NILUG

NATIONAL INDEPENDENT LYNX USER GROUP

NEWS

Magazine for **LYNX** Users

Volume 1. Issue 4.

NILUG NEWS

VOLUME 1. ISSUE 4.

CONTENTS

ANNUAL SUBSCRIPTIONS
UK £9.00 per year
OVERSEAS £12.00 per year
(six issues)

BACK ISSUES
Copies of all previous issues are still available for £1.50 each.

Send cheques/P.O. payable to NILUG, to:-
53,Kingswood Avenue
Sanderstead
South Croydon
CR2 9DQ.

EDITORIAL

This Issue has been a litle disappointing for me. I had hoped that by now the magazine would be bigger. As you can see it isn't. There are two factors which influence magazine size. The first is membership and the second is copy. Neither are in particularly good supply. NILUG needs more members so if you are lending your copy of NILUG NEWS to a friend do yourself a favour and tell him to buy his own. As far as copy is concerned if you have any reviews, articles or programs to offer I would be very grateful. NILUG pays about £10.00 per printed page for published material.

Two members have asked me to publish their details so that local users can contact them. They are:-

Mr M.Fenton, Fleur-De-Lis, Hatchet Lane, Winkfield Row, Nr Windsor BERKS., SL4 2ES.

Peter Colingridge, Higher Ledges, Waterrow, Wivelscombe Somerset TA4 2BA.

Peter has asked me to point out that this is his home address and that letters will have to forwarded to him when he is away studying.

There was an error in the letter from Robert White published in Issue 3. Line 55 should have read:-

```
55 FOR J = 0 TO 80 STEP 10
```

Please accept my apologies for my mistake.

Robert Poate,
EDITOR

LETTERS

FREE OFFER to NILUG Members.

If there are people out there who have got stuck on their Lynxes or their computing in some way, I am willing to try to answer queries. My experience in machine code is limited but otherwise I should be able to offer some advice on most difficulties

Obviously if I get a lot of letters I will have to start asking for SAEs, but for now by all means just write. If any of my answers are brief or in note form it will be due to constraints on my time, not poor consideration of your questions.

Lastly please note I am a Lynx owner and NILUG member but otherwise I am not connected with either Camputers or the Editor of NILUG NEWS.

Chris Mathews,179 Goldhurst Terrace,London,NW6 3ER.

Dear Robert

The GOMOKU prog. was very good, even better with a bit of extra sound added.

```
ADD    50   PROC BE
       355  BEEP 70,5,50
       407  BEEP 50,8,50
       425  BEEP RAND(150)+50,3,63
       465  BEEP RAND(100)+20,3,63
AMEND  4013.5 BEEP (RAND(100)+50)+10*i,5,50-2*i
       5041 PROC BE
ADD    5042 PAUSE 50000
       6000 DEFPROC BE
       6010 FOR J=1 TO 29
       6020   BEEP 100-2*J,30-J,40+J
       6030   BEEP 10+2*J,5+J*2,60-J
       6040 NEXT J
       6070 ENDPROC
       7000 DEFPROC TE
       7010 BEEP RAND(200)+100,5,50
       7020 BEEP RAND (75)+20,10,60
       7050 ENDPROC
ADD    175  PROC TE
       195  PROC TE
       215  PROC TE
       235  PROC TE
       1050 PROC TE
       1575 PROC TE
       1635 PROC TE
       5065 PROC TE
```

I was interested to read in the LYNX magazine that someone else has discovered the SOUND command. Here are some of the routines that I have been using:-

```
100 REM U.S. POLICE SIREN ???
110 FOR J=1 TO 50
120   SOUND 220,J
130   SOUND 110,J
140   SOUND 55,J
150 NEXT J
160 GOTO 110
170 REM E.T. MUSIC
180 LET X=0
190 FOR J=1 TO 50
200   SOUND 220+X,J
210   SOUND 110+X,J
220   SOUND 55,J
230   SOUND 27,J
240 NEXT J
250 LET X=X+20
260 GOTO 190
270 REM UPWARD SPIRALS (ABOUT 30 SECS)
280 FOR J=1 TO 40
290   FOR N=1 TO 50
300     SOUND 50+J*10,60-N
310     SOUND 25+J*10,60-N
320   NEXT N
330 NEXT J
```

Try RUN 100; RUN 180 & RUN 280.

I have spent the last few months writing a 'Guitar' program and one of the most difficult parts proved to be trying to find out what the correct wavelength should be for the BEEP command. After trying all the previously published scales I found that over two octaves none of them were correct. Eventually I borrwed an electronic organ and tuned the LYNX by ear. These are the results and whats more they sound correct!

NOTE	WAVELENGTH	FREQUENCY (Hz)
E	670	164.8
F	627	174.6
F*	593	185
G	562	196
G*	530	207.7
A	499	220
A*	467	233.1
B	443	246.9
MID C	419	261.6
C*	397	277.2
D	374	293.7
D*	352	311.2
E	333	329.7
F	311	349.2
F*	295	370
G	278	392
G*	262	415.3
A	248	440
A*	232	466.2
B	220	493.9
C	206	523.3
C*	195	554.4
D	185	587.4
D*	173	622.3
E	163	659.3
F	153	698.4
F*	145	740
G	135	784
G*	128	830.7
A	120	880
A*	112	932.3

[Editor's note: The sharp notes have been flagged with an asterisk because I don't have a hash symbol on my printer.]

The frequencies shown are for the "EQUAL TEMPERED SCALE". Every interval has the same frequency ratio of 1.05946 which is the twelfth root of two.

Yours sincerely, Michael Lawson.

REVIEWS

REVERSALS from QUAZAR Computing By N Jenkins

This is an excellent version of the popular board game Othello. The program itself is written partly in machine code and partly in BASIC, so response times are very fast. Full instructions and a demonstration game are included for the novice, while the program features a number of attractions for the more advanced player. These include 3 levels of play, an option to set the pieces up to and position, to let the computer start first or to have a replay of the last game. Level one provides an easy game, level two is a competitive game and level three is almost unbeatable. The graphics are good and the sound effects reasonable. Value for money 8 out of 10.

SPACE TREK from QUAZAR Computing By R.B.Poate

I first saw this program at the PCW show in September. It is a Lynx version of an old favourite in which you hunt down the Klingon baddies and save the Universe. Unlike other versions of this game I have played you are given a display which resembles what the ship's Captain would see ie a 'screen' on which all the action takes place.

The game is neatly packaged and comes with an instruction sheet and a small keyboard layout sheet which sits above the numeric keys to remind you what they do - a good idea.

The tape auto loads and plays the Star Trek theme - unfortunately I wasn't a Star Trek fan so I didn't recognize it. You then choose from three levels of play. They are described as Cadet, Captain and Admiral. I think of them as playable, competitive and suicidal. I haven't yet beaten the Klingons as an Admiral.

When you find the Klingons you try to manoeuvre your sights in the middle of the 'screen' onto them. You can then blast them with phasers or torpedoes. Unfortunately (for you) the Klingons aren't too happy about this and they take avoidance action. The higher the rank you carry the more they move.

Overall, a good implementation of this popular game and at £4.75 (£4.25 for NILUG members) good value for money.

COLOSSAL ADVENTURE from LEVEL 9 Price £9.95 By Lisa Israel

Adventure games from software house Level 9 Computing have won nothing but praise in the magazine reviews. After six weeks of continious amusement and bafflement with Colossal Adventure I now know why.

Adventure games are different to the high-speed arcade games found in pubs ans amusement arcades. Instead of sending down waves of hostile aliens, adventures describe in words complicated landscapes which contain hords of well hidden treasure.

It is your job to get at the treasure. To find them, you will need hours of concentration to draw maps of the terrain, search out all the bizarre tools and weapons you need and avoid the dwarves, pirates and other vicious obstacles that will block your goal.

Colossal Adventure puts the very first adventure game ever - from the days when computers were always as big as a room - onto the Lynx. It holds literally hundreds of locations to be explored - an amazing number for a micro. After weeks of searching I still haven't found them all. Only last week, I climbed up a beanstalk to discover further underground vistas where trolls and giants lurked alongside emeralds as big as plover's eggs.

Loading the game takes about eight minutes, which is a bit annoying, and you have to wait a few seconds while the Lynx prints out lengthy room descriptions. But the game is so good that the wait is well worth it. If you like long, intricate puzzles and are prepared to be frustrated time and time again as your lamp batteries run out and you fall into and underground hole in the dark - then I have only one piece of advice for you: go out and but Colossal Adventure or one of Level 9's other games for the Lynx.

Lynx-users everywhere will thank Level 9 for making a rare move to put its games on the Lynx. They are the best micro-computer adventure games around.

KEYBOARD AID from PERIPHERAL PRODUCTS By R.B.Poate

Peripheral Products have launched a keyboard aid for the LYNX. The total package consists of a stand and a set of three cards giving information about which keys do what.

The stand is made of grey acrylic and it can be fitted over the LYNX or used separately. It can support the cards or a magazine without tipping over.

The cards are quite large at about 350mm by 180mm and are covered in plastic. This means that you can write on them and then wash them clean for re-use. The first card has a keyboard layout on the front on which the standard keys are documented. For each key the following information is displayed:-

1. The normal letter/symbol.
2. The shifted letter/symbol.
3. The single key entry.
4. The ASCII code for 1 above.
5. The ASCII code for 2 above.

On the back of the card is a blank keyboard layout which allows you to document your own user defined keys.

The second card is printed in red and has the information for the CONTROL/1 mode of operation. There is another blank layout on the back.

The third card has a blank layout on the front and some information regarding the conversion of numbers from one base to another (ie hex to decimal).

Overall I view the cards as useful but I would have prefered to see them A4 compatible. They would have only needed to be reduced by about 15% to do this.

The cards cost £4.95 and the stand+cards £14.95. I understand that the stand is available separately for £11.95.

ASSEMBLER REVIEWS By R.B.Poate

To my knowledge there are now four 'assembler' products available for the Lynx. There are two 'instant' assemblers (CODER and MODER-80) and two full assemblers (ZEN and COMPASS). In order not to confuse you I will explain the difference between them before reviewing the products.

The purpose of an assembler is to make the creation of machine code programs as easy as possible. It does this in several ways although most notable is the use of mnemonics which are easy to remember and understand. Instead of entering hex codes like '76' you enter 'HALT' and the assembler will translate the 'HALT' into the code '76'. The increase in legibility is obvious. Even if you are not familiar with Z80 mnemonics I'm sure you can guess what HALT does to the Z80 processor. Assemblers also work out jump and call addresses and provide pseudo opcodes to make life even easier. They have a file of source code which the user enters (eg HALT etc). When the user is ready he instructs the assembler to convert the source code into machine code. Hence it is a two stage process.

The 'instant' assemblers are different in that they assemble the instructions as they are typed in. This means that they can't work out jump or call addresses so the user has to supply these in one way or another.

COMPASS

COMPASS comes very neatly packaged complete with a 36 page manual and a command summary card. Two versions are supplied. The first loads into user RAM in the usual way. The second version loads initially into user RAM but it may then be loaded into the alternate green video RAM. This frees almost all of the 8K used by the first version enabling larger programs to be written for a given amount of RAM.

In addition to the memory saving available by using the alternate green video RAM the source code is compressed or tokenised like BASIC which not only saves space but speeds up the assembly process.

The source code is entered using the editor which is started using 'I' for insert. I didn't like the editor very much. To begin with every time you enter a new line or delete an existing line the whole source file is renumbered. On the face of it this is not too much of a problem but when you print the source file and then make changes (insertions or deletions) the listing is immediately incompatible with the source file. This means that you can't go directly to a given line and modify it. In addition to this the editing of lines is a two step process. First of all you have to use the P (position) command to display the line you want to edit. Then you have to use CONTROL/Q to place the line in the edit buffer. It would be better if there were an E (edit) command which would perform both operations in one go. It would be even better if you could enter lines of code from command mode simply by prefixing them with a line number as you can in BASIC.

COMPASS supports the full set of standard Zilog opcode mnemonics as well as ten pseudo opcodes which include conditional assembly. It only supports two operands (+ and -) and will accept numbers in hex or decimal. It has 26 error messages which I am sure a beginner will find most helpful.

The tape handling on COMPASS is rather poor. Tapes are saved using the W command but this simply puts you into the Monitor. You then have to press CONTROL/Q to pick up a command line which will dump the source code to tape. If you want to give the file a filename this must be edited in. To return to COMPASS you then have to remember 'G 9F03 [RETURN]'. Loading of source files is also done with the monitor.

The manual, on the other hand, is a very professional effort with cardboard covers and wire bound. The printing is clear and well laid out. There are one or two printing errors. In particular the memory map given shows the red and blue banks reversed. The manual also omits to explain the pseudo ops ORG and ENT.

ZEN

ZEN is supplied on tape with a 24 page manual printed on fluorescent yellow paper. I have found this to be useful since it means that I can easily find it amongst all the papers on my desk. I had trouble loading ZEN at first. I finally used a phase-shift lead and found that I could then load it every time. ZEN loads from 6C00H to about 8400H, then auto runs and displays its logo and copyright notice. Pressing any key puts it into command mode.

ZEN is described as an 'EDITOR + ASSEMBLER + DEBUGGER + MONITOR' and consequently has an extensive set of commands. It has a line editor which is entered via the E command. Lines of source code may then be entered. As with the COMPASS editor the lines are automatically renumberd each time and the comments I made about COMPASS' editor apply equally well.

ZEN supports the full set of ZILOG mnemonics plus eleven pseudo opcodes. It will accept numeric data in decimal, binary, octal or hex. ZEN accepts six operands (+,-,*,/,AND and OR) which allow the user to define quite flexible expressions. ZEN's DEBUGGER/MONITOR is a duplication of some of the LYNX's monitor and BASIC. It has 5 Debugger commands and 10 Monitor commands. The Debugger allows you to jump to your program, return to BASIC and update the registers. The Monitor commands allow you to perform such tasks as modify memory, change ink and paper, perform a byte search etc.

Tape handling is good with LOAD,VERIFY and SAVE commands all available for both source files and object files. The only improvement would be the inclusion of an APPEND command for source files. It is also possible to assemble code straight to a tape in MLOAD format. This makes it easy to develop programs which will sit where ZEN normally is. There are only ten error messages and these are rather blunt. Newcomers might find them a little unhelpful. One that I did like was "Huh?" which I like better than the usual "Syntax error".

Comparison ZEN v COMPASS

You could trade insults (or compliments) between ZEN and COMPASS for some time. ZEN supports binary and octal numbers. COMPASS has conditional assembly. ZEN has some debugging commands and COMPASS uses text compression, etc, etc etc. I don't actually use either of these assemblers and so before I reviewed them I analysed what features of my assembler I actually use. It was surprisingly few. Consequently I view the 'frills' which these assemblers offer to be of secondary importance. Take binary numbers and conditional assembly for example. I can think of times when they would have been useful but I have managed without.

The real criteria are cost and ease of use. In terms of cost COMPASS wins by a short head [COMPASS £15.00; ZEN £17.50 (special NILUG price)]. In terms of ease of use then ZEN has the advantage. Having used many systems on all sorts of machines I am well aware of how easy it is to confuse the commands of one system with those of another. COMPASS requires you to use the Lynx monitor (and perhaps BASIC) whereas ZEN is a complete package. Although some of ZEN is a duplication I would find it easier to use. It has a consistent set of commands and once learnt you need never stray from ZEN.

If you have a 96K Lynx then the space advantage offered by COMPASS will be of little significance. You will still have more than 20K available for source code. If, however, you have a 48K Lynx and you don't intend upgrading to 96K then the banked switched version of COMPASS may well be a better buy.

CODER

CODER is an 'instant' assembler/disassembler which is written in BASIC with a substantial amount of machine code in CODE lines. It will allow you to work on any area of memory including CODE lines themselves.

Although CODER only supports about 300 of the standard Z80 mnemonics this does not restrict the user very much since any which are missing are seldom used and may be entered directly by the user if required.

CODER supports 19 single letter commands to allow manipulation of the code. Once the area of memory on which you wish to work has been entered, CODER disassembles the first byte(s) and displays it. You may now enter new mnemonics and CODER will instantly assemble them and show the opcode(s) and the mnemonic.

One interesting feature of CODER is that it provides a call up table which may be used to make calls from one CODE line to another and so make machine code programs relocatable.

An eleven page manual is provided which starts with a rather amusing HEALTH WARNING. 'COMPUTER ADDICTION AND IN PARTICULAR MACHINE CODE OBSESSION CAN CAUSE LOSS OF SLEEP, EYESIGHT DAMAGE AND LOSS OF SENSE OF PRIORITIES LEADING TO NEGLECT OF SELF, LOVED ONES, RESPONSIBILITIES AND POSSIBLE DIRE CONSEQUENCES'. Its a pity all assembler manuals don't have such a warning - because its all to easy to get hooked! The health warning is unlike the rest of the manual which is terse and beginners might find it a little difficult to pick up.

MODER-80

MODER-80 is described as a 'Non-symbolic assembler/ disassembler'. It is written in machine code and I had no trouble in loading it. The first action the user must do is to define a workspace. This is where the instructions will be assembled to. Unfortunately it will not allow you to work on BASIC CODE lines.

The 'manual' is written on the cassette tape inlay which unfolds to form about 7 'pages' of explanation. Although small I found it well written and quite adequate. It describes the six commands available which are as follows.

1. F. Fill the workspace with the breakpoint byte F7H
2. J. Jump to monitor.
3. M. This command allows you to modify bytes in your workspace. It will allow insertion and deletion by means of CONTROL/I and CONTROL/D. Although I found this somewhat confusing I dare say you would get used to it.
4. P. This command prints the workspace in mnemonic form.
5. S. S allows you to save the disassembled code onto tape. It claims that you can then reload the tape into a symbolic assembler but doesn't describe how to do this.
6. W. This command is used to define the workspace.

Overall I found MODER-80 somewhat disappointing. The editor (M command) does not use the standard Lynx method (ie insert mode) which I found odd and secondy it doesn't like trailing spaces on the end of input lines. This caused numberous BEEPS to be sounded along with the message 'SYNTAX ERROR'. I didn't like the P command. It insists on two arguments every time. It would be much better if it defaulted to print from the beginning if no arguments were given and print from the first arguement if only one were given. Finally, I felt that it should provide a clear screen command and the BEEP you get with errors could drive me mad.

Comparison CODER v MODER-80

I was rather disapointed with MODER-80. I was expecting a version of CODER written in machine code. In spite of MODER-80 covering all the Z80 mnemonices I think that I would soon get frustrated with it. I couldn't,for example, imagine myself assembling large machine code programs using it. Since it doesn't allow you to create CODE line directly I find myself wondering who will use it.

I view CODER as an open ended product. Since you can find out how it works it would be possible to extend it (not that it needs it). If you want to write large machine code programs I couldn't advise you to buy CODER. But if you want to simply add a few machine code routines to BASIC programs by using CODE lines then I think CODER is quite a good way of doing it.

LYNX BUGS by Colin I. Clayman.

1. Zero is not zero in Version 1.

Whilst 0-0 may be printed ad zero, when compared with 0 in a logical expression it turns out to be unequal.

```
eg IF 0-0<>0 THEN PRINT "UNEQUAL!"
```

does indeed print UNEQUAL!

This is most noticeable in games where the arrow keys are used to move an object between two limits:-

```
X=X+(KETN=1 AND X<>240)-(KEYN=22 AND X<>0)
```

When X=0 and no key is pressed X becomes 0+0-0 which is not zero, so that if "left arrow" is pressed next , the object falls off the end of the screen.

Avoidance. Rewrite the expression so that the minus part comes first. The above example could be written as:-

```
X=-(KEYN=22 AND x<>0)+(KEYN=12 AND X<>240) + X
```

2. INACCURATE ARITHMETIC

There are many examples of unacceptable arithmetic accuracy. One of the most striking is :-

1.000001**1E6 gives 3.89 instead of 2.71828 (E). So a teacher wishing to demonstrate that :-

(1+1/n)**n tends to e as n tends to infinity would be foiled by the Lynx. I believe these errors derive from small but unnecessary errors in LN and LOG ie:-

LN(1)= 3.68E-7 instead of exact 0.

LOG(1)=1.6E-7 instead of exact 0.

and these small errors magnify in subsequent operations.

3. String functions of string functions

Some string functions of string functions are inexplicably rejected or require unnecessary brackets eg:-

```
ASC(RIGHT$(A$,1))-ASC("0")
```

is rejected as a "Syntax error" and must be re-written with extra brackets, to be accepted:-

```
(ASC(RIGHT$(A$,1)))-ASC("0")
```

Also , inexplicably, the string:-

```
"A"+CHR$(ASC("0")+1)+"B"+CHR$((ASC("0"+2)
```

gives 0 instead of A1B2 but is correct if ASC("0") is replaced by its ASCII code, 48.

4. RENUM ignores ELSE IF.

RENUM does not renumber line numbers in ELSE IF ...... THEN GOTO line.

(Neither does it renumber LCTN(line no) - ED.)

5. Conditions that corrupt BASIC

Any of the following conditions will corrupt the BASIC program and possibly cause the micro to go dead:-

a. IF containing "z"

b. REM containing <CONTROL> 1 graphic characters.

c. PROC with more parameters than defined in DEFPROC.

d. If containing many string functions of string functions, eg IF LEFT$(A$,VAL(N$))="X" THEN ....

It can be very annoying if you have been typing for hours and then lose your program on LISTing or EDITing it, just because of one of these apparently simple conditions.

Driving a Serial Printer

RS232 is a very common method of driving printers, modems etc and many micros have some form of serial interface. The Lynx is no exception. Using the Lynx's serial interface is not particularly straightforward.

Chararacters are stored in the Lynx as 8-bit ASCII codes. In order for serial communication to take place these eight bits must be sent one at a time along a wire to the receiving device. Various control signals are required and since parallel to serial (and serial to parallel) communication is very common special chips have been produced to handle the operation. They are called Universal Asynchronous Receiver/ Transmitter or UARTs to you and me. The Lynx has such a chip.

Characters are sent to the UART one at a time which stores them before transmitting. In order to inform the receiving device that a character is about to be sent a START BIT is sent first. Once the receiving device has detected the start bit it 'clocks' the following data bits in. Finally the transmitting device sends a STOP BIT. This informs the receiving device that the character has been sent.

Now if there is 1 start bit, 8 data bits and 1 stop bit then it takes 10 bits to send one character. The Lynx's UART is clocked to send 2400 bits per second or in other words 240 characters per second. Now if the receiving device can receive characters at this rate there is no problem. You simply send the data.

Things aren't that simple when it comes to driving a printer. Most printers are slower than 240 characters per second. As a result if you send characters to a printer at a rate of 240 characters per second either you will lose some of the characters or damage the printer. Obviously a control signal is necessary and it is usually provided by a signal from the receiving device know as Data Terminal Ready or DTR for short. Polarity of this signal varies but basically it flags a 'ready' or 'busy' condition.

There seems to be two types of printers on the market. The first only sets the DTR signal to busy when the printer's buffer is full. The second type continually alternates the DTR signal between 'busy' and 'ready'. If you have the first type of printer then all that is necessary is to slow down the rate at which the characters are sent. This is what Camputers' S.PRINT software does and it explains why it will not work with all printers.

If you have the second type of printer then you will have to provide the DTR signal. Since my Epson printer is of this latter type I have spent a fair amount of time trying to provide this signal.

METHOD 1

The first method I tried was to use the A/D input as DTR input signal. The method works but rather slowly. Although it is not a particularly satisfactory method it is easiest since it doesn't interfere with the Lynx and requires a minimum of components.

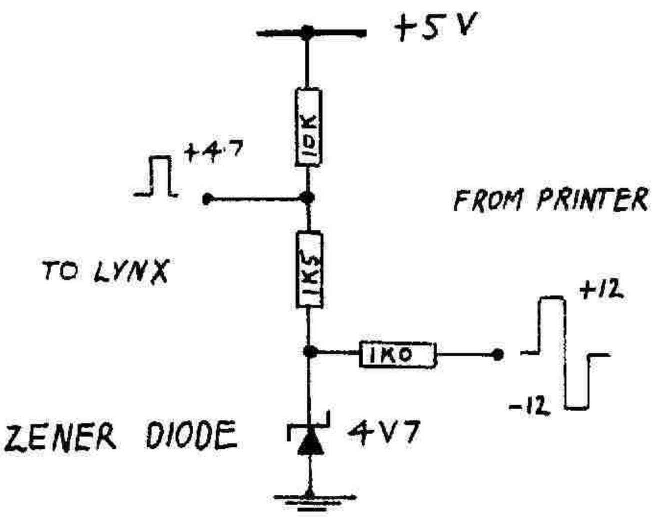
The A/D input port appears to have a large amount of hysteresis. This means that the input signal which the Lynx 'sees' lags behind the input signal. When reading programs from tape this doesn't matter but for a handshake signal line it is critical. Normally it would be enough just to test the line to see if it is 'ready' and then send the data. Owing to the hysteresis it is necessary to wait until the handshake signal changes from 'ready' to 'busy' and then to 'ready' again before any data is sent.

The assembler routine below has been assembled at 8000H but is in fact fully relocatable. It needs to be initialised by DPOKE &6202,&8000. Then LLIST, LPRINT and LINK will all work. The TABLE of values given at the end are for my Epson printer. Your printer may require different values. For example mine has been set up to take the line feed character (0AH) to be 'line feed, carriage return and print the buffer'. You may find that your printer will require a carriage return (0DH) instead.

```
8000          0010        ORG  8000H
              0020 ;SERIAL PRINT DRIVER ROUTINE
              0030 ;USING THE A/D INPUT PORT AS
              0040 ;A DTR HANDSHAKE SIGNAL
              0050 ;BY R.B.POATE
              0060 CR     EQU  0DH
              0070 LF     EQU  0AH
8000 FE80     0080 PRINT  CP   80H
8002 D0       0090        RET  NC ;DON'T PRINT GRAPHICS
8003 FE20     0100        CP   £20
8005 300E     0110        JR   NC,PRT1
              0120 ;A CONTROL CHARACTER
              0130 ;PICK UP CHARACTER FROM THE TABLE
8007 2A0262   0140        LD   HL,(£6202)
800A 014300   0150        LD   BC,TABLE-PRINT
800D 09       0160        ADD  HL,BC
800E 0600     0170        LD   B,0
8010 4F       0180        LD   C,A
8011 09       0190        ADD  HL,BC
8012 7E       0200        LD   A,(HL)
8013 B7       0210        OR   A
8014 C8       0220        RET  Z ;RETURN IF ZERO
8015 F5       0230 PRT1   PUSH AF  ;PROTECT CHARACTER
              0240 ;IS THE UART CLEAR?
8016 DB84     0250 TUART  IN   A,(£84)
8018 CB77     0260        BIT  6,A
801A 28FA     0270        JR   Z,TUART ;LOOP UNTIL CLEAR
              0280 ;SET UP A/D INPUT
801C 3E0C     0290        LD   A,12
801E D386     0300        OUT  (£86),A
8020 3E20     0310        LD   A,£20
8022 D387     0320        OUT  (£87),A
8024 3E3C     0330        LD   A,60 ;TEST VALUE
8026 D384     0340        OUT  (£84),A
8028 01800B   0350        LD   BC,£0B80
802B ED78     0360 CTL1   IN   A,(C) ;LOOP WHILE 'BUSY'
802D 1F       0370        RRA
802E 30FB     0380        JR   NC,CTL1
8030 ED78     0390 CTL2   IN   A,(C) ;LOOP WHILE 'READY'
8032 1F       0400        RRA
8033 38FB     0410        JR   C,CTL2
8035 ED78     0420 CTL3   IN   A,(C) ;LOOP WHILE 'BUSY'
8037 1F       0430        RRA
8038 30FB     0440        JR   NC,CTL3
              0450 ;NOW ITS 'READY' SEND THE CHARACTER
803A F1       0460        POP  AF
803B D382     0470        OUT  (£82),A ;SEND THE CHARACTER
803D AF       0480        XOR  A
803E D384     0485        OUT  (£84),A ;ZERO SPEAKER
8040 D387     0490        OUT  (£87),A ;DISABLE A/D
8042 C9       0500        RET
8043 00       0510 TABLE  DEFB 0
8044 00       0520        DEFB 0
8045 00       0530        DEFB 0
```

```
8046 00        0540        DEFB 0
8047 00        0550        DEFB 0
8048 00        0560        DEFB 0
8049 00        0570        DEFB 0
804A 07        0580        DEFB 7
804B 00        0590        DEFB 0
804C 09        0600        DEFB 9
804D 0A        0610        DEFB £A
804E 0B        0620        DEFB £B
804F 0C        0630        DEFB £C
8050 0A        0640        DEFB £A
8051 0E        0650        DEFB £E
8052 0F        0660        DEFB £F
8053 00        0670        DEFB 0
8054 11        0680        DEFB £11
8055 12        0690        DEFB £12
8056 13        0700        DEFB £13
8057 14        0710        DEFB £14
8058 00        0720        DEFB 0
8059 00        0730        DEFB 0
805A 00        0740        DEFB 0
805B 18        0750        DEFB £18
805C 00        0760        DEFB 0
805D 00        0770        DEFB 0
805E 1B        0780        DEFB £1B
805F 00        0790        DEFB 0
8060 00        0800        DEFB 0
8061 00        0810        DEFB 0
8062 0A        0820        DEFB £A ;LF FOR £1F
```

The DTR signal may go from +12V to -12V, it may go from 0V to +12V or it may go from 0V to +5V. It will depend upon the make of the printer. It is important to realise that applying +12 or -12V to your Lynx will do it a power of no good. Some form of voltage limiting and current limiting circuit is required. This is fairly straight-forward and the circuit given in the diagram will suffice. I normally mount the components inside one of the plugs so no PCB is necessary.

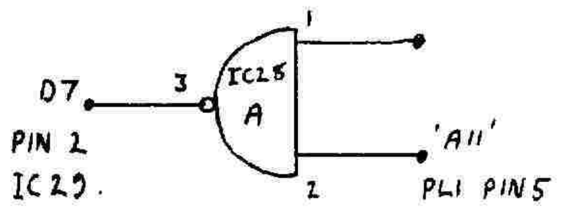


METHOD 2

The second method of providing the handshake signal is to solder four wires inside the Lynx. Please note that this invalidates your guarantee but it is quite a good way of solving the problem. The wires are soldered as follows:-

1. Pin 4 of the serial connector to pin 1 of IC28
2. The A11 keyboard select line (pin 5 of PL1) to pin 2 of IC28
3. Pin 3 of IC28 to D7 of the keyboard port which may be found on pin 2 of IC29.
4. Pin 5 of the serial port to a +5V supply

DTR INPUT SIGNAL



The drawback of this method is that it sets bit 7 on ports 0980H and 0880H to 0 whereas it would normally be 1. As a result if you run any BASIC programs which test for the cursor keys with lines like:-

100 IF INP(&0980)=232 THEN .....

the program will not work. One way around this is to make up a plug to replace the serial lead. It has pins 2 and 4 connected together. This ensures that bit 7 of ports 0880H and 0980H will be set.

```
8000            0010          ORG  8000H
                0020 ;SERIAL PRINT DRIVER ROUTINE
                0030 ;USING BIT 7 ON PORTS &0880 AND &0980
                0040 ;FOR A DTR HANDSHAKE SIGNAL
                0050 ;BY R.B.POATE
                0060 CR        EQU  0DH
                0070 LF        EQU  0AH
8000 FE80       0080 PRINT     CP   80H
8002 D0         0090           RET  NC ;DON'T PRINT GRAPHICS
8003 FE20       0100           CP   £20
8005 300E       0110           JR   NC,PRT1
                0120 ;A CONTROL CHARACTER
                0130 ;PICK UP CHARACTER FROM THE TABLE
8007 2A0262     0140           LD   HL,(£6202)
800A 012800     0150           LD   BC,TABLE-PRINT
800D 09         0160           ADD  HL,BC
800E 0600       0170           LD   B,0
8010 4F         0180           LD   C,A
8011 09         0190           ADD  HL,BC
8012 7E         0200           LD   A,(HL)
8013 B7         0210           OR   A
8014 C8         0220           RET  Z ;RETURN IF ZERO
8015 F5         0230 PRT1      PUSH AF  ;PROTECT CHARACTER
                0240 ;IS THE UART CLEAR?
8016 DB84       0250 TUART     IN   A,(£84)
8018 CB77       0260           BIT  6,A
801A 28FA       0270           JR   Z,TUART ;LOOP UNTIL CLEAR
```

```
                  0280 ;TEST THE DTR SIGNAL
801C 018008       0290      LD    BC,£0880 ;OR £0980
801F ED78         0300 LOOP IN    A,(C)
8021 17           0310      RLA
8022 38FB         0320      JR    C,LOOP ;LOOP UNTIL CLEAR
                  0330 ;NOW ITS 'READY' SEND THE CHARACTER
8024 F1           0340      POP   AF
8025 D382         0350      OUT   (£82),A ;SEND THE CHARACTER
8027 C9           0360      RET
8028 00           0370 TABLE DEFB 0
8029 00           0380      DEFB  0
                                .
                                .
                                .
       TABLE as above           .
```

METHOD 3

The third method involves using the interface given in Issue 2 as an input line for the handshake signal. It is very similar to the previous method except that the routine reads the handshake signal from D0 of port A.

```
8000              0010      ORG   8000H
8000 000D         0020 CR   EQU   0DH
8000 000A         0030 LF   EQU   0AH

                  0050 ;SERIAL PRINT DRIVER ROUTINE
                  0060 ;USING PARALLEL PORTS (N.NEWS ISSUE 2)
                  0070 ;FOR A DTR HANDSHAKE SIGNAL
                  0080 ;DTR SIGNAL COMES IN ON D0 OF PORT A
                  0090 ;BY R.B.POATE
8000 FE80         0100 PRINT CP   £80
8002 D0           0110      RET   NC
8003 FE20         0120      CP    £20
8005 300E         0130      JR    NC,PRT1
                  0140 ;A CONTROL CHARACTER
                  0150 ;PICK UP CHARACTER FROM THE TABLE
8007 2A0262       0160      LD    HL,(£6202)
800A 012500       0170      LD    BC,TABLE-PRINT
800D 09           0180      ADD   HL,BC
800E 0600         0190      LD    B,0
8010 4F           0200      LD    C,A
8011 09           0210      ADD   HL,BC
8012 7E           0220      LD    A,(HL)
8013 B7           0230      OR    A
8014 C8           0240      RET   Z ;RETURN IF ZERO
8015 F5           0250 PRT1 PUSH AF   ;PROTECT CHARACTER
                  0260 ;IS THE UART CLEAR?
8016 DB84         0270 TUART IN   A,(£84)
8018 CB77         0280      BIT   6,A
801A 28FA         0290      JR    Z,TUART ;LOOP UNTIL CLEAR
                  0300 ;TEST THE DTR SIGNAL
801C DB49         0310 LOOP IN    A,(£49) ;READ PORT A
801E 1F           0320      RRA
801F 30FB         0330      JR    NC,LOOP ;LOOP UNTIL CLEAR
                  0340 ;NOW ITS 'READY' SEND THE CHARACTER
8021 F1           0350      POP   AF
8022 D382         0360      OUT   (£82),A ;SEND THE CHARACTER
8024 C9           0370      RET
8025 00           0380 TABLE DEFB 0
                                .
                                .
                                .
          TABLE as above
```

SOME FINAL POINTS

The Lynx can have lines of up to 240 characters in length. Since most printers have either 80 columns or 132 columns unless some form of wrapping software is used long lines can't be printed. The following rountine can be placed in front of those already given to wrap long lines. It uses memory location 61C1H to store the number of characters printed. At first sight it might seem that this location would need to be zeroed before any printing could take place. In practice all that is required is for a carriage return to be printed which can be done with :- LPRINT[RETURN].

In the previous routines there is a line which reads 'LD BC TABLE-PRINT'. This will have to be changed to ' LD BC,TABLE-WRAP' if the wrap routine is placed in front.

```
8000              0010      ORG   8000H
8000 000D         0020 CR   EQU   0DH
8000 000A         0030 LF   EQU   0AH
8000 61C1         0040 COUNT EQU  £61C1 ;CHARACTER COUNT

8000 FE80         0060 WRAP CP    £80
8002 D0           0070      RET   NC ;DON'T PRINT GRAPHICS
8003 F5           0080      PUSH  AF ;SAVE THE CHARACTER
8004 FE0A         0090      CP    LF
8006 281F         0100      JR    Z,CRLF
8008 FE0D         0110      CP    CR
800A 281B         0120      JR    Z,CRLF
800C FE1F         0130      CP    £1F
800E 2817         0140      JR    Z,CRLF
8010 FE20         0150      CP    £20
8012 3817         0160      JR    C,END
                  0170 ;NORMAL CHARACTER
8014 21C161       0190      LD    HL,COUNT
8017 34           0200      INC   (HL)
8018 3E46         0210      LD    A,80 ;TEST PRINT WIDTH
801A BE           0220      CP    (HL)
801B 300E         0230      JR    NC,END
801D 3E0D         0260      LD    A,CR
801F 212B80       0270      LD    HL,END
8022 E5           0280      PUSH  HL ;DUMMY RETURN ADDRESS
8023 2A0262       0290      LD    HL,(£6202)
8026 E9           0300      JP    (HL)

8027 AF           0320 CRLF XOR   A
8028 32C161       0330      LD    (COUNT),A
802B F1           0350 END  POP   AF ;RETURN THE CHARACTER
```

You will also find that LINK ON/OFF will print spaces on any attached printer. The reason for this is that the cursor on the Lynx is actually 'printed' on the screen. Since the cursor is made up of two characters (20H and EFH) the space character (20H) is printed each time the cursor is blinked. The way around this is to ignore graphics characters (which the routines do) and to use two graphics characters for the cursor. This can be done with

CCHAR &80EF [RETURN].

As you should now realise, driving a serial printer from the Lynx is far from easy and this isn't the end of the story. What do you do if you wish to drive a device at a speed below 2400 baud? Perhaps I will answer that in another issue.

Machine Code for Beginners Part 3.

There are several problems associated with writing machine code programs in the way I have been doing it in this series. You may have noticed that once you have decided what register operations you want to perform you have the laborious task of finding the correct opcode in opcode tables. Admittedly you may get to learn some of them (it does come in time) but since there are more than 600 of them it is very unlikely that you will ever remember more than a few.

The second problem concerns jumps. Once the mechanics of calculating the jump address has been sorted out it is a fairly easy calculation to arrive at the jump value - but its very tedious. There must be a better way.

You will only fully appreciate the third point if you tried the last example in Part 2 ie relocating the machine code. All you had to do was re-work the given program to work somewhere else. Once again it should have been a straightforward exercise since all the logic was there to follow - but its another tedious task.

It can also be tedious to edit/modify machine code programs. If you tried to make the 'character blink' program blink much more slowly you must have re-worked some or all of the program.

Lastly machine code doesn't lend itself to what is known as DOCUMENTATION. This means putting your thoughts down so that you or someone else could understand what the program does and possibly modify it. This is one of the reasons the programmers prefer to use standard languages.

On the face of it there isn't much to recommend machine code programming. The way around most of the points raised is to use an ASSEMBLER. An assembler uses a standard language which is then translated (or assembled) into machine code.

ASSEMBLER LANGUAGE

Each of the 600+ opcodes has its own MNEMONIC to represent it in Z80 assembler language. The syntax of the language has been definded by Zilog (designers of the Z80) and since it is quite simple you will find that you will be able to remember the mnemonics to represent several hundred opcodes. The assembler is normally entered into a file (as are BASIC programs). Conceptually the file has four fields. The first is a LABEL field. This is followed by the OPCODE, the OPERAND and the COMMENT fields. Only the opcode field is mandatory. An example should clarify the fields.

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

Here the label is 'START', the opcode is 'LD', the operand is 'HL,0'. Finally the comment, which is separated by a semi-colon, explains what is happening. You will be able to find other examples in the rest of the magazine.

The label field is used to mark a location in the program. Normally this is used for subroutine calls, jumps or to mark the start of a section of the program. In some assemblers labels are followed a colon to separate the label field from the opcode field. The opcode field or the mnemonic field as it is sometimes called is where the mnemonic is placed which defines what operation is to be performed. The mnemonics have been defined by Zilog and they attempt to describe the operation so as to be memorable. So far you have met some of the opcode represented by the following mnemonics:-

```
ADD,CALL,LD,JP,RET,DJNZ
```

The first two don't need any explanation. The third (LD) stands for LOAD as in load one register with another. JP is short for JumP and RET is short for RETurn. DJNZ is an example of how difficult it is at times to form mnemonics which describe the operation. DJNZ stands for Decrement (B register) Jump relative if Not Zero. In actual fact this one is easy to remember because you will use is quite often, but not all mnemonics explain themselves quite as clearly.

The operand field is used to indicate on what the action is going to be done. It may be an address, or a register or both. Some mnemonics use two operands such as ' LD A,B'. Here the point to note is that it is the B register which is loaded into the A register and not the other way round. The comment field needs no explanation except to say that it is separated from the rest by a semi-colon.

PSEUDO-OPCODES

These are not really opcodes at all but since they may be used to define bytes within a program it is reasonable to call them pseudo-opcodes. Different assemblers will support a different set of these so I will only deal with some of the most common.

ORG

ORG stands for origin and it informs the assembler where you want the machine code to be assembled. So if you were to put ' ORG 9000H' as the first line in a file of source code the assembler would assemble your program at 9000H. Some assemblers have a default origin if you don't specify one.

DEFB

DEFB stands for define byte and it does exactly that. So if ' DEFB 31H' were in a source code file the assembler would put the byte 31 into the program. DEFB can be used for example to set up a data table.

DEFW

DEFW is similar to DEFB except that it defines a WORD instead of a byte. A word is two bytes.

DEFM

DEFM stands for define message and that is normaly what it is used for. So ' DEFM /This is a message/' entered into a source file would assemble the ASCII codes for the message.

EQU

EQU stands for equate. It is used to equate a name to a value. This helps to make the program more readable and easier to change. If you had a program dealing with a carriage return character you might set up an equate as follows:- 'CR EQU ODH'. From then on in the program you may use 'CR' whenever you wish to use the value for a carriage return. As an example to load the A register with the ASCII for a carriage return you would put:- ' LD A,CR'.

PORTS

The Z80 can perform two types of communication, internal and external. Internal communication is concerned with memory. External comunication is concerned with external devices such as keyboards, disks and tape recorders. External communication is done via PORTS.

So that you know which port to use each is numbered. Normally there are 256 ports on a Z80 machine and they are numbered 0 to FFH. On the Lynx the designers have chosen to design the hardware to recognise 65535 ports numberd from 0 to FFFFH. For obvious reasons not all ports are used - can you think of 65535 devices you could have linked up to your Lynx?!

THE KEYBOARD

The hardware of a machine decides which port you must use for a given operation. The I/O (input/output) maps for the Lynx were published in the first copy of LYNX USER. They show that the keyboard is attached to ten ports numbered 0080H, 0180H, 0280H up to 0980H. You can read in an 8-bit binary number from each port. The bits of the ports are attached directly to keys on the keyboard. Normally the bits of each port are held high (ie set to a 1) but when a key is pressed they go low (ie 0). So all that is necessary to find out which key is being pressed is to read all the ports and find out which is not set to FFH (ie all bit set to 1). Having found the port you can then look up the data read from the port in a look up table to find out which key(s) is down.

OUTPUT TO PORTS

Output to ports involves sending an 8-bit number to a selected port. Once again it is the hardware of the micro-computer which dictates to which port the data is sent. The 6-bit A/D output port (speaker) is on port number 84H. So to get some sound we need to send data to that port. Sound is produced by vibrating the speaker so the data must 'vibrate' or alternate between high and low values.

Some of the opcodes for port operations are given below.

MNEMONIC	OPCODE	EXPLANATION
IN A,(Port No)	DB Port No	Read data from port into the A register.
IN A,(C)	ED 78	Read data from port number in C register into the A register
OUT (Port No),A	D3 Port No	Send A out to port
OUT (C),A	ED 79	Send A to port (C)

Having opted to explain ports with reference to the keyboard and the sound port it seems rather logical to set up an 'electronic organ' using the Lynx keyboard and the sound port.

The program consists of a main routine which scans the keyboard and looks for keys which are pressed. Once it has found an input from the keyboard which isn't FFH (ie no keys pressed) it then calls a routine called FIND which looks for the key in a table. If it finds a match it picks up a wavelength and number of cycles from the table and plays the note.

There are a few instructions in the program which haven't been covered before. This is unfortunate and it demonstrates that you need to know a fair number of instructions before you can program in machine code. I will be covering them in subsequent parts of this series so for now I will give you a brief explanation as to what they do.

The first is CP. This stands to CompPare and it instructs the Z80 to compare the A register with the argument. Thus CP 80H would make a comparison between the A register and 80H. If the A register contained 80H then the zero flag would be set and this flag can then be used to direct program control.

The second new instruction is BIT. The BIT instruction tests to see if a bit is set. If it is not set (ie 0) then the zero flag will be set. Otherwise the zero flag will be reset to 0. Again this can be used to direct the flow of a program.

The CPL instruction flips the bits in the A register. When the keyboard is read you will get an 8-bit binary number with 0 for a key pressed. CPL is used to invert this hence giving a 1 to flag which key has been pressed.

The AND instruction performs a logical AND between the A register and the argument. This means that a bit by bit analysis is carried out on the A register and the argument. Only when both bits are set to a 1 will the result be a 1.

DJNZ you have met before but not as DJNZ. It was used in the first part of the series. It decrements the B register and tests to see if it is zero. If it is then you continue. If it is not zero then you make a relative jump according to the argument.

The final new instruction is XOR and it is used in the program to set the A register to zero. This could have been done with LD A,0 but that would take two bytes and XOR A only takes one so it is normal to use XOR A.

The program uses the following keyboard layout:-

```
         Q  W  E     T  Y     I  O  P     ]  del
     cu cd  A  S  D  F  G  H  J  K  L  :  :  cl  cr
```

Where cu,cd,cl and cr stand for cursor up,down,left and right.

The notes are as follows (sharps flagged by #):-

```
         F# G# A#    C# D#    F# G# A#    C# D#
     E  F  G  A  B  C  D  E  F  G  A  B  C  D  E
```

```
8000                ORG  £8000
8000 0006      WIDTH EQU  6 ;WIDTH OF TABLE
8000 0084      AUDIO EQU  £84  ;SPEAKER PORT
8000 0080      AVCONT EQU £80 ;AUDIO VISUAL CONTROL PORT
8000 0001      ON    EQU  1  ;ON/OFF FOR SPEAKER
8000 0000      OFF   EQU  0
8000 0004      CLS   EQU  4 ;CLEAR SCREEN VALUE

8000 3E04            LD   A,CLS
8002 CF              RST  8
8003 3E01            LD   A,ON
8005 D380            OUT  (AVCONT),A
8007 060A      SCAN  LD   B,10 ;SCAN THE KEYBOARD
8009 0E80            LD   C,£80
800B 05        SLOOP DEC  B
800C ED78            IN   A,(C) ;READ THE KEYBOARD PORT
800E FEFF            CP   £FF ;TEST IF ANY KEY PRESSED
8010 C5              PUSH BC ;SAVE BC VALUE
8011 C42580          CALL NZ,FIND ;FIND WHICH KEY PRESSED
8014 C1              POP  BC ;RETURN BC
8015 04              INC  B  ;RESET B
8016 10F3            DJNZ SLOOP ;BACK FOR NEXT PORT
               ;TEST FOR ESCAPE - B IS NOW ZERO
8018 ED78            IN   A,(C) ;READ PORT £0080
```

```
801A CB77     0250        BIT  6,A
801C 20E9     0260        JR   NZ,SCAN
801E 3E00     0270        LD   A,OFF ;SWITCH SPEAKER OFF
8020 D380     0280        OUT  (AVCONT),A
8022 D384     0290        OUT  (AUDIO),A
8024 C9       0300        RET
8025 2F       0320 FIND   CPL  ;FLIP ALL BITS IN A REGISTER
8026 4F       0330        LD   C,A
8027 216E80   0340        LD   HL,TABLE-WIDTH
802A 110600   0350 FLOOP  LD   DE,WIDTH
802D 19       0360        ADD  HL,DE
802E 7E       0370        LD   A,(HL) ;LOAD WHAT HL
                                     POINTS TO INTO A
802F FEFF     0380        CP   £FF ;END MARKER
8031 C8       0390        RET  Z ;TEST FOR END OF TABLE
8032 B8       0400        CP   B ;TEST FOR PORT NUMBER
8033 20F5     0410        JR   NZ,FLOOP ;ANOTHER TRY?
8035 23       0420        INC  HL ;POINT HL TO BIT
                                 COMBINATION
8036 7E       0430        LD   A,(HL) ;LOAD INTO A
8037 A1       0440        AND  C ;LOOK FOR ONE AT A TIME
8038 BE       0450        CP   (HL);IS THIS KEY DOWN?
8039 2B       0460        DEC  HL ;DEC HL IN CASE IT ISN'T
803A 20EE     0470        JR   NZ,FLOOP
              0480 ;FOUND IT!
              0490 ;PICK UP DELAY AND FREQUENCY
803C 23       0500        INC  HL
803D 23       0510        INC  HL ;HL POINTS TO WAVELENGTHS
803E 5E       0520        LD   E,(HL)
803F 23       0530        INC  HL
8040 56       0540        LD   D,(HL)
8041 D5       0550        PUSH DE ;THE WAVELENGTH
              0560 ;NOW PICK UP THE NUMBER OF CYCLES
8042 23       0570        INC  HL
8043 5E       0580        LD   E,(HL)
8044 23       0590        INC  HL
8045 56       0600        LD   D,(HL) ;DE HAS No OF CYCLES
8046 E1       0610        POP  HL ;THE DELAY
8047 23       0620        INC  HL
8048 24       0630        INC  H
8049 2C       0640        INC  L
804A 2D       0650        DEC  L
804B 2001     0660        JR   NZ,BEEP
804D 2C       0670        INC  L
804E AF       0680 BEEP   XOR  A ;ZERO A
804F D384     0690        OUT  (AUDIO),A  ;SEND TO SPEAKER
8051 CD6B80   0700        CALL DELAY
8054 3E3F     0710        LD   A,63 ;MAX VOLUME FOR SPEAKER
8056 D384     0720        OUT  (AUDIO),A
8058 CD6B80   0730        CALL DELAY
805B C5       0740        PUSH BC ;TEST FOR ESCAPE
805C 018000   0750        LD   BC,£0080
805F ED78     0760        IN   A,(C)
8061 CB77     0770        BIT  6,A
8063 C1       0780        POP  BC
8064 C8       0790        RET  Z
8065 1B       0800        DEC  DE ;DECREMENT WAVELENGTH
8066 7A       0810        LD   A,D ;TEST IF ITS ZERO
8067 B3       0820        OR   E
8068 20E4     0830        JR   NZ,BEEP
806A C9       0840        RET

806B E5       0860 DELAY  PUSH HL
806C 2D       0870 DEL1   DEC  L
806D 20FD     0880        JR   NZ,DEL1
806F 25       0890        DEC  H
8070 20FA     0900        JR   NZ,DEL1
8072 E1       0910        POP  HL
8073 C9       0920        RET
              0940 ;A LOOK-UP TABLE FOR KEYS/NOTES
              0950 ;BYTE 1 HAS THE PORT NUMBER FOR THE KEY
              0960 ;BYTE 2 HAS THE BIT COMBINATION
              0970 ;BYTES 3 & 4 HAVE THE WAVELENGTH
              0980 ;BYTES 5 & 6 HAVE THE NUMBER OF CYCLES
              0990 ;THE TABLE IS ENDED BY BYTE £FF
8074 0020     1000 TABLE  DEFB 0,£20;E:DOWN ARROW
8076 9E021000 1010        DEFW 670,16 ;E
807A 0010     1020        DEFB 0,£10;F:UP ARROW
807C 73021100 1030        DEFW 627,17
8080 0202     1040        DEFB 2,2;F#:Q KEY
8082 51021300 1050        DEFW 593,19
8086 0220     1060        DEFB 2,£20;G:A KEY
8088 32021400 1070        DEFW 562,20
808C 0204     1080        DEFB 2,4 ;G#:W KEY
808E 12021500 1090        DEFW 530,21
8092 0210     1100        DEFB 2,£10 ;A: S KEY
8094 F3011600 1110        DEFW 499,22
8098 0104     1120        DEFB 1,4 ;A#:E KEY
809A D3011700 1130        DEFW 467,23
809E 0110     1140        DEFB 1,£10;B:D KEY
80A0 BB011900 1150        DEFW 443,25
80A4 0320     1160        DEFB 3,£20 ;MID C:F KEY
80A6 A3011A00 1170        DEFW 419,26
80AA 0304     1180        DEFB 3,4;C#:T KEY
80AC 8D011C00 1190        DEFW 397,28
80B0 0310     1200        DEFB 3,£10 ;D:G KEY
80B2 76011D00 1210        DEFW 374,29
80B6 0402     1220        DEFB 4,2 ;D#:Y KEY
80B8 60011F00 1230        DEFW 352,31
80BC 0404     1240        DEFB 4,4 ;E: H KEY
80BE 4D012100 1250        DEFW 333,33
80C2 0520     1260        DEFB 5,£20 ;F:J KEY
80C4 37012300 1270        DEFW 311,35
80C8 0602     1280        DEFB 6,2 ;F#:I KEY
80CA 27012500 1290        DEFW 295,37
80CE 0620     1300        DEFB £6,£20 ;G: K KEY
80D0 16012700 1310        DEFW 278,39
80D4 0604     1320        DEFB 6,4;G#: O KEY
80D6 06012A00 1330        DEFW 262,42
80DA 0704     1340        DEFB 7,4 ;A:L KEY
80DC F8002C00 1350        DEFW 248,44
80E0 0702     1360        DEFB 7,2 ;A#:P KEY
80E2 E8002F00 1370        DEFW 232,47
80E6 0720     1380        DEFB 7,£20 ;B :";"KEY
80E8 DC003100 1390        DEFW 220,49
80EC 0820     1400        DEFB 8,£20 ;C:":" KEY
80EE CE003400 1410        DEFW 206,52
80F2 0902     1420        DEFB 9,2 ;C#:"]" KEY
80F4 C3003700 1430        DEFW 195,55
80F8 0904     1440        DEFB 9,4;D:LEFT ARROW KEY
80FA B9003B00 1450        DEFW 185,59
80FE 0901     1460        DEFB 9,1 ;D#:DEL KEY
8100 AD003E00 1470        DEFW 173,62
8104 0920     1480        DEFB 9,£20;E: RIGHT ARROW
8106 A3004C00 1490        DEFW 163,76
810A FF       1500        DEFB £FF
```

LYNX EXPANSION By C. Cytera

The Lynx was designed so as to be expandable at a low cost. The following article demonstrates this by describing how it can be upgraded from 48K to 96K for less than £40.

You will need the following items:-

1. Eight 4164 dynamic RAM chips. These are readily available from suppliers such as Midwich and Technomatics. You can also use 4864 RAMs instead.
2. A soldering iron.
3. A pair of fine nosed pliers.
4. A small screwdriver.
5. Some solder and connecting wire.

Do not be put off by the need for some soldering: it is very simple and the computer cannot be damagged unless you are extremely clumsy and careless.

To expand your Lynx, read the instructions that follow, and proceed carefully, step by step.

1. Remove all cables connected to the Lynx.

2. Remove the four Philips screws holding the case together, and turn the computer so that the keyboard faces upwards, ie in the normal direction.

3. Detach the keyboard, and lay it upside-down in front of the machine. You will see that it is connected to the printed circuit board by means of a ribbon cable. Unplug this cable from the board.

4. Just below the keyboad connector, you will see eight 4116 RAMs arranged as two rows of four. This is the 16K workspace RAM in the 48K Lynx. Remove these chips as carefully as you can. If you do not have an IC insertion/extraction tool, gently lever the chips out using a screw-driver at either end. Do not bend or touch the legs, as you may need these ICs some time in the future. Also be careful not to damage the nearby ceramic decoupling capacitors.

5. Now refer to the diagrams. Locate the links to the left of the now empty RAM sockets. Using the screwdriver, or a better tool if you have one, cut the tracks which I have labelled. This disconnects the -5v supply from pin 1, which is not required by the 4164. If possible use an ohm-meter to ensure that the track is cut.

6. The arrangement of links which I have labelled 2 should now be changed from configuration A to configuration B. The diagram printed on the Lynx's PCB should make clear what has to be done. First, cut the tracks bridging the lower two pairs of connection points on the board. This disconnects the +12v supply from pin 8, and the +5v supply from pin 9. Again ensure that the tracks are clearly cut.

7. Using two short pieces of connection wire make two soldered links bridging the top two pairs of connection points on the board. This will connect +5v to pin 8 of the 4164s, and A7, the extra address line, to pin 9. Make sure that your joints are shiny, and check them for continuity.

8. Taking extreme care, remove the 4164s from their protective packing and insert them one by one into the sockets vacated by the 4116s. Make sure that pin 1 is at the top. The best way to make the chips fit is to press one side against a flat surface, bending in eight of the splayed-out pins. Handle the RAMs with great caution, as they are MOS and can be damaged by static electricity.

9. Plug the keyboard back into the PCB. Note that there is a spare socket on the end of the ribbon cable which does not connect to the board. Now place the keyboard right-way-up on top of the machine. Do not screw it on yet; it is better to check the memory first.

Now for the moment of truth: reconnect and switch on the Lynx. If it does not work, then either the tracks have not been properly cut, or the links have been made incorrectly. If all is well so far type:- PRINT MEM [RETURN]. The Lynx should respond with 38558. Now type in the following program:-

```
10 FOR A=&7000 TO &FFC0
20 R=RAND(256)
30 POKE A,R
40 IF PEEK(A)<>R THEN PRINT "FAIL AT ";(hash)A
50 NEXT A
```

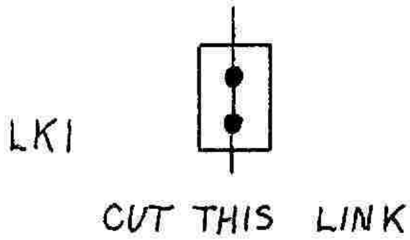
(NB I don't have hash on my printer; see line 40.- ED.)

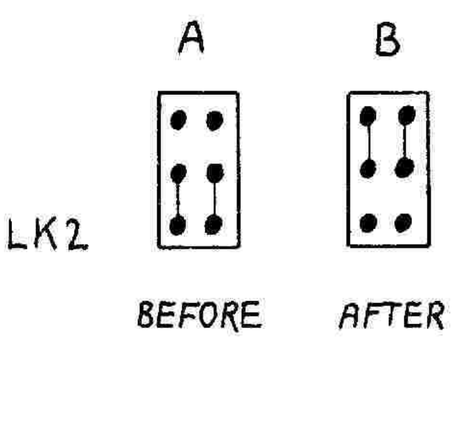
This will test most of the memory. If the normal prompt appears after several minutes, then the expansion has been successful. If you find that one bit in every memory location is incorrect, then the fault could be a faulty chip or an incorrectly inserted one. If just a few memory locations are not functioning, then you certainly have a faulty RAM.

Once you are satisfied with your extra memory, screw the keyboard back on and put the eight 4116s in the protective packing in which the 4164s were supplied.

Although you have inserted 64K of RAM, making a 96k Lynx, 24K is overlaid with the space allocated for the ROMs. This therefore is not usable without switching out the ROMs from the memory map.

DIAGRAMS





3-D ROTATION By Chris Cytera

The following program by Chris Cytera allows you to define and then rotate objects. Points are defined using coordinates in the X,Y and Z planes which are then joined with lines to create an object. Once defined the object will be displayed and it may then be rotated about any axis by using the cursor, '<' and '>' keys. You may also zoom in and out using the 'I' and 'O' keys.

To define an object you need to set up a number of coordinate positions in a DATA statement to define the points. The DATA statement must be preceded by the number of points. See line 20 of the program and Diagram 1. The second stage is to define the lines which join the points. This is done by giving pairs of points which are to be joined. Once again the DATA statement is preceded by the number of lines required. See line 40.

Following Chris' program you will find some modifications which I have made and a new procedure called 3D. I soon got tired of working out all the coordinates of a complex shape and decided to write a procedure which would rotate a two dimentional 'shape' about the Z-axis to generate objects. To do this you must modify the 'lines' and 'points' procedures to pick up and store two-dimentional data. The procedure 3D then takes each point in turn and rotates it to produce 12 lines to form a circle. It then generates the lines which will complete the object.

Generating objects is done by defining the points in the X-Y plane and the lines which will join them. Points are defined by X and Y coordinate pairs in line 20. The first value in the DATA statement is the number of points. Line 40 is used to define the lines which will join the points. Once again the first value is the number of lines. See Diagram 2.

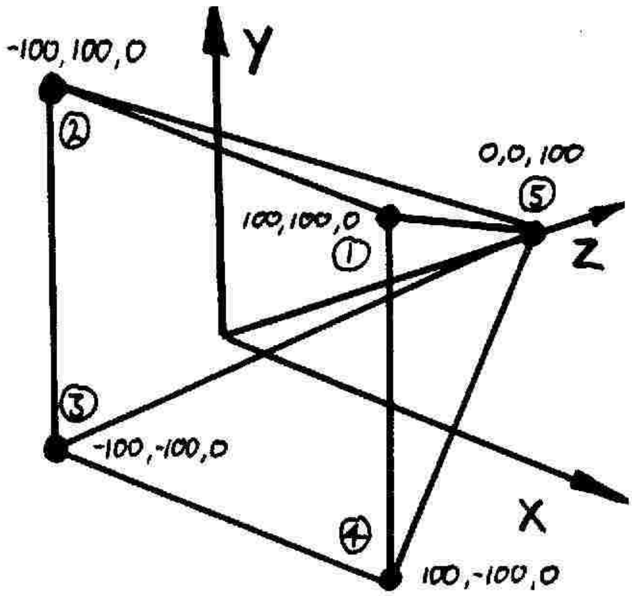


DIAGRAM 1

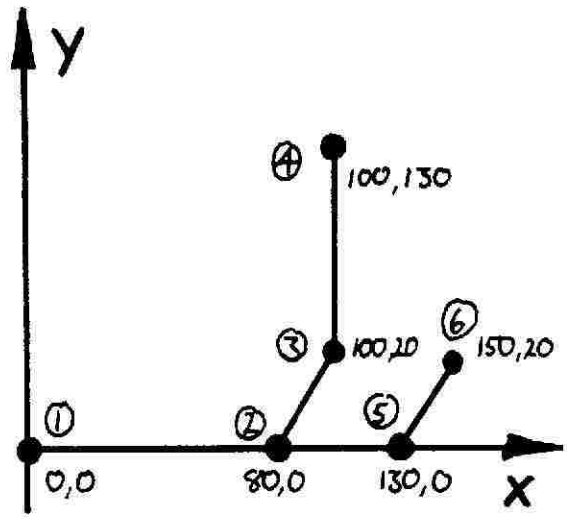


DIAGRAM 2

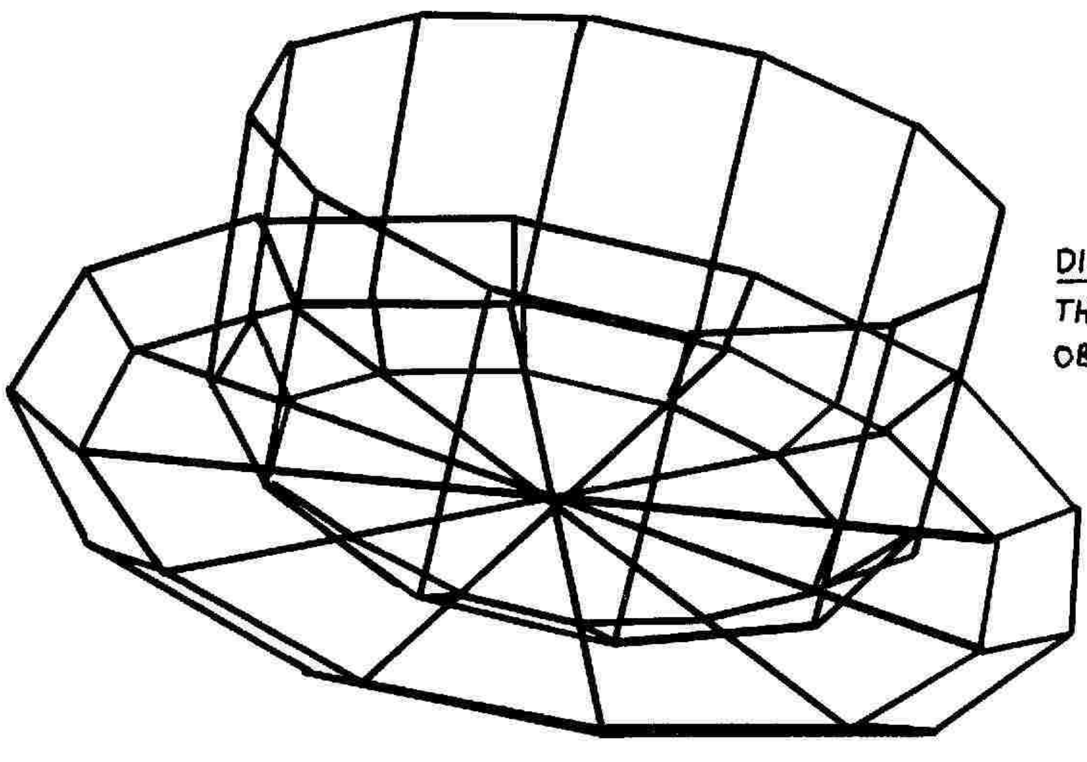


DIAGRAM 3
THE GENERATED OBJECT

```
10 REM Points data
20 DATA 5,100,100,0,-100,100,0,-100,
-100,0,100,-100,0,0,0,100
30 REM Lines data
40 DATA 8,1,2,2,3,3,4,4,1,5,1,5,2,
5,3,5,4
1000 PROC points
1010 PROC lines
1020 LET D=1000
1030 LET N=100
1040 LET A=PI/16
1050 REPEAT
1060   PROC 2D
1070   PROC draw
1080   PROC update
1090   PROC rotate
1100 UNTIL FALSE
1110 DEFPROC points
1120 REM Dimension point arrays and
read in points data
1130 READ R
1140 DIM X(R),Y(R),Z(R),x(R),y(R)
1150 FOR C=1 TO R
1160   READ X(C),Y(C),Z(C)
1170 NEXT C
1180 ENDPROC
1190 DEFPROC lines
1200 REM Dimension line arrays and
read in lines data
1210 READ L
1220 DIM S(L),E(L)
1230 FOR C=1 TO L
1240   READ S(C),E(C)
1250 NEXT C
1260 ENDPROC
1270 DEFPROC 2D
1280 REM Convert to 2D
1290 FOR C=1 TO R
1300   LET x(C)=X(C)*500/(D-Z(C))
1310   LET y(C)=Y(C)*500/(D-Z(C))
1320 NEXT C
1330 ENDPROC
1340 DEFPROC draw
1350 CLS
1360 FOR C=1 TO L
1370   PROC MVFIX(128+x(S(C)),124-y(
S(C)),128+x(E(C)),124-y(E(C)))
1380   MOVE x,y
1390   PROC DRFIX(h,k)
1400   DRAW x,y
1410 NEXT C
1420 ENDPROC
1430 DEFPROC update
1440 LET F=0,T=0,P=0
1450 REPEAT
1460   LET K=GETN
1470 UNTIL K=22 OR K=12 OR K=10 OR K=11
OR K=41 OR K=40 OR K=79 OR K=73
1480 REM Rotate about X axis ?
1490 IF K=10 THEN  LET F=A
1500 IF K=11 THEN  LET F=-A
1510 REM Rotate about Y axis ?
1520 IF K=22 THEN  LET T=A
1530 IF K=12 THEN  LET T=-A
1540 REM Rotate about the Z axis ?
1550 IF K=41 THEN  LET P=-A
1560 IF K=40 THEN  LET P=A
1570 REM Change viewing distance ?
1580 IF K=79 THEN  LET D=D+N
1590 IF K=73 THEN  LET D=D-N
1600 ENDPROC
1610 DEFPROC rotate
1620 IF F<>0 THEN  PROC Xrotation
1630 IF T<>0 THEN  PROC Yrotation
1640 IF P<>0 THEN  PROC Zrotation
1650 ENDPROC
1660 DEFPROC Xrotation
1670 REM Rotate about X axis
1680 LET H=COS(F),J=SIN(F)
1690 FOR C=1 TO R
1700   LET Y=Y(C),Z=Z(C)
1710   LET Y(C)=Y*H-Z*J
1720   LET Z(C)=Z*H+Y*J
1730 NEXT C
1740 ENDPROC
1750 DEFPROC Yrotation
1760 REM Rotate about Y axis
1770 LET U=COS(T),V=SIN(T)
1780 FOR C=1 TO R
1790   LET X=X(C),Z=Z(C)
1800   LET X(C)=X*U-Z*V
1810   LET Z(C)=Z*(U)+X*V
1820 NEXT C
1830 ENDPROC
1840 DEFPROC Zrotation
1850 REM Rotate about Z axis
1860 LET Q=COS(P),S=SIN(P)
1870 FOR C=1 TO R
1880   LET X=X(C),Y=Y(C)
1890   LET X(C)=X*Q-Y*S
1900   LET Y(C)=Y*Q+X*S
1910 NEXT C
1920 ENDPROC
1930 DEFPROC DRFIX(x,y)
1940 PROC MVFIX(x,y,PEEK(&6265),
PEEK(&6267))
1950 ENDPROC
1960 DEFPROC MVFIX(x,y,h,k)
1970 IF x=h THEN  LET m=INF
1980 ELSE  LET m=(y-k)/(x-h)
1990 IF x<0 THEN  LET x=0,y=k-h*m
2000 IF x>255 THEN  LET x=255,
y=m*(255-h)+k
2010 IF y<0 THEN  LET y=0,x=h-k/m
2020 IF y>255 THEN  LET y=255,
x=(255-k)/m+h
2030 ENDPROC
```

```
10 REM Points data
20 DATA 6,0,0, 80,0,100,20,100,130,
130,0,150,20
30 REM Lines data
40 DATA 5,1,2,2,3,3,4, 2,5,5,6
1000 PROC points
1010 PROC lines
1015 PROC 3D
1020 LET D=1000
1030 LET N=100
1040 LET A=PI/16
1050 REPEAT
1060   PROC 2D
1070   PROC draw
1080   PROC update
1090   PROC rotate
1100 UNTIL FALSE
1110 DEFPROC points
1120 REM Dimension point arrays and
read in points data
1130 READ N
1140 DIM M(N),N(N)
1150 FOR C=1 TO N
1160   READ M(C),N(C)
1170 NEXT C
1180 ENDPROC
1190 DEFPROC lines
1200 REM Dimension line arrays and read
in lines data
1210 READ M
1220 DIM s(M),e(M)
1230 FOR C=1 TO M
1240   READ s(C),e(C)
1250 NEXT C
1260 ENDPROC
```

```
1395 IF C>p*12 THEN  INK RED
1396 ELSE  INK YELLOW
```

```
3000 DEFPROC 3D
3005 LET R=12*N,L=12*(N+M)
3006 DIM X(R),Y(R),Z(R),x(R),y(R),z(R),
S(L),E(L)
3010 LET L=1
3020 FOR I=1 TO N
3030   FOR J=1 TO 12
3040     LET X(J+(I-1)*12)=INT(M(I)*
COS(RAD(J*30)))
3050     LET Y(J+(I-1)*12)=N(I)
3060     LET Z(J+(I-1)*12)=INT(M(I)*
SIN(RAD(J*30)))
3070     LET S(L)=L
3080     LET E(L)=S(L)+1
3085     IF 0=(L MOD 12) THEN
LET E(L)=S(L)-11
3087     LET L=L+1
3090   NEXT J
3100 NEXT I
3110 FOR I=1 TO 12
3120   FOR J=1 TO M
3140     LET S(L)=I+(s(J)-1)*12
3150     LET E(L)=I+(e(J)-1)*12
3155     LET L=L+1
3160   NEXT J
3170 NEXT I
3180 LET L=L-1
3190 LET p=N
3200 ENDPROC
```

LYNX 48K SOFTWARE

DISASSEMBLER £4.75 (£4.25)
This basic utility program enables you to examine machine code programs in ROM or RAM in standard Z-80 mnemonics.
Features:– Full disassembler or relative and absolute jumps, option of disassembly in decimal or hexedecimal notation.

LABYRINTH £4.75 (£4.25)
This is a top quality 3-D maze game made possible by the superb graphic capabilities of the LYNX. You have to think spatially to find your way through the many complex mazes in the shortest route. Do not despair, help is at hand if needed.

CHANCELLOR £4.75 (£4.25)
This program provides a realistic simulation of the UK economy, working from either one of two economic theories (which are fully explained). Dare you unleash your ideas upon the unsuspecting population? Can your policies work? Try to remain in office for the full term of ten years.

REVERSALS £4.75 (£4.25)
This is our version of the popular board game, an exciting, fast and highly addictive game of logic and skill, easy to learn yet hard to master. The program is written in machine code, so no more long waits between moves.

SPACE TREK £4.75 (£4.25)
Space – the final frontier. You are in command of the Star Ship Enterprise, and the Federation of Planets is in grave danger from invading forces. Your mission – to save the Universe by seeking out and destroying the evil Klingons.

THE WORM £5.95 (£5.35)
Our best yet! Guide Wilberforce the worm to the flowers which are his staple diet. The problem is that he will try to eat rocks, walls and himself all of which are fatal. He also moves at incredible speed.

Nilug members price in brackets – quote mem. No.

QUAZAR COMPUTING DEPT. N.
29 WESTERN RD., NEWICK, E. SUSSEX
overseas – add 10%
Other programs available send for details

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

Send s.a.e. for FREE CATALOGUE
SOFTWARE (Quazar) Labyrinth, Chancellor, Reversals, Space Trek, Disassembler £4.75 e.a. The Worm £5.95; (Level 9) Colossal Adventure, Adventure Quest, Dungeon Adventure, Snowball £9.90 e.a. (Gem) Monster Mine, Golf, Sultan's Maze, Gempack IV £7.95 e.a. Discounts off total; 2 off = 50p; 3 off = £1; 4 off = £2. BOOKS "Lynx Computing" £6.95 HARDWARE parallel connector + computer reset £4.50 DUST COVERS £2.75 CASSETTE LEADS DIN to DIN + remote or DIN to 3 jacks £2.35

FULCRUM PRODUCTS
14 STEEP LANE, FINDON
W SUSSEX BN14 0UF

ZEN

EDITOR • ASSEMBLER • DEBUGGER

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

STATE WHICH MACHINE YOU HAVE AND 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 £14.95 DATA CARDS ONLY £4.95.

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.