
	G	Colossal Adventure	£9.90	
	G	Dungeon Adventure	£9.90	
	G	Snowball	£9.90	
Lynxman	U	Scroller	£10.00	2764 EPROM Chip
	G	Vorlon Invaders	£5.00	
A.Miller	G	Zombie,Panic & Deathball	£5.00	Popular Computing Weekly 7/7/83
Multisoft	G	Everest	£7.95	
D.Naik	U	Lynx Cassette File Handler	£16.00	Same address as RAD Systems
Origin Software	U	Disassembler	£4.00	Small adds Personal Computer News 20/5/83
Peach Software	G	3D Labyrinth	£4.95	Small adds Personal Computer News 25/8/83
W.Pope	U	Disassembler	£4.00	Popular Computing Weekly 12/5/83
Quazar Computing	G	Chancellor	£4.75	
	U	Disassembler	£4.75	
	G	Labyrinth	£4.75	
	G	Othello	£4.75	
	G	Reversals	£4.75	
	G	Spacetrek	£4.75	
	G	The Worm	£5.95	
RAD Systems	U	Cassette File Handler	£6.95	Personal Computer News
	U	Home Accounts	£6.95	
	G	Space Invaders	£6.95	
	U	Telephone & Address Database	£6.95	
	G	Voodoo Treasure Island	£6.95	
Romik Software	G	Atom Smasher	£9.99	Which Micro? Dec 83
	G	Floyd's Bank	£9.99	
	G	Moons of Jupiter	?	Personal Computer News
	G	Power Blaster	£9.99	Which Micro? Dec 83
	G	3D Monster Craze	£9.99	Which Micro? Dec 83
Seven Stars Publishing	U	Moder-80	£6.95	Personal Computer News 24/11/83
Sharads Software	G	Grid Attack	£4.95	Reviewed Which Micro? 12/83
Laurie Shields Software	U	Zen	£22.50	
Sian Softare	G	Racer	£5.95	Reviewed Which Micro? Dec 83
	G	Protector	?	
T.T.Titchmarsh	U	Disassembler	£5.00	Small adds Personal Computer News 12/5/83
Viscount Software	U	Teach Yourself BASIC	£6.95	"available soon" Personal Comp. News 27/5/83
Willowsoft	G	Lynx Games Vol 1.	£5.00	Which Micro? Nov 83

TYPES

E = Educational; G = Games; U = Utility

SUPPLIER	ADDRESS
Abersoft	7 Maesfallem, Bo St., Dyfed SY24 5BA 0970 828851
Acme Software	Address not known 051 236 8062
Active Software	117 Icknield St. Birmingham,B18 6RZ
Albasoft	180,Terregles Ave.Glasgow G414RR
G.L.Banks	25 Woodfield Park,Walton,Wakefield WF2 6PL
Basic Concepts	P.O.Box 46 Leatherhead Surrey KT22 7XF Tel 0372 378694
Colin Baxter	4 Low Ash Ave. Wrose, Shipley W. Yorkshire BD18 1JJ
Bond Systems	15 Belmont Rd. Harrogate N.Yorkshire HG2 0LF

SUPPLIER	ADDRESS
Bus-Tech	19 Landport Tce. Portsmouth Hants P01 2RG
Camsoft	33A Bridge St. Cambridge CB2 1UW 0223 315063
Clive Carter	110 Llancayo St. Bargoed Mid-Glam CF8 8TP
Cascade Software	Cascade House Bargans Lane Llandogo Gwent S.Wales NP5 4PA
A.Cowell	61 Whitney Rd. Leyton London E10
D.T.Microsystems	287 Orphanage Rd. Sutton Coldfield W.Midlannds B72 1BH
Daletek	8 Smithfield Close Ripon N.Yorkshire
Durell Software	Castle Lodge Castle Green Taunton TA4 1AB
F.L.Software	13 St. Ronans Ave. Southsea Hants. PO4 0QE
Fiddament	7 Elmtree Ave. W. Bridgford Nottingham NG2 7JU
Gem Software	Unit D The Maltings Station Rd. North Sawbridgeworth Herts. 0279 723567
Harvest Software	West Winds Tredragon Rd. Mawgan Porth Newquay Cornwall TR8 4DH
Hilton Comp. Services	Address not known
Hisoft	13 Gooseacre Cheddington Leighton Buzzard Beds. LU7 0SR 0296 668995
Kuma Computers	11 York Rd. Maidenhead Berks. SL6 1SQ 0628 71778/9
Imperial	Address not known
M.Lawson	"Rosemary" Woodstock Rd. Stonefield Oxford.
Level 9 Computing	229 Hughenden Rd. High Wycombe Bucks HP13 5PG
Lynxman	270 Tithe Pit Shaw Lane Warlingham Surrey CR3 9AQ
A.Miller	50 Orchard Rd. Seer Green Beaconsfield Bucks.
Multisoft	32 Redfield Hill Bristol BS15 6TQ
D.Naik	17 Devonshire Hill Lane London Stockport Cheshire
Origin Software	5 Catterwood Drive Compstall Stockport Cheshire SK6 5JT
Peach Software	37 Walker St. Earlsheton Dewsbury W.Yorkshire
W.Pope	147 Hawkhurst Rd. Brighton BN1 9EB
Quazar Computing	29 Western Rd. Newick E.Sussex
RAD Systems	17 Devonshire Hill Lane London N17 8LJ
Romik Software	272 Argyll Ave. Slough SL1 4HE
Seven Stars Publishing	15 Gloucester Ave. London NW1 7AU
Sharads Software	189 Eaton Rd. Ilford Essex IG1 2UQ 01 514 4871
Laurie Shields Software	151 Longedge Lane Wingerworth Chesterfield S42 6PR
Sian Softare	139 Roseberry Ave. Manor Park London E12
T.T.Titchmarsh	21 Blenheim Drive St. Ives Cambridgeshire PE17 4UW
Viscount Software	address not known
Willowsoft	The Willows, Eaton Bishop, Hereford.

REVIEWS

GOBBLE DE SPOOK from CAMSOFT By G.Hindle

This is a fairly good implementation of the old arcade favourite 'PAC-MAN'. Written mainly in machine code (as far as I could tell), it is neat and well thought out.

The game comes neatly packaged in the popular 'Video' type plastic box. On opening the box one finds a cassette and instruction leaflet. Although the leaflet is small it is descriptive and to the point.

To load the game one enters the baud rate ie TAPE 0 or 3, followed by RESERVE &6D70. Next comes MLOAD "SPOOK". Both baud rates loaded first time for me on side A, side B has a different recording pattern.

Once loaded the game auto runs commencing with a title page, followed by a menu which is returned to after each game. There are eight levels of play; 1 is the hardest, 8 the easiest. The controls should be easy to learn. The graphics remind me of early Spectrum graphics and although rather shaky they are effective. Sound is well used considering the limited facilities. It would be nice to be able to switch it off during the menu.

In conclusion, well thought out game with HI-SCORE position and room for three initials, good use of facilities, expensive at £10.00, value eight out of ten.

CONNECT 4 from CAMSOFT by J.Batty

This is one of the programs in the first batch released by Camsoft. It is a colourful, well-presented version of the game which I am sure eveyone knows. One good feature is that you can choose to play the computer or another person (with the computer as a referee). The computer plays a very good game but even so is beatable and it is very satisfying when you do so. The computer's response is fast (within five seconds), and if you ponder for too long it hurls (friendly) abuse at you! When you win all of his counters obtain sad little faces and they all smile when it wins. There are also a number of sound effects and tunes. I have found that as well as me (age 18) liking it much younger children greatly enjoy it, with the faces and sounds. Altogether, I think that it is very good entertainment and excellent value at £4.95

SPANNERMAN from GEM SOFTWARE by R.B.Poate

A nuclear reactor is flooding due to leaks in the cooling system. Your job is to stop the leaks and save the reactor. What could be more simple? Unfortunately the drains are blocked and the leaking coolant is flooding the reactor. Furthermore the water disturbs rats which, because they have lived in the reactor for some time, have degenerated into giant anti-matter rats. If you get too near to them you will die. The final killer is falling girders - all in all not a particularly healthy environment! You have your trusty spanner which you use to tighten the leaking joints and so stop the leaks. One last imaginative point is that if you overtighten the joints they will leak permanently and can't be repaired.

The game starts with you in the centre of a four floor reactor. The floors are connected by ladders. You move using the cursor keys and press the spacebar to make the plumber use the spanner. Additional controls are RETURN to kick the rats and Z to jump. The coordination of sound and movement is excellent.

Although the game is fun to play two things spoilt the game for me. Firstly the instructions say that you can give the mutant rats a kick with your steel capped boots. I tried many times but without success. I have since found (after many games) that the timing of the kick is rather critical and that it is better to avoid the rats. The second annoying thing is that the falling girders hit the various floor levels and change direction. You can, for example, have a girder falling vertically on to you. You take a step to the right (or left) only to find that it will hit the floor above your head, change direction and kill you. This means the getting a high score is more a matter of luck rather than skill.

On the whole an excellent game with good sound, graphics, three levels of play and a hall of fame. Overall eight out of ten (nearly nine out of ten).

OH MUMMY from GEM Software Price £7.95 by R.B. Poate

You are the leader of an expedition to explore some newly found pyramids. You search five levels of a pyramid for the Royal Mummies. Guardians protect the pyramids from intruders and these must be avoided. You must find the Royal Mummy and the Key to be able to continue to the next level. If you unearth a Guardian Mummy then it will join the other Guardians which are chasing you.

For me the game depended too much on luck. If you found the scroll, the Key and the Royal Mummy before the lurking Guardian Mummy then you stood a good chance of making a good score. If not you will lose your expedition members. Overall eight out of ten.

GAMES PACK 1 from GEM Software Price £7.95 By R.B.Poate

This games pack has three games on it and after playing SPANNERMAN and OH MUMMY I had a great deal of trouble loading it. Eventually I decided to look at the instructions. I found that whereas SPANNERMAN and OH MUMMY needed to be MLOADed these games had to be LOADed. Once I had got that sorted out the games loaded first time.

FRUIT MACHINE is the first game. Until I had played it I could never see the point of a fruit machine game - you never lost any money so whats the point? The game has the usual three reels which display the apples, pears, cherries etc. You have optional HOLD and NUDGE if you are lucky and a DOUBLE or QUIT option. You are asked how much money you want to stake and then the game ends when you run out of money or quit. If you have won the program will give you an IOU for your winnings. Its a pity you can't send this off to GEM because I staked a pound to begin with and ended up with eleven and on my second try I staked another pound and ended up with £41.30! I think that the odds are more in your favour than in real life. In spite of the fact that the game is nearly all luck and no skill I liked it (possibly because I won!).

The second game is TORPEDO and as its name suggests this is about sinking ships with torpedoes. You have a periscope view as a convoy passes by. You try to sink as much tonnage as you can. The ships have funnels and you get most points for a hit below it and less points as the hit moves further away. There are three skill levels and the level determines how many diferent speeds the ships may have. The skill comes in judging the speed of the ships and in launching your torpedoes at the right time. The game doesn't give you much time to assess the speed of the faster ships so its a game of pure skill.

The final game is MINEFIELD. You have five tanks with which to clear a minefield. They fire proximity shells which explode near mines. You control the tank with the cursor keys and fire shells with the F key. The red tanks are difficult to control at first because they don't respond to normal cursor key directions. The shells are hard to see and could have been made much bigger. I feel that this game was a make-weight becaue its not very good. You can fire a shell across a mine and it doesn't explode. You then move down the path of the shell (thinking it to be safe) only to find that you hit a mine.

Overall quite a fair tape. Value for money seven out of ten.

ROCKETMAN from Bamby Software By M.LAWSON

This programme was sent to me free by Bamby Software for being patient in waiting for 'Treasure Island'. Very decent of them I thought. There are two games in one,both aimed at children. In the first game you are required to subtract one number from another. Simple!,the numbers range from 1 to 99 and the answer is always less than 10. Now for the catches. If you are incorrect you do not get a second chance. From the time the two numbers appear a rocket moves across the screen and every time that a correct answer is received a brick is positioned in a wall,and the rocket starts again. If the rocket crosses the wall,you lose. Once the wall is large enough,another wall is started further across the screen. Eventually the wall is being built right in front of the rocket being launched. All of a sudden panic sets in and adults find that they are not as quick as they thought they were. Well written, good sound and colour,most enjoyable. Aimed at children of 10 and upwards, and adults whose mental arithmetic is rusty.

The second program Factorfire is another mental arith- metic exercise. The idea is to find the highest divisor for a given dividend. The game takes the form of a space ship moving across the screen and 9 lasers on the ground. The space craft bears the dividend and the lasers the divisor. When a number is pressed that will divide exactly,a score of 10 times that number is added to your score. If the divisor is incorrect then 10 times that number is deducted from the score. When those poor Martians get blasted a rather pleasant explosion is observed. Again good colour, good sound,good format. Simple and well thought out. Aimed at anyone old enough to be able to press a number as opposed to a letter. My 4 year old loves this game. On the easiest level of play she has learned to recognise certain numbers and knows the corresponding number to press. i.e.A number ending in 0 or 5 is always divisable by 5. If a number ends in 2 then it will be divisable by 2,etc.etc. In conclusion if you have children it is a good educational game.

```
20 REM SOLITARE by David Hall
30 LET c=32
40 PAPER BLUE
50 CLS
60 VDU 1,6,2,2,24
70 PRINT "** SOLITARE **";CHR$(25);
80 DIM B(49),C(4)
90 FOR N=1 TO 49
100   LET B(N)=1
110 NEXT N
120 DATA 1,2,6,7,8,9,13,14,36,37,41,42,
43,44,48,49
130 FOR N=1 TO 16
140   READ X
150   LET B(X)=-1
160 NEXT N
170 LET B(25)=0
180 CODE  OF 1F 3F 3F 3F 3F 3F 1F OF
00 20 30 38 38 38 38 38 30 20 00
190 DPOKE GRAPHIC,LCTN(180)
200 LET C$=CHR$(128)+CHR$(129)
210 REM ** PRINT SCREEN **
220 PAPER BLUE
230 FOR N=1 TO 7
240   FOR X=1 TO 7
250     INK YELLOW
260     IF B(((N-1)*7)+X)=-1 THEN
GOTO 290
270     IF B(((N-1)*7)+X)=0 THEN
INK BLACK
280     PRINT  @ (X*10)+20,(N*20)+40
;C$;
290   NEXT X
300 NEXT N
310 INK CYAN
320 FOR N=1 TO 7
330   PRINT  @ 20,(N*20)+40;N; @
(N*10)+20,40;N;
340 NEXT N
350 INK YELLOW
360 REM ** MAIN GAME **
370 PROC CLS
380 PRINT  @ 3,225;"Please enter your
move ";
390 INPUT M$
400 IF LEN(M$)<>2 THEN  GOTO 360
410 LET J=ASC(LEFT$(M$,1))
K=ASC(RIGHT$(M$,1))
420 IF J<49 OR K<49 OR J>55 OR K>55
THEN  GOTO 360
430 LET x=J-48,y=K-48
440 LET C(1)=0,C(2)=0,C(3)=0,C(4)=0
450 PROC CHECK
460 IF Z=1 OR Z=-2 THEN  GOTO 520
470 IF Z=0 OR Z=-3 THEN  GOTO 500
480 PROC ALT
490 GOTO 460
500 PROC CHEAT
510 GOTO 360
520 PROC FLASH(x,y,0)
530 PROC FLASH(a,b,6)
540 PROC FLASH(f,g,0)
550 LET c=c-1,W=FALSE
560 FOR N=1 TO 7
570   FOR X=1 TO 6
580     IF B(((N-1)*7)+X)=1 AND
B(((N-1)*7)+X+1)=1 THEN  LET W=TRUE
590   NEXT X
600 NEXT N
610 IF W=TRUE THEN  GOTO 360
620 FOR N=1 TO 6
630   FOR X=1 TO 7
640     IF B(((N-1)*7)+X)=1 AND
B((N*7)+X)=1 THEN  LET W=TRUE
650   NEXT X
660 NEXT N
670 IF W=FALSE THEN  PROC END
680 GOTO 360
690 REM ** SYSTEMS PROCEDURES **
700 DEFPROC CHECK
710 IF y<3 OR C(1)=-1 THEN  GOTO 740
720 IF B(((x-1)*7)+y-2)=0 AND
B(((x-1)*7)+y-1)=1 THEN  LET C(1)=1
730 ELSE  LET C(1)=0
740 IF y>5 OR C(2)=-1 THEN  GOTO 770
750 IF B(((x-1)*7)+y+2)=0 AND
B(((x-1)*7)+y+1)=1 THEN  LET C(2)=1
760 ELSE  LET C(2)=0
770 IF x<3 OR C(3)=-1 THEN  GOTO 800
780 IF B(((x-1-2)*7)+y)=0 AND
B(((x-1-1)*7)+y)=1 THEN  LET C(3)=1
790 ELSE  LET C(3)=0
800 IF x>5 OR C(4)=-1 THEN  GOTO 830
810 IF B(((x-1+2)*7)+y)=0 AND
B(((x-1+1)*7)+y)=1 THEN  LET C(4)=1
820 ELSE  LET C(4)=0
830 LET Z=C(1)+C(2)+C(3)+C(4)
840 IF Z=1 OR Z=-2 AND
B(((x-1)*7)+y)=1 THEN  GOTO 860
850 ENDPROC
860 IF C(1)=1 THEN  LET a=x,
b=y-2,f=x,g=y-1
870 IF C(2)=1 THEN  LET a=x,
b=y+2,f=x,g=y+1
880 IF C(3)=1 THEN  LET a=x-2,
b=y,f=x-1,g=y
890 IF C(4)=1 THEN  LET a=x+2,
b=y,f=x+1,g=y
900 LET B(((x-1)*7)+y)=0,
B(((a-1)*7)+b)=1,B(((f-1)*7)+g)=0
910 ENDPROC
920 DEFPROC CLS
930 PRINT  @ 3,225;CHR$(30);CHR$(30)
940 VDU 1,6,2,1
950 ENDPROC
960 DEFPROC CHEAT
970 PROC CLS
980 PRINT  @ 3,225;"I dont allow
cheating round here !";
990 PAUSE 20000
1000 PROC CLS
1010 ENDPROC
1020 DEFPROC ALT
1030 LET C(1)=-1,C(2)=-1,C(3)=-1,
C(4)=-1
1040 PROC CLS
1050 PRINT  @ 3,225;"Which way ?";
CHR$(31);"L= left , R= right , U= up
, D= down";
1060 LET D$=GET$
1070 IF D$="L" OR D$="R" OR D$="U"
OR D$="D" THEN  GOTO 1090
1080 GOTO 1060
1090 IF D$="L" THEN  LET C(1)=1
1100 IF D$="R" THEN  LET C(2)=1
1110 IF D$="U" THEN  LET C(3)=1
1120 IF D$="D" THEN  LET C(4)=1
1130 PROC CHECK
1140 ENDPROC
1150 DEFPROC END
1160 PROC CLS
1170 PRINT  @ 3,225;"You were left
with ";c;" peg";
1180 IF c>1 THEN  PRINT "s"
1190 PRINT  @ 3,235;"Do you want
another go (Y or N)?";
1200 LET A$=GET$
1210 IF A$="Y" THEN  RUN
1220 PROC CLS
1230 PRINT  @ 3,225;"Be like that
then.";
1240 NEW
1250 ENDPROC
1260 DEFPROC FLASH(m,n,i)
1270 INK (i=0)*6
1280 FOR N=1 TO 3
1290   PRINT  @ (n*10)+20,(m*20)+40;
C$;
1300   PAUSE 1500
1310   PRINT  @ (n*10)+20,(m*20)+40;
"  ";
1320   PAUSE 1500
1330 NEXT N
1340 INK i
1350 PRINT  @ (n*10)+20,(m*20)+40;C$;
1360 ENDPROC
```

```
100 REM PATTERNS by C Cytera
110 PROTECT 0
120 RANDOM
130 LET C=1
140 DIM X(3),Z(3),V(3),C(3),W(3)
150 REPEAT
160   CLS
170   FOR J=0 TO 3
180     LET V(J)=RAND(4)*2
X(J)=RAND(256)
190     LET X(J)=X(J) MOD 256
Z(J)=X(J)+256
200   NEXT J
210   FOR G=1 TO 0 STEP -1
220     REPEAT
230       FOR J=0 TO 3
240         LET Z(J)=Z(J)+V(J)
250         LET W(J)=ABS(Z(J) MOD
512-256),W(J)=W(J)-(W(J)=256)
260       NEXT J
270       LET C=(C MOD 7+1)*6
280       INK C
290       MOVE W(0),W(1)
300       DRAW W(2),W(3)
310     UNTIL W(0)=X(0) AND W(1)=X(1)
AND W(2)=X(2) AND W(3)=X(3)
320   NEXT G
330 UNTIL FALSE
```

XOR PLOTTING By Chris Cytera

Have you ever wondered how ten-line programs for the BBC Microcomputer containing the statement GCOL 3,X manage to produce extraordinarily complex and beautiful patterns? The secret lies in something called exclusive-or plotting, so I will start by explaining what this is.

When the Lynx draws a line on the screen, anything underneath that line is 'pasted over' and forgotten; the new colour is not combined in any way with the colour that previously existed in its place on the screen. It is possible to produce a combination effect by the statement:-

```
PROTECT 7-INK
```

after every INK statement. This achieves what is known as or-plotting, because the current colour number is effectively logically ored with the colour underneath. However, this effect is not particularly spectacular because the OR operation produces no cancellation: if you attempt to use it to produce complicated patterns then you will end up with a screen containing rather a lot of white.

The exclusive-or operation when applied to binary digits produces a '1' if one of the two operands is a '1' but produces a '0' if they are both '1' or both '0'. So, for example, exclusive-oring 010 with 110 results in

```
      010
XOR   110
      ===
      100
```

If you are unsure about binary numbers, see NILUG NEWS Issue 2, page 5.

If this operation is applied to the Lynx's eight screen colours, some odd results occur:-

```
RED     XOR BLUE   gives MAGENTA
RED     XOR YELLOW gives GREEN
RED     XOR RED    gives BLACK
MAGENTA XOR CYAN   gives YELLOW
etc
```

You can check these if you wish by exclusive-oring the binary numbers of each of the colours together. One thing which you should find is that any colour exclusively-ored with itself gives black.

The obvious method of implementing exclusive-or plotting on the Lynx is to rewrite the routine which accesses the screen so that it first reads the screen, exclusive-ors what it finds with what is to be placed there ,and then writes the result to the screen.

Unfortunately, this does not work. The reason is that the Lynx's line drawing routine is not accurate enough; when it draws a line, it occasionally plots two dots in the same place. Normally this does not matter, but in exclusive-or mode, the second dot , being the same colour, cancels out the first. The result is a line with gaps in it. Therefore it is also necessary to rewrite the line drawing routine.

However, this opens up an opportunity for optimisation, as the normal line drawing routine is not terribly efficient. For example, consider a horizontal line of eight pixels long, with all the pixels on one screen byte. Normally, the screen is accessed eight times to draw such a line, which seems rather silly when it could be done in one screen access with a massive increase in speed. Also, the screen address and bit pattern are calculated from scratch for each pair of new screen co-ordinates, which is wasteful as they can more quickly be calculated from their previous values. Finally, the standard screen access routine is normally called, which wastes time fiddling around with the PAPER colour. The value of PAPER is irrelevant when drawing lines, so the new screen access routine ignores it.

The complete machine code program which accomplishes all the above is located at 9900H and occupies 019AH bytes of RAM. Relocation should be no problem provided that all lines referring to the labels 'pattern' and 'screen' are changed (31,42,4E,7E,8D,95,11D). Note that the code MUST be located on a page boundary, and line 82 should be changed so that the B register is loaded with the page number.

When the code has been typed in, save it on cassette. Then try it with:-

```
DPOKE &6263,&990A
POKE &9909,1
```

Try drawing a few lines. When things are working, save the code again.

You are now ready to try the demonstration programs CARPET and SQUARE DANCE. (To stop the latter, press the space bar. The pattern can then be continued with 'C' or a new one started with 'S').

POKE &9909,1 is used to set exclusive-or mode. To use the new line drawing routine in normal mode, type POKE &9909,0. Make this change to the above-mentioned programs and notice how inferior the results are.

To demonstrate the speed differences try the programs with the usual line drawing routine and the new one. Note the differences in speed, especially for horizontal and near-horizontal lines.

Many other beautiful patterns become possible when exclusive-or plotting is available. Experiment as much as you can - even simple programs can produce spectacular results.

CARPET

```
100 DPOKE &6263,&990A
110 POKE &9909,1
120 PROTECT BLACK
130 CLS
140 FOR Y=0 TO 255
150   INK Y
160   MOVE 0,Y
170   DRAW 255,255-Y
180 NEXT Y
190 FOR X=0 TO 255
200   INK X
210   MOVE X,0
220   DRAW 255-X,255
230 NEXT X
240 GOTO 240
```

SQUARE DANCE

```
100 DPOKE &6263,&990A
110 POKE &9909,1
120 PROTECT BLACK
130 VDU 2,BLACK,4
140 LET P=128,Q=124
150 LET X=0,Y=0
160 LET x=RAND(10)+1,y=RAND(10)+1
170 LET C=RAND(7)+1
180 REPEAT
190   REPEAT
200     INK C
210     MOVE P+X,Q+Y
220     DRAW P-X,Q+Y
230     DRAW P-X,Q-Y
240     DRAW P+X,Q-Y
250     DRAW P+X,Q+Y
260     LET X=X+x,Y=Y+y
270     IF ABS(X)>127 THEN  LET x=-x,X=X+x,C=RAND(7)+1
280     IF ABS(Y)>123 THEN  LET y=-y,Y=Y+y,C=RAND(7)+1
290   UNTIL NOTKEY$=""
300   REPEAT
310     LET K$=KEY$
320   UNTIL K$="S" OR K$="C"
330 UNTIL K$="S"
340 RUN
```

```
9000            ink EQU &625B
9000            protect EQU &626B
9000            write EQU &8B6
9000            inredblu EQU &69
9000            ingreen EQU &70
9900             ORG &9900
9900 B0402010    DEFB &80,&40,&20,&10
9904 08040201    DEFB 8,4,2,1
9909            dir DEFS 1
990A            xor DEFS 1
990A            draw
990A 2A6862      LD HL,(&6268) ;TO
990D E5          PUSH HL
990E ED5B6562    LD DE,(&6265) ;FROM
9912 2600        LD H,0
9914 54          LD D,H
9915 AF          XOR A
9916 ED52        SBC HL,DE
9918 17          RLA
9919 320899      LD (dir),A ;(dir)=0 if right
991C EB          EX DE,HL
991D 2A6462      LD HL,(&6264)
9920 2E80        LD L,&80
9922 D9          EXX
9923 2A6A62      LD HL,(&626A)
9926 E5          PUSH HL
9927 ED5B6762    LD DE,(&6267)
992B 2600        LD H,0
992D 54          LD D,H
992E B7          OR A
992F 012000      LD BC,32
9932 ED52        SBC HL,DE
9934 3003        JR NC,down
9936 01E0FF      LD BC,-32
9939            down
9939 EB          EX DE,HL
993A 2A6662      LD HL,(&6266)
993D 2E80        LD L,&80
993F 7A          LD A,D
9940 B3          OR E
9941 D9          EXX
9942 0601        LD B,1
9944 B3          OR E
9945 B2          OR D
9946 200B        JR NZ,line
9948 D9          EXX ;Switch to x
9949 7C          LD A,H
994A D9          EXX ;Back to y
994B CDD999      CALL pattern
994E 187C        JR end
9950            line
9950 05          DEC B
9951 CD5110      CALL &1051
9954 200B        JR NZ,11
9956 37          SCF
9957            12
9957 CB18        RR B
9959 CD4610      CALL &1046
995C CD5110      CALL &1051
995F 28F6        JR Z,12
9961            11
9961 D9          EXX ;Get y
9962 7C          LD A,H
9963 D9          EXX ;Back to x
9964 E5          PUSH HL
9965 CDD999      CALL pattern
9968 E1         POP HL
9969            nextdot
9969 7C          LD A,H
996A 19          ADD HL,DE
996B D9          EXX
996C 08          EX AF,AF'
996D 7C          LD A,H
996E 19          ADD HL,DE
996F BC          CP H
9970 282E        JR Z,novertdisp ;Same byte
9972 D9          EXX
9973 CDF899      CALL screen
9976 D9          EXX
9977 DD09        ADD IX,BC
9979 D9          EXX ;New pattern for new address
997A 08          EX AF,AF' ;Has x-coord changed?
997B BC          CP H
997C 3A0899      LD A,(dir)
997F 2011        JR NZ,hrdisp
9981             ;New pattern after vert but no horiz
9981 0F          RRCA
9982 79          LD A,C
9983 3804        JR C,leftmost
9985             ;Get rightmost bit
9985 CB21        SLA C
9987 1802        JR endbit
9989            leftmost
9989 CB39        SRL C
998B            endbit
998B B1          OR C
998C A9          XOR C
998D 4F          LD C,A
998E 10D9        DJNZ nextdot
9990 183A        JR end
9992            hrdisp
9992 0F          RRCA
9993 79          LD A,C
9994 3005        JR NC,rightvert
9996             ;Move left and vertically
9996 07          RLCA
9997 30F2        JR NC,endbit
9999 182B        JR lvdec
999B            rightvert
999B 0F          RRCA
999C 30ED        JR NC,endbit
999E 1812        JR rvinc
99A0            novertdisp ;Update bit pattern
99A0 D9          EXX
99A1 08          EX AF,AF'
99A2 BC          CP H ;x changed?
99A3 2825        JR Z,next ;No, do next coords
99A5 3A0899      LD A,(dir)
99A8 0F          RRCA
99A9 79          LD A,C
99AA 380E        JR C,left
99AC             ;Move right only
99AC 0F          RRCA
99AD 300E        JR NC,sameaddr
99AF             ;Increment address
99AF CDF899      CALL screen
99B2            rvinc
99B2 0E80        LD C,&80
99B4 DD23        INC IX
99B6 10B1        DJNZ nextdot
99B8 1812        JR end
99BA            left
99BA 07          RLCA
99BB 3806        JR C,decaddr;Next byte left
99BD            sameaddr
99BD B1          OR C
99BE 4F          LD C,A
99BF 10A8        DJNZ nextdot
99C1 1809        JR end
99C3            decaddr
99C3 CDF899      CALL screen
99C6            lvdec
99C6 0E01        LD C,1
99C8 DD2B        DEC IX
99CA            next
99CA 109D        DJNZ nextdot
99CC            end
99CC             ;Dump last pattern
99CC CDF899      CALL screen
99CF E1          POP HL
99D0 7D          LD A,L
99D1 326762      LD (&6267),A
99D4 E1          POP HL
99D5 226562      LD (&6265),HL
99D8 C9          RET
99D9            pattern
99D9 C5          PUSH BC
99DA 6F          LD L,A
99DB 7C          LD A,H
99DC 2600        LD H,0
99DE 29          ADD HL,HL
99DF 29          ADD HL,HL
99E0 29          ADD HL,HL
99E1 29          ADD HL,HL
99E2 29          ADD HL,HL
99E3 4F          LD C,A
99E4 0F          RRCA
99E5 0F          RRCA
99E6 0F          RRCA
99E7 E61F        AND &1F
99E9 85          ADD A,L
99EA 6F          LD L,A
99EB E5          PUSH HL
99EC DDE1        POP IX
99EE 79          LD A,C
99EF E607        AND 7
99F1 4F          LD C,A
99F2 0699        LD B,&99 ;B=High byte of ORG
99F4 0A          LD A,(BC)
99F5 C1          POP BC
99F6 4F          LD C,A
99F7 C9          RET
99F8            screen
99F8 08          EX AF,AF'
99F9 F5          PUSH AF
```

```
99FA C5          PUSH BC
99FB D5          PUSH DE
99FC E5          PUSH HL
99FD DDE5        PUSH IX
99FF E1          POP HL
9A00 51          LD D,C
9A01 7A          LD A,D
9A02 2F          CPL
9A03 5F          LD E,A
9A04 D5          PUSH DE
9A05 3A6B62      LD A,(protect)
9A08 0F          RRCA
9A09 3827        JR C,red
9A0B 3A5B62      LD A,(ink)
9A0E 0F          RRCA
9A0F 3802        JR C,bluewr
9A11 1600        LD D,0
9A13            bluewr
9A13 3A0999      LD A,(xor)
9A16 0F          RRCA
9A17 300C        JR NC,nobluexor
9A19 010080      LD BC,&8000
9A1C E5          PUSH HL
9A1D 09          ADD HL,BC
9A1E CD6900      CALL inredblu
9A21 7D          LD A,L
9A22 AA          XOR D
9A23 57          LD D,A
9A24 E1          POP HL
9A25            nobluexor
9A25 3EE8        LD A,&E8
9A27 08          EX AF,AF'
9A28 3E63        LD A,&63
9A2A 010080      LD BC,&8000
9A2D CDB608      CALL write
9A30 D1          POP DE
9A31 D5          PUSH DE
9A32            red
9A32 3A6B62      LD A,(protect)
9A35 CB4F        BIT 1,A
9A37 2026        JR NZ,green
9A39 3A5B62      LD A,(ink)
9A3C CB4F        BIT 1,A
9A3E 2002        JR NZ,redwr
9A40 1600        LD D,0
9A42            redwr
9A42 3A0999      LD A,(xor)
9A45 0F          RRCA
9A46 300C        JR NC,noredxor
9A48 E5          PUSH HL
9A49 0100C0      LD BC,&C000
9A4C 09          ADD HL,BC
9A4D CD6900      CALL inredblu
9A50 7D          LD A,L
9A51 AA          XOR D
9A52 57          LD D,A
9A53 E1          POP HL
9A54            noredxor
9A54 3EE8        LD A,&E8
9A56 08          EX AF,AF'
9A57 3E63        LD A,&63
9A59 0100C0      LD BC,&C000
9A5C CDB608      CALL write
9A5F            green
9A5F D1          POP DE
9A60 3A6B62      LD A,(protect)
9A63 CB57        BIT 2,A
9A65 2026        JR NZ,exit
9A67 3A5B62      LD A,(ink)
9A6A CB57        BIT 2,A
9A6C 2002        JR NZ,greenwr
9A6E 1600        LD D,0
9A70            greenwr
9A70 3A0999      LD A,(xor)
9A73 0F          RRCA
9A74 300C        JR NC,nogreenxor
9A76 E5          PUSH HL
9A77 0100C0      LD BC,&C000
9A7A 09          ADD HL,BC
9A7B CD7000      CALL ingreen
9A7E 7D          LD A,L
9A7F AA          XOR D
9A80 57          LD D,A
9A81 E1          POP HL
9A82            nogreenxor
9A82 3EE4        LD A,&E4
9A84 08          EX AF,AF'
9A85 3E65        LD A,&65
9A87 0100C0      LD BC,&C000
9A8A CDB608      CALL write
9A8D            exit
9A8D E1          POP HL
9A8E D1          POP DE
9A8F C1          POP BC
9A90 F1          POP AF
9A91 08          EX AF,AF'
9A92 C9          RET
9A93            init
9A93             ENT
9A93 210A99      LD HL,draw
9A96 226362      LD (&6263),HL
9A99 C9          RET
```

STOP by A.C. Karsten

```
10 PROTECT 0
100 PROTECT 0
110 PROC UITLEG
120 PROC INPUT NAMEN
130 LET A=2,B=2
140 PROC PRINT NAMEN
160 REPEAT
170   REPEAT
175     LET A=A+1
180     PROC MOVE(A,GREEN,165)
190     LET Z=KEYN
220   UNTIL Z>0
230   IF A=B THEN  LET S=S+1
240   PROC SCORE(20,S)
250   REPEAT
251     LET B=B+1
260     PROC MOVE(B,RED,205)
270     LET Z=KEYN
300   UNTIL Z>0
310   IF A=B THEN  LET s=s+1
320   PROC SCORE(40,s)
330 UNTIL s=10 OR S=10
340 PROTECT 0
350 VDU 1,1,2,7,24
360 IF S=10 THEN  PRINT  @ 3,20;A$;
370 IF s=10 THEN  PRINT  @ 3,40;B$;
380 VDU 25
390 FOR I=100 TO 1000 STEP 10
400   BEEP I,10,63
410 NEXT I
420 TEXT
430 INPUT "Another game Y/N";J$
440 IF J$="N" THEN  END
441 PRINT
450 INPUT "Same players Y/N ";J$
460 IF J$="Y" THEN  GOTO 130
470 ELSE  GOTO 120
480 DEFPROC MOVE(a,I,Y)
490 PROTECT BLUE
491 IF a=39 THEN  PROC EINDE
500 INK I
510 PRINT  @ a*3+3,Y;" ";
520 ENDPROC
530 DEFPROC SCORE(P,e)
540 PROTECT 0
550 INK BLUE
560 VDU 24
570 IF A=B THEN  PRINT  @ 90,P;e;
580 VDU 25
590 PAUSE 10000
600 ENDPROC
610 DEFPROC UITLEG
620 CLS
630 PRINT CHR$(24);"STOP THE BLOCKS";
CHR$(25);CHR$(31);CHR$(31);CHR$(31)
640 PRINT "You must try to stop your
moving block"
650 PRINT "ABOVE/UNDER your opponents
block"
660 PRINT "Your block has the same colour
as your  name"
670 PRINT "The one who does 10 stops
right wins"
671 PRINT "Both players can use any key"
680 PRINT CHR$(31);CHR$(31);CHR$(94);CHR
$(127);"  TON KARSTEN  19 JANUARY 1984"
690 ENDPROC
700 DEFPROC INPUT NAMEN
710 PRINT
720 INPUT "First players name ";A$
730 PRINT
740 INPUT "Second players name ";B$
750 ENDPROC
760 DEFPROC PRINT NAMEN
770 CLS
780 PROTECT 0
790 INK GREEN
800 VDU 24
810 PRINT  @ 3,20;A$;
820 INK RED
830 PRINT  @ 3,40;B$;
840 VDU 25
850 LET S=0,s=0
860 PROC MOVE(A,GREEN,165)
870 PROC MOVE(B,RED,205)
880 ENDPROC
890 DEFPROC EINDE
900 IF A=39 THEN  LET A=0,a=0
901 IF B=39 THEN  LET B=0,a=0
910 PRINT  @ 39*3+3,Y;" ";
920 ENDPROC
```

Machine Code For Beginners Part 4

It has ben pointed out to me that the the explanation of Assembler Language in part 3 was a little mis-leading. I stated that conceptually there are four fields. These are LABEL, OPCODE, OPERAND and COMMENT. The example was:-

```
        START LD HL,0 ;zero the HL register pair
```

I then stated that you would be able to find other examples in the rest of the magazine. This was slightly mis-leading because in the explanation I was talking about INPUT STATEMENTS to an ASSEMBLER whereas the examples published would be output from an assembler. Output has three extra columns. These are the assembly address, the generated machine code and source code line number. So you might type in:-

```
        LD C,£80 ;set C register
```

and the output might look like:-

```
        800C 0E80     0230        LD C,£80 ;set C register
```

In this example '800C' is the address where the machine code was assembled, '0E80' is the actual machine code which will load the C register with 80 and '0230' is the line number. The label field is blank, then comes the opcode and operand (LD C,£80) and finally there is the comment field.

Are assembler listings any good to you if you don't have an assembler? As long as you know which part is the machine code and where it has to be assembled you can enter it by using the Monitor M command. As an example consider the following program which sets the INK colour.

The input

```
 ORG  £8000
INK    EQU  £625B
WHITE  EQU  7
 LD   A,WHITE
 LD   (INK),A
 RET
```

The output

```
8000          0010        ORG  £8000
8000 625B     0020 INK    EQU  £625B
8000 0007     0030 WHITE  EQU  7
8000 3E07     0040        LD   A,WHITE
8002 325B62   0050        LD   (INK),A
8005 C9       0060        RET
```

The first thing to note is that lines 20 and 30 generate values in the machine code column (ie 625B and 0007). These values are however all generated at the origin (8000H) and hence don't count as machine code. The first opcode is in fact the '3E' from 'LD A,WHITE'. So if you wanted to enter this code using the Monitor M comand you would type the following:-

```
>MON [return]
*M8000 [return]
8000 (xx) 3E 07 32 5B 62 C9 [return]
```

Where 'xx' stands for the existing data at location 8000H. I hope that will help you to use the assembler listings published in NILUG NEWS. Now for some more theory.

RESTARTS

When I first started playing with micro-computers, some four years ago, machines where coming onto the market with small amounts of RAM, typically 4 or 8K. Some even had only 1K of RAM. A 16K machine was considered to be quite a large machine. Since memory was expensive and in short supply micro-processor designers gave their products instructions which would save memory. The Z80 is no exception. The RESTARTS are one byte call instructions (normal calls require 2 or 3). They allow the program designer to make calls to eight specified locations in memory. So if you were to put very commonly called routines at these places you could save quite a few bytes of memory.

RESTARTS

	Address	Opcode	assembler
Call Address	0000H	C7	RST 0
	0008H	CF	RST 8
	0010H	D7	RST 10
	0018H	DF	RST 18
	0020H	E7	RST 20
	0028H	EF	RST 28
	0030H	F7	RST 30
	0038H	FF	RST 38

Although you can't use these instructions for general purpose calls they are very useful. It is normal to use them for calling routines which are called very often. Lynx BASIC already uses the Restarts. Unfortunately only three of the restarts are of much use to the machine code programmer.

RST 8

RST 8 will output the ASCII character in the A register. For example 50H is the ASCII for the character P. If you load the A register with 50H then use RST 8 the P will appear on the screen.

```
>10 CODE 3E 50 CF C9
>CALL LCTN(10)[return]
P
>
```

RST 30

RST 30 jumps to the TRAP routine. This routine saves all the current registers, displays them and then is ready to accept monitor commands. This is how the monitor BREAK-POINT command works. When you set a breakpoint at a particular address the byte is saved and replaced by F7 (RST 30).

RST 38

RST 38 jumps to 6297H. At 6297H you will find a jump to print the message 'NOT YET IMPLEMENTED'. Since RST 38 jumps to a RAM vector it is possible to use this restart. If you had a particular routine which was called very often in your program it may pay you to reset the RAM vector and use RST 38 instead of a CALL instruction.

LOGICAL INSTRUCTIONS

There are three main logical instructions. These are AND, OR and XOR. Before I discuss these instructions I want to make a point about bad English. I recently saw a sign which said:-

**PRIVATE PROPERTY
NO PARKING AND
TURNING**

Now I KNOW what it means but I also know what it says. Its OK if I park there. Its OK if I turn there but they don't want me parking AND turning. I am sure that it should have said 'no parking or turning'.

Not so long ago it was my nephew's birthday. Since he is only 3 I gave him a bottle of pop and gave some money to my sister for him. Would he thank me? no. So my sister said 'either you thank Uncle Robert or you don't have the fizzy'. Although he eventually thanked me as I was leaving I wonder whether my sister would have carried out her threat - I doubt it.

The point is that we use English in a rather slack manner. It doesn't seem to matter to us humans. We seem to understand what is meant even if it is said incorrectly. On the other hand the Z80 is absolutely precise in its use of its English words. They mean one thing and one thing only.

AND

The AND instruction performs a bitwise comparison between the A register and the other operand(ie another register, RAM location or data). If both bits are 1 then the result is a 1. Otherwise the result is a 0. Consider A5H AND F0H.

```
         A5H is 1010 0101 in binary
     AND F0H    1111 0000
         ===    =========
         A0H    1010 0000
```

The AND instruction can be used to mask out selected bits. The BEEP command expects the volume in the range of 0 to 63. In actual fact you can give any volume you like. The reason is that the volume is ANDed with 3FH which happens to be 63 decimal. If you look at what happens when you AND 255 (maximum 8-bit number) with 3FH you will see why you can set any volume you like.

```
         FFH is 1111 1111 in binary
     AND 3FH    0011 1111
         ===    =========
         3FH    0011 1111
```

The 3FH acts as a mask and prevents a number larger than 63 being sent to the sound output circuitry.

OR

The OR instructions is similar to the AND instruction. It performs a bitwise comparison between the A register and the operand. If one of the bits is a 1 then the result is a 1. Otherwise the result is a 0. Consider A5H AND F0H.

```
         A5H is 1010 0101 in binary
      OR F0H    1111 0000
         ===    =========
         F5H    1111 0101
```

Unlike the AND instruction the OR instruction can be used to force the inclusion of set bits in the result. If you have an 8-bit value and you wish to ensure that a particular bit is set you simply OR the original value with a bit pattern that has the appropriate bit set. Suppose you want to ensure that bit 3 is set in a number which (for example) might be 77H. This is achieved by ORing 77H with 08H. In binary it looks like:-

```
        77H is 0111 0111 in binary
     OR 08H    0000 1000
        ===    =========
        7FH    0111 1111
```

As you can see bit 3 is now set and it will be set regardless of the original number.

All the logical instructions set the zero flag if the result is zero. Consequently you can use the OR instructions to test for zero. Consider a small program to print a string of text.

```
8000               ORG  £8000
8000 000D     CR   EQU  £0D
8000 0000     NULL EQU  0

8000 210980        LD   HL,TEXT-1
8003 23       LOOP INC  HL
8004 7E            LD   A,(HL)
8005 B7            OR   A ;Test for zero
8006 C8            RET  Z
8007 CF            RST  8
8008 18F9          JR   LOOP
800A 54484953 TEXT DEFM /THIS IS A MESSAGE/
     20495320
     41204D45
     53534147
     45
801B 0D00          DEFB CR,NULL
```

If you type this in at 8000H using the M command and then CALL &8000 [return] you will see the message printed on your screen.

The routine works as follows. Firstly the HL register pair is pointed to the byte before the start of the message. Then we enter a loop to display the characters. The HL register pair is incremented and the A register is loaded with what the HL register pair is pointing to. By oring A with itself the zero flag will be set if the contents of the A register are zero. If the zero flag is set the routine is ended. Otherwise characters to be printed are displayed by the RST 8 routine. The loop then continues until the '0' character at the end of the message (ie NULL) is found.

XOR

XOR stands for eXclusive OR and may be considered as an extension of the normal OR. In the normal OR instruction if either bit was a 1 then the result is a 1. Hence if both bits are 1 the result is 1. With the exclusive OR if both bits are 1 then the result is a 0.

```
         AAH is 10101010 in binary
     XOR F0H    11110000
         ---    --------
         5AH    01011010
```

There is one XOR instruction which is used very often. It is 'XOR A'. The A register is exclusively ored with itself. This has the effect of setting A to zero. You could use 'LD A,0' but this takes two bytes instead of the one for XOR A. Furthermore XOR A sets the zero flag whereas LD A,0 does not.

LOGICAL INSTRUCTIONS

	SOURCE								
	A	B	C	D	E	H	L	(HL)	data
AND	A7	A0	A1	A2	A3	A4	A5	A6	E6nn
OR	B7	B0	B1	B2	B3	B4	B5	B6	F6nn
XOR	AF	A8	A9	AA	AB	AC	AD	AE	EEnn

BIT MANIPULATION

As well as being able to manipulate bits indirectly with the logical instructions the Z80 provides instructions which permit direct manipulation and testing. These are SET, RESET and BIT. SET allows you to specify that a particular bit be set to a 1. RESET allows you to reset a bit (ie make it a zero). BIT allows you to test a bit. If the bit is zero then the zero flag will be set. Consequently you can use the BIT instruction to control the flow of a program.

The ESCAPE key is connected to bit 6 of port 0080H. If you set up a loop to test bit 6 of this port you can detect when the ESCAPE key has been pressed.

```
8000           0010        ORG  8000H
8000 018000    0020        LD   BC,0080H
8003 ED78      0030 LOOP   IN   A,(C) ;READ PORT 0080H
8005 CB77      0040        BIT  6,A ;TEST BIT 6
8007 20FA      0050        JR   NZ,LOOP ;ESCAPE?
8009 C9        0060        RET  ;RETURN IF IT HAS
```

You can enter the code via the M command or put it in a CODE line.

```
10 CODE 01 80 00 ED 78 CB 77 20 FA C9 [return]
CALL LCTN(10)[return]
```

Once you call the routine the Lynx will 'hang' until the ESCAPE key is pressed. Try changing the program to test for a different key. After that try testing for two keys to be pressed at the same time.

We have already met one way of ensuring that certain bits are not set when we talked about the BEEP command. Another way would be to use the RES instruction. Hence the first part of the BEEP command could have looked like the following:-

```
BEEP RES 7,A
     RES 6,A
        .
```

The SET instruction is used in the following program. Zero is loaded to location 9000H. Then bit 1 of the byte pointed to by the HL register pair is set. After you have run the program examine the contents of location 9000H (use PEEK in BASIC or the Monitor M command).

```
8000           0010        ORG  8000H
8000 210090    0020        LD   HL,9000H
8003 AF        0030        XOR  A
8004 77        0040        LD   (HL),A
8005 CBCE      0050        SET  1,(HL)
8007 C9        0060        RET
```

Eight-Bit Subtraction

Eight-bit subtraction takes place between the A register and another register,RAM location or data. The result always ends up in the A register. Binary subtraction is

BIT MANIPULATION AND TEST INSTRUCTIONS

BIT INSTRUCTIONS

bit	A	B	C	D	E	H	L	(HL)
0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46
1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E
2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56
3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E
4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66
5	CB 6F	CB 68	CB 69	CB 6F	CB 6B	CB 6C	CB 6D	CB 6E
6	CB 78	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76
7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E

SET INSTRUCTIONS

bit	A	B	C	D	E	H	L	(HL)
0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6
1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE
2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6
3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE
4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6
5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE
6	CB F8	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6
7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE

RESET INSTRUCTIONS

bit	A	B	C	D	E	H	L	(HL)
0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86
1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E
2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96
3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E
4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6
5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE
6	CB B8	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6
7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE

quite simple since you only have to deal with 0s,1s and 2s. Consider the example A5H subtract 73H. If you can remember that AH is ten (decimal) you should be able to see that the result is 32H (10-7 and 5-3). In binary the sum looks like this:-

```
        A5H is 1010 0101 in binary
    SUB 73H     0111 0011
        ===     =========
        32H     0011 0010
```

Starting at the right-hand side we have 1-1 which is 0. Then 0-1 which can't be done so you borrow TWO (remember its binary). Now we have 2-1=1. In the third column we have 1-0 but you must remember that we had to borrow 1 previously so the sum now becomes 1-1=0. Lastly for the nibble we have 0-0=0. The second nibble is done in a similar way

I can hear you say that I've cheated. What happens with examples like 22H - 77H? The answer is quite simple. You perform the subtraction exactly as before but when you get to the last bits (bits 7) the borrow that you need to make sets the carry flag to indicate that a borrow has taken place (ie in effect the result is negative)

```
        22H is 0010 0010 in binary
    SUB 77H     0111 0111
        ===     =========
        ABH     1010 1011  and the carry flag will be set.
```

It is easy to see the results of various subtractions by using a simple CODE routine.

```
        10 CODE AF 3E xx 06 yy 90 F7 [return]
        CALL LCTN(10) [return]
```

The xx value is loaded into the A register (opcode 3E) and the yy value is loaded into the B register (opcode 06). The opcode 90 subtracts the B register from the A register and puts the result in the A register. The F7 opcode is a restart which will call the Monitor and display the registers and flags. The registers are as follows:-

```
        AF HL DE BC
        AF' HL' DE' BC' IX  IY  SP  PC
```

Under the registers come the flags. There are eight bits to the F register (ie as normal) but only six bits are used. So far I have only told you about the carry and the zero flags. These are located as follows:-

```
        .Z.. ...C
```

If the flags are set then you will be given a C or a Z as appropriate. If the flag is not set a dot (.) is displayed.

SUBTRACT WITH CARRY

There is a second subtraction operation known as SUBTRACT WITH CARRY (SBC). This is very similar to the subtract instruction except that the carry flag is involved. The simplest way to consider this instruction is to imagine that the carry flag is subtracted from the result of a normal subtraction operation. Since the carry flag can either be a 1 or a zero there will only be a difference in the result if the carry flag is set. Lets try some examples. Try 55H - 42H with the carry flag set (ie 1).

```
                  55H is 0101 0101
              SBC 42H    0100 0010
                  ===    =========
                  13H    0001 0011
subtract carry    1              1
                  ===    =========
                  12H    0001 0010
```

As already noted if the carry flag were zero then it wouldn't affect the result.

COMPARE

It is convenient to put the COMPARE instruction in here. Its operation is the same as a normal subtraction except the A register remains unaltered ie a dummy subtraction is performed. The flags take the results of a normal subtraction. The compare instruction is used to test one 8-bit value against another. There can be three results

1. The argument is smaller than the A register.
   In this case the zero flag and the carry flag
   will not be set.
2. The argument is equal to the A register.
   In this case the zero flag will be set
   because the result of the dummy subtraction
   is zero.
3. The argument is larger than the A register.
   In this case the carry flag will be set
   because a carry was necessary to enable
   the subtraction to be performed.

You can try the various combinations with the following little program.

```
            10 CODE 3E xx 06 yy B8 F7[return]
            CALL LCTN(10)[return]
```

By loading different values into the A register (xx value) and the B register (yy value) you can examine the effects of the comparison on the zero and carry registers.

	SOURCE								
	A	B	C	D	E	H	L	(HL)	data
SUB	97	90	91	92	93	94	95	96	D6 nn
SBC	9F	98	99	9A	9B	9C	9D	9E	DE nn
CP	BF	B8	B9	BA	BB	BC	BD	BE	FE nn

Sixteen-Bit Subtraction

Sixteen-bit subtraction is done between the HL register pair and another register pair. The result ends up in the HL register pair. There is no normal subtraction only the subtract with carry operation. For this reason it is very common to see XOR A, OR A or AND A before the subtraction since these operations reset the carry flag.

	SOURCE		
	BC	DE	HL
Subtract with carry 'SBC'	ED 42	ED 52	ED 62

You can use the following program to experiment with the 16-bit subtraction opcodes. Load different values into HL and DE to examine the results. As with the 8-bit subtractions the zero and the carry flags will be set according to the result.

```
8000            0010        ORG  8000H
8000 210010     0020        LD   HL,1000H
8003 110003     0030        LD   DE,300H
8006 AF         0040        XOR  A
8007 ED52       0050        SBC  HL,DE
8009 F7         0060        RST  30H
```

LYNX 48/96K SOFTWARE

QUAZAR COMPUTING.	
Siege Attack	£ 5.95
Puzzle Pack	£ 5.95
The Worm	£ 5.95
Labyrinth	£ 4.75
Chancellor	£ 4.75
Reversals	£ 4.75
Space Trek	£ 4.75
Disassembler	£ 4.75
Midnight Blitz	£ 4.75
GEM SOFTWARE	
Monster Mine	£ 7.95
Golf	£ 7.95
Sultan's Maze	£ 7.95
Gempack IV	£ 7.95
Oh Mummy!	£ 7.95
Gamespack I	£ 7.95
Spanner Man	£ 7.95
LEVEL 9 COMPUTING.	
Colossal Adventure	£ 9.90
Adventure Quest	£ 9.90
Dungeon Adventure	£ 9.90
Snowball	£ 9.90
Compass (Compression Assembler)	£15.00
ABERSOFT.	
Mazeman	£ 4.95
LYNXMAN.	
Vorlon Invaders	£ 5.00

NILUG members 10% discount - quote mem. No.
QUAZAR COMPUTING DEPT. N.
29 WESTERN ROAD, NEWICK, EAST SUSSEX, BN8 4LE.
overseas - add 10%

LYNX MACHINE CODE PROGRAMS

"ROADER" **£5.95**

A roadrace game with twisting fast road, obstacles, fuel, time and distance display, real time clock, hiscore etc. Fuel consumption increases as you leave the track. How far can you travel and how fast before your fuel runs out or you crash? 100% Machine Code with fast graphics (yes fast) and sound.

"CODER" **£7.50**

This is the assembler we use to write our games. It is also a disassembler, machine code editor, and test tool. What other assembler works with CODE LINES, RAM or ROM, includes a FAST BLOCK PRINT routine to demonstrate BANK SWITCHING, has an integral Disassembler and costs only £7.50?

AND FROM ANDREW GOSLING "TOEDER" **£5.95**

A highly addictive version of a favourite arcade game. With 4 levels of play, 11 sheets to clear and hiscore. Deadly enemies such as mutant rubber ducks, submarines, crocodiles, blobmen and harpoons must be avoided to reach the gaps in the coral reef. Once 3 of your "TOES" have reached the gaps in the coral reef they may safely board the waiting boats which then sail off and a new sheet appears. Each sheet becomes progressively harder. 100% machine code with fast graphics (yes fast) and sound.

SEND CHEQUE OR P.O. TO:-

FL SOFTWARE,
13 St. Ronans Avenue,
Southsea,
Hants, PO4 0QE
Tel: (0705) 828295

☆ NEW! ☆ GAMES PACK FOR THE LYNX 48/96

Includes as follows :

- MATCHSTICKS : a stimulating variation of Nim.
- SQUASH : bat the ball against the walls.
- MASTERMIND : crack the computers secret code.
- LUNARLANDER : land your craft on safe terrain.
- GUIDED MISSILE : swallow the pills of the labyrinth before the missile hits you
- WORDGAMES : create wordsquares with your or the computer's vocabulary.

£5 cheque or P.O. to : M. G. Williams
134, Barrowgate Road,
Chiswick,
London W4 4QP.

DEALER ENQUIRIES WELCOMED.

ZEN

EDITOR • ASSEMBLER • DEBUGGER

A COMPLETE PACKAGE FOR WRITING & DEBUGGING Z80 MACHINE CODE PROGRAMS ON THE LYNX C/W MANUAL
LYNX CASSETTE (NILUG PRICE) £17.50
LYNX DOS £27.50
From LAURIE SHIELDS SOFTWARE
151, LONGEDGE LANE,
WINGERWORTH,
CHESTERFIELD,
S42 6PR.

STATE LYNX DOS OR LYNX CASSETTE
QUOTE YOUR MEMBERSHIP NUMBER

KEYBOARD AID

UNIQUE DATA STAND AND DATA CARDS

THE DATA STAND HAS BEEN DESIGNED TO HELP IN TYPING THE PROGRAMS FROM PUBLISHED MATERIAL AND IS MADE FROM ONE PIECE OF ACRYLIC PLASTIC. IT CAN BE USED EITHER OVER THE LYNX OR TO ONE SIDE WITHOUT TIPPING OVER WITH A MAGAZINE PLACED ON IT.

THERE ARE 3 DATA CARDS SUPPLIED WITH THE STAND AS FOLLOWS:-
CARD 1 PROVIDES THE USER WITH ALL THE DATA ON THE KEYBOARD, i.e. ALL ASCII CODES AND SINGLE KEY ENTRY COMMANDS.
CARD 2 SHOWS THE 'HIDDEN' KEYBOARD UNDER CONTROL 1 MODE, i.e. WHERE ALL THE LYNX GRAPHICS ARE AND HOW TO ACCESS USER DEFINED CHARACTERS DIRECTLY WHILST REMAINING IN THE ALPHA MODE. IT IS PRINTED IN RED FOR EASE OF USE.
CARD 3 IS OF GENERAL USE, FOR EASY CONVERSION TO BINARY, HEX. OR DECIMAL. ALL CARDS HAVE A BLANK KEYBOARD PRINTED ON THE REVERSE AND ARE LAMINATED FOR EASY CLEANING.

A FULLY DESCRIPTIVE LEAFLET IS AVAILABLE ON REQUEST, SEND A SAE.
DATA STAND + DATA CARDS £9.50 DATA CARDS ONLY £3.00

PERIPHERAL PRODUCTS
209 KENTON LANE, KENTON, MIDDX.
Telephone: 01-907 3406

VORLON INVADERS

The city of VORLON is experiencing an asteroid storm. The asteroids are burnt up in the atmosphere. Your enemy has decided to attack using missiles to destroy the atmosphere and guided mines to destroy your defence satellite. When the atmosphere is destroyed the asteroids will destroy Vorlon. A fast action game exploiting the versatile graphics of the LYNX. With numerous fast-moving aliens, missiles and asteroids on the screen it requires tactics as well as good coordination. How well can YOU defend Vorlon?

VORLON INVADERS by LYNXMAN Price £5.00
(10% Discount for NILUG members)
270, Tithe Pit Shaw Lane,
Warlingham, Surrey, CR2 9AQ.