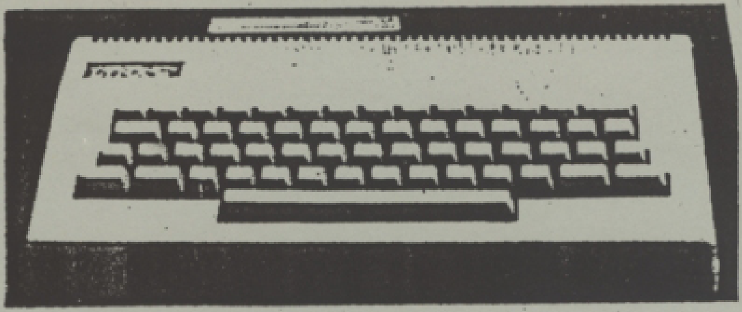
LYNX USER GROUP



MAGAZINE

Vol. 1 Issues 5&6.

CONTENTS

All enquiries and correspondence to:-
LYNX USER GROUP (LUG), 39 ASHTON CLOSE, NEEDINGWORTH,
St.IVES, CAMBS PE17 3UA.   Tel: 0480-68339

Editorial

****

SURPRISE SURPRISE SURPRISE !!!

Here at long last (originally scheduled for mid '85!) are the final issues of the magazine, nos 5 and 6, combined into one bumper issue.

This now completes the subscription period for '84 and '85. Since the group has been in "limbo" for 4 years, there has still been support from various sources, these have been:-

1) Exhibition support by myself and helpers for;
PCW show Olympia '85
PCW show Olympia '86
Alternative Micro show Birmingham, Nov '88
Alternative Micro show London, Apr '89 and shortly Alternative Micro show Stafford, Nov '89.
Only '87 was missed.

2) Exhibition support by the Reading LUG for;
Alternative Micro show Birmingham, Nov '88.
Alternative Micro show London, Apr '89.

3) A series of user articles, compiled by Chris Mathews from various sources, available free if you send an SAE (C4 size). This alleviated to a certain extent the absence of the LUG magazine.

Other developments have taken place like;

4) Two versions of BANK 4 DRAM expansion, created by;
Ken Halpin, software only for the 128K at present using BIOS 1.1 under CP/M and also LYNX DOS. NOT bug free but hardware works!
J Koushapis using different, scarce and expensive DRAMS, the 4464s!
At the time of writing I have completed the PCB track layout for Ken's expansion but am waiting for a non-distorted output from a printer!

5) The Reading LYNX USER GROUP now has CP/M working in 80 columns (by linking Green and Alternate Green together), for the 96K LYNX.

6) Work is now in hand for *SUPERLYNX*, a new set of ROMs vers. SL 0.1 having been successfully blown. This in principle moves start of BASIC to &6A00, resulting in a much enlarged vector/pointer table, while only using a very small portion of the normal workspace. Over 900 bytes of room has been created in the second ROM by shifting code up to &5000 in the third ROM. This now allows sideway ROM changing to be implemented.

7) Several more "discoveries" have been made, eg, a byte missing from the OUTBYTE routine, only minor, but could be important. It has also been discovered that XORing the graphics only requires a further 15 bytes of code! Much more could be mentioned but there is other news.

8) Because of 6) above, and other facilities, it is to be hoped that the *SUPERLYNX* hardware is not that far off.

9) New commands have now been written, i.e. ON GOTO, ON GOSUB and ON ERROR, the latter being perhaps the more important of the three.

10) A custom IC is under development to provide hardware spriting, only one port and half a byte to activate. What is perhaps more exciting is that it can be fitted to ANY computer of any make supporting RGB outputs. This should allow very fast graphic effects, for the Z80, only 12 T-states (just the OUT instruction!) or a change within 15 pixels for the 96K LYNX!

Perhaps one of the best items of software for the 96K machine, has been John Ridgways "ANIM", which allows the user to create their own animation images cycling through a maximum of 12 individual frames and with the option of 9 different speeds. This is currently vers. 2.0 but he tells me he is now working on vers. 3.0, with many more powerful enhancements. This is disk only.

As many of you now know, I have moved to the fens of Cambridgeshire but for those who don't know, my current address is:-

39-ASHTON CLOSE,
NEEDINGWORTH,
St.IVES,
CAMBS. PE17 3UA

My new number is:-

0480-68339

but I am only available after 6pm and generally at weekends. Yes I have a job!

Editor- R B JONES

LYNX BOOKS
****

To date, only three books have been published which deal specifically with the LYNX. They are:-

SINCLAIR I. "LYNX COMPUTING"
London:GRANADA,1983. ISBN 0246121319

CHAPMAN F. "Learn. to use the LYNX computer"
Aldershot:GOWER,1983. ISBN 0566034913

JEDOWSKI S. "Getting the most from your LYNX"
Harmondsworth:Penguin.ISBN 0140078118

All three are very much beginner's guides and those by Sinclair and Chapman only deal with the 48KLYNX because they were published soon after the machine's launch. Jedowski's book includes a 13 page section devoted to the 96K and 128K versions, dealing largely with the additional BASIC commands available.

For children and anyone completely new to microcomputing, the book by Chapman is a good introduction - in effect a simplified version of the manual supplied with the machine. After a brief explanation of the hardware, which rather illogically includes instructions on editing before anything has been keyed in to edit, it covers simple BASIC programming (in 15 pages) and then graphics in more detail. The concluding chapter is a muddled mixture, covering specifications and internal construction (which surely should have been put in the hardware section at the beginning), sound and "using the LYNX as a timer"(!). The book is clearly written, there are plenty of example programs and routines, and the illustrations of screen displays are mostly relevant to the LYNX, although some would benefit from being in colour. My main complaint is that I can't get the "Tower of Hanoi" program at the end to work.

"LYNX Computing" is a more substantial work, considerably expanding and improving on the user manual. It was reviewed by both M.Lawson and R.B.Poate in "Nilug News", issue 2, and I do not feel I can add much to their views, except to say that I bought it just after buying my LYNX and have found it to be well written, amusing and above all useful despite some minor errors which are presumably a consequence of its hasty production.

Steven Jedowski's book shows the benefits of being written after the LYNX had been on the market for some time: many routines and useful tips which first appeared in various magazines and newsletters have been incorporated and the memory maps for the three models are brought together in one place. The book covers BASIC programming in some detail over four chapters and, unlike the other books, suggests programs to write based on the material in each chapter, with sample solutions. There is also a good explanation of graphics but the section on sound is not as comprehensive as that in "LYNX Computing". As a beginner, I found the chapter on how the LYNX computer works inside very well written, although I am puzzled as to why it, and that on peripherals (Add-ons), have been put in the middle of explanations of the BASIC language. There are four pages of colour illustrations but they add little to the text and there is one glaring omission - no index! I also found the typeface rather small if you are reading the book while using your computer, although the actual listings are clear enough. Like the other two, it is a pity that all three are paperbacks, which are difficult to keep open without damaging their spines. No doubt economic considerations prevent publication in hardback but some kind of spiral binding (as used for the user manual) would be a great help.

If I had to recommend one book for the LYNX, at present I would go for "Getting the most from your LYNX" as the best all round value for beginners but "LYNX Computing" comes a close second and neither would be without interest to experts as well. I sincerely hope that the plans currently being made to ensure the LYNX's survival come to fruition, and that these do not turn out to be the only books to be written on this very good machine.
J.NEWTON.

WASP from ROMIK software

"WASP" loaded first time on TAPE 3, then auto-ran, followed by a welcome screen. It then asks you to press any key, which produced an explanation of the game. The idea of this game is to shoot 7 waves of nine "nasties" and then go on to destroy the WASP, which is invading your planet. You have three lives in which to do this, the screen also explains the scoring system, i.e. You score 10 x Wave number for each nastie. The waves are in this order and get progressively harder:-
1: FLYMEN 2: FIREBIRDS 3: BUBBLE BUGS 4: MUTANTS
5: SNAKES 6: STARSHIPS 7: ROX
and last of all is the WASP!

After you have scored more than 3500 points, you are entitled to have a Hi-score (assuming that there is no previous one). There is only one Hi-score at a time, which is a pity. If you have beaten the Hi-score you are invited to enter your name to go beside your score. This screen also gives the controls and the following one gives a joystick option and the choice of three speeds 1-3, 1 being the slowest.

After the game starts, the first wave appears and the idea is to shoot all 9 of the FLYMEN down and then to progress to the FIREBIRDS and so on, while avoiding being killed yourself by the bullets or the nasties themselves. If you do get killed you have to start that particular screen from scratch which is very inconvenient.

The graphics are very good, smooth and flicker free, the best graphics I have seen so far are the ROX on the seventh screen (but I haven't reached the WASP yet!), nine large asteroids appear and after hitting them, they break up into smaller rocks which can then be destroyed. Another aspect is that after a short time, the formation breaks up and the smaller pieces then fly all over the place. Very dangerous on the third wave!

Overall it is an excellent game and great value for money at £4.99 so I award it:- Value for money = 8/10 and overall = 8/10
P.S. Can anyone beat my Hi-score of 6790?
R HARRIES-HARRIS

RALLY BRITAIN
****

On receiving the cassette it appears somewhat bland, having no sleeve to the cassette. However once loaded it is a different matter. The program loads in two sections at tape 0. Whilst loading the second section a map of Great Britain is drawn and the message "Loading Rally Britain" is displayed. Loading timeis about three minutes. Once loaded a destination town or city is displayed along with your current position. The object of the game is to get to the destination in less miles than the "Target" milage given by the computer. A small menu of towns are displayed and you have to choose which is the correct one on your route. Your current location is always displayed on the map by a flashing dot. Having chosen your town the menu changes to a new selection of towns you can go to from the last town chosen. This process is repeated until you reach the destination. If you succeed, one of two messages appear, either "Well Done" or "Phew it's close". If you fail "Missed a bit!" appears.

Despite the lack of sleeve to the cassete, I feel the overall presentation is good. It's a good, interesting educational game for all ages. I have even caught my wife playing the game, and she displays a distinct lack of interest in computers.

USE OF GRAPHICS	7
USE OF SOUND	3
PLAYABILITY	9
VALUE FOR MONEY	8

A.R. BRISTOW

RELOCATABLE PARALLEL PRINT ROUTINE

I have a Parallel Printer interface, which I use with a 48K machine expanded to 96K (as per NILUG NEWS Issue 4) so it does not have the print routine resident in ROM. On the other hand the print routine supplied for the 48K sits inconveniently in the middle of user RAM. Apart from this problem I had a number of criticisms of the routine supplied:

1) It cannot be used in conjunction with any program that has Graphics characters, or machine code routines stored above HIMEM.

2) My printer (a SANPLE DAISY STEP 2000) uses all the control codes in conjuction with ESCape sequences, but produces unwanted characters when some of them are used seperately. In particular characters CHR$(29) through CHR$(31). This is unfortunate in that the LYNX sends a CHR$(31) followed by a CHR$(30) for carriage return.

3) As my printer is a Daisywheel, I don't usually want the graphics. The exception being when I am working with alternative character sets.

The answer to all of these problems was to produce a completely relocatable routine that can be adapted to reject some or all of the control codes unless they immediately follow an ESCape. It must also be possible to make the routine reject Graphics characters. This routine could then be put ionto CODE lines and appended to a BASIC program when needed. Alternatively it could be used in machine code programs by entering it through the Monitor.

The following routine answers all of these requirements:

```
1E+63 DEFPROC LPRINT
2E+63 CALL LCTN(3E+63)
3E+63 CODE  E5 C5 CD 88 62 3B 3B E1 01 0
C 00 09 22 02 62 C1 E1 D5 F5 E5 C5 DD E5
 FD E5 DD 2A 02 62 01 8A 00 DD 09 FD 2A
02 62 01 80 00 FD 09 FD BE 01 30 58 FE 2
0 30 2F 6F FE 1F 20 13 FD 7E 00 FE 00 20
 0C 3E 0D 18 1F
4E+63 CODE  7D DD BE 00 20 07 FD 7E 00 F
E 00 28 2F DD 23 3E FF DD BE 00 20 EA 7D
 6F 01 80 00 ED 78 CB 77 28 1B DB 7C CB
47 28 F4 CB 4F 28 11 7D D3 7E FE 1B 20 0
6 FD 36 00 01 18 04 FD 36 00 00 FD E1 DD
 E1 C1 E1 F1 D1 C9 00 80
5E+63 CODE  1E FF
6E+63 ENDPROC
7E+63 PROC LPRINT
```

USING THE ROUTINE.

The routine is self locating, so to initialise the printer simply call it. If it is to be kept on tape to be appended it should be SAVEed with AUTO RUN, so when appended it will initialise itself. It must be re-initialised whenever the program underneath is edited. If loaded from the monitor a space of 8 bytes must be left where the line number and CODE statements in the middle of the routine fall.

The final byte of the second code line is used to test for graphics characters. The routine will not print a character with an ASCII code greater than this byte. Set to &80 it will mask out the Graphics, set it to &FF to print everything.

The last code line holds the table of control characters not to be printed except as part of an ESC sequence, the &FF is a terminating character. Thus to modify the routine to ignore control characters except when they follow an ESC simply insert them in the line before the &FF. It is set to ignore &1E.

J.S. Colombo.

A LOUDER LYNX.

A small modification can easily be made to any cassette recorder to obtain a much louder sound output from the LYNX. The simple circuit below (Fig.1a) shows a typical speaker and EAR socket arrangement. An additional 3.5mm jack is fitted and wired into the lead to the speaker the OPPOSITE way round to the EAR socket, (Fig.1b), thereby isolating the recorder speaker from the rest of the circuit (when a plug is in the new socket).

The sound ouput from the LYNX is taken from pins 2 and 4 on the cassette socket. A convenient method is to utilise the existing DIN plug on the end of the cassette lead and add a single screended miniature microphone cable into the plug. This plug is left permanently attached to the rear of the LYNX and this method saves wear and tear on both socket and cable by avoiding constant changing of plugs.

The polarity of the signal from pins 2 and 4 must be maintained, with pin 4 connected to the centre core of the microphone cable and the centre pin of the jack plug and so straight to the speaker. Pin 2 must be connected to the screening of the cable and so to the outer connection of the plug and chassis, and to the other side of the speaker. This is important because the other connections to the recorder do not have to be disconnected when the speaker is in use.

The recorder speaker, which is otherwise unused, now becomes the extension speaker for the LYNX. If the sound is ever too loud simply remove the plug from the recorder, leaving the LYNX speaker to function on its own as before. When the new socket is wired in correctly and without a plug fitted, the speaker functions normally for the recorder as the original signal path remains unchanged.

N.B. When SAVING DATA it is ESSENTIAL to REMOVE the speaker plug from the recorder in order to maintain correct signal levels.

M. GEORGE.

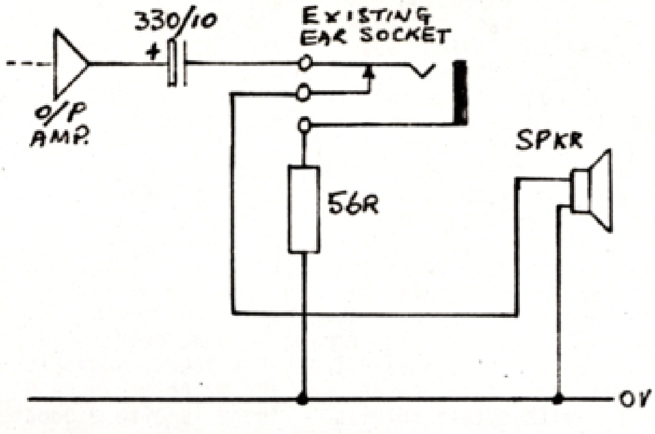


FIG. 1A

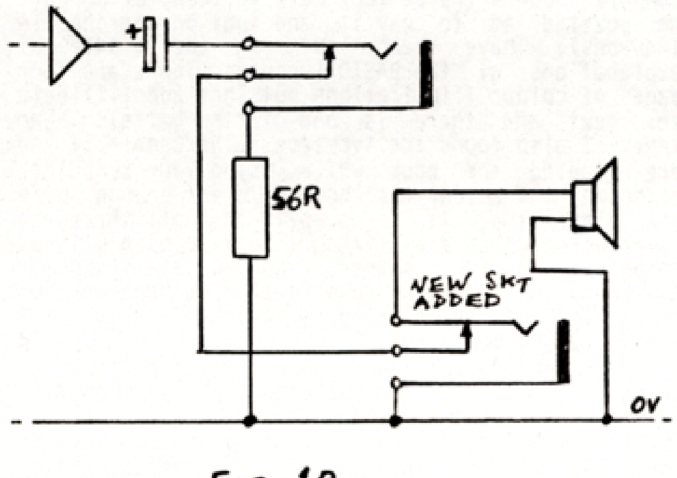


FIG. 1B

The 'LS138 TTL DECODER IC

This IC is commonly used to decode address lines to obtain various PORT addresses. It has a number of inputs and a range of outputs which can be used as particular PORTs. What this means is that with certain conditions set up on the inputs representing the particular PORT number, one of the outputs will go Active Low. The inputs are divided into two groups, namely, ENABLE and SELECT. A truth table follows and it will be noted that the ENABLE inputs can override the SELECT inputs, so these are generally left in a fixed state.

Data on this device is to be found in any standard TTL manual, so details of pin numbering etc., will not be given.

TRUTH TABLE

INPUTS						OUTPUTS							
ENABLE			SELECT										
G 1	G 2A	G 2B	C	B	A	Y 0	Y 1	Y 2	Y 3	Y 4	Y 5	Y 6	Y 7
x	H	L	x	x	x	H	H	H	H	H	H	H	H
x	L	H	x	x	x	H	H	H	H	H	H	H	H
x	H	H	x	x	x	H	H	H	H	H	H	H	H
L	x	x	x	x	x	H	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	L	L	H	H	H	H	L	H	H	H
H	L	L	H	L	H	H	H	H	H	H	L	H	H
H	L	L	H	H	L	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L

It can be readily seen from the above TRUTH TABLE that most of the inputs tend to follow positive logic whereas the outputs are inverted i.e. negative logic. This latter point is quite useful as many peripheral ICs tend to have an ENABLE or Chip Select pin which requires an active LOW condition to be activated. This is generally indicated by a bar (or line) above the pin's designation. In the case of the ENABLE inputs, these can also be treated as inputs for specific selection of addresses and would be used to prevent "reflections" in the selected ports. Of course any input can also be reversed in action by a simple inverter feeding it.

It can also be seen from the above that one 74LS138 can support 8 separate outputs, and by careful design, an economic system can be evolved, namely minimising the "chip count". A further consideration is that although one tends to think in a sequence of numbers, especially where address lines are concerned, these can be shuffled before feeding the inputs of this IC. Again it becomes obvious that by using a degree of "cunning", a multitude of PORTS can be obtained from a set of address lines, however some reflections would occur. So it is down to the designer to work out a sensible solution to their requirements.

To illustrate the above some simple examples are given:-

Using a form of short-hand, let X=G1(=1), x=G1inv.(=0), Y=G2Ainv.(=1), y=G2A(=0), Z=G2Binv.(=1), z=G2B(=0), A=A(=1), a=A(=0), B=B(=1), b=(=0), C=C(=1), and c=C(=0).

ADDRESS LINES 5	4	3	2	1	0	PORT No.
y	z	X	c	b	A	9
Y	z	X	c	b	A	29
Y	Z	X	c	b	A	39
Y	Z	X	c	B	a	3A
Y	Z	X	C	b	a	3C
Y	Z	X	C	B	B	3F

For clarity as the 74LS138 has only 6 inputs, only address lines 0-5 are shown, but of course there several higher address lines which would produce reflections. To extend the port decoding further, it would be perfectly feasible to use more '138s to drive each other and so obtain a minimum of repeated port numbers.

Summarising, it is obvious that this particular IC is very popular, and although ICs are increasing in complexity, in spite of its relative simplicity, this particular device still has a lot to offer.

R B JONES.

FAST MENU

This is a useful utility to provide a fast return to a menu using the ALTernative GREEN bank.

```
100 REM >>> FAST ACCESS MENU <<<
110 REM ### Number of options can be alt
ered in line 290 ###
120 REM ### Subroutines written to start
 at lines 1000,2000,3000 etc. ###
130 REM ### Ending with "RETURN" which w
ill put MENU back on the screen ###
140 LET V$=CHR$(18),R$=CHR$(31)+CHR$(10)
150 DPOKE &6292,&8000
160 CLS
170 PRINT  @ 30,5;V$;" MENU SELECTION ";
V$;R$;
180 PRINT 1;":FRED";R$;
190 PRINT 2;":DAVE";R$;
200 PRINT 3;":PHIL";R$;
210 PRINT 4;":BILL";R$;
220 PRINT 5;":HARRY";R$
230 PRINT 6;":ROY";R$;
240 PRINT 7;":ALF";R$;
250 PRINT 8;":BOBBY";R$;
260 PRINT  @ 35,180;V$;" SELECT OPTION "
;V$;
270 REPEAT
280    OUT &0080,16
290    REPEAT
300       LET X=GETN-48
310    UNTIL (X>0 AND X<9)
320    DPOKE &6292,&C000
330    OUT &0080,4
340    GOSUB X#1000
350    CLS
360 UNTIL FALSE
1000 REM ### Subroutine for option 1 ###
1010 REM
....
1040 REM ### Return at end of subroutine
 ###
1050 RETURN
```

The 3 'D's: DOS, DISKS AND DRIVES

Over the period of time I have been supplying members with Disk Drives, I have reached the opinion, that sadly, we are stuck with a relatively inferior Disk Operating System for the LYNX.

1) Disks. Possibly the main reason for my experience of failures with disks of many brands, can be firmly placed at the foot of the supplied drive, namely the ALPS FDD2124BC1 disk drive.
I have used many brands of disk, i.e. WABASH, 3Ms, DYSAN, CDC, NASHUA, and CENTECH; the best has been CENTECH with a close second, NASHUA. The worst has been CDC and WABASH, with the latter shedding oxide! The main problem with 3Ms, is the sleeve, which seems to be thicker than normal and tends to get jammed in the drive. This problem has also occured with a friend's BBC Cumana drives.
The reason why I mentioned the ALPS drive is that when formatting a new disk, to get the system to respond, I generally have to lightly place my finger on the pressure arm to tell the drive that a disk is present! This is of course due to a weak spring loading and varies enormously from drive to drive. Even CAMPUTER's engineers were aware of this problem.

2) DOS. As has now been determined from various sources, the LYNX was used as a 'test bed' for Intelligent Software's new DOS. Unfortunately it is still very rough and ready, and is now likely to remain so, as IS has now gone bust (due to the demise of the ENTERPRISE) and all source code is probably irretrievably lost. In fact the LYNX was used as a guinea pig for the ENTERPRISE! One major weakness is the FORMAT routine, as far as I can determine, the sequence is as follows:-
a) Carry out a READ operation, i.e. check to see whether there is magnetic noise emanating from the disk! The problem is of course that due to modern manufacturing methods, the residual noise level is extremely low, so the FORMAT routine keeps reading and reading and reading, ad nauseum! Use of the ESCape doesn't work and the only answer is to switch off or reset the LYNX! Try to format your disks BEFORE doing this, else loss of program. Ugh!
In fact with some disks, because the LYNX DOS can't detect them, I format on the 5 year old P2 and then re-format under LYNX DOS. I have even used a magnet to put spurious signals onto the disk. In fact comparing the way the FORMAT routine operates, it can't tell whether or not you've inserted a disk of cardboard!
b) After it decides that there is actually magnetic media inserted, it then checks whether it can move to the centre of the disk, what for, I ask?
c) After this it then looks for track 0, and if there is any tiny mis-alignment, it can sit there hunting. Cure, switch off and try another disk!
d) If it has got past this stage, writing to the disk commences, BUT with no verifying that the track has been written to! OK so it does a verify operation after the writing, not a problem with only a 40 track drive but when dealing with 160 tracks, this can be frustrating having to wait for all tracks being written to! If verifying was carried out track by track, one wouldn't have to wait for the entire sequence to be carried out before learning that it might be a duff disk.
Basically the FORMAT routine needs to be completely re-written with the above problems put right, and ALSO as is carried out with many other machines, to decide whether it is either a piece of cardboard or actually a magnetic disk, to carry out on track 0, a dummy write operation first of all with a read operation following.
There exists a juicy bug in the CHECK routine, if during checking side 2 (1Mb drives), a faulty track is detected, the inherent counter in the software doesn't start on side 2 at track 80, no, it starts from track 128 and counts happily up to track 207! Whoops!
Another untidy bit of software writing is the DIR command. Who in their right mind would want file names 30+ characters long. A directory similar to CP/M would be perfectly adequate, i.e. a maximum of 8 characters and displayed in four columns. As I have a mass of files on 1Mb drives, to have the directory writing and writing in one column is very frustrating.
Another weakness is the fact that the LYNX DOS cannot accept various drive capacities. The LYNX only can deal with two and even with these two, because of the software is very critical concerning the drive used. It is a very fussy system, however under CP/M (currently only 128K owners), the DOS seems to be more flexible. LYNX DOS is more critical towards 40 track drives and less so with 80 track drives. This indicates to me, that the software was written specifically for the ALPS drive and is therefore somewhat customised and as such is a b..... nuisance! One problem is that the ALPS drive is no longer available! Perhaps someone is prepared to correct the DOS.
These are perhaps the major weaknesses of LYNX DOS, maybe you know of some more?

3) Drives. Below is a list of drives which will work and also those which won't:-

Compatible
****

3"	MATSUSHITA EME-101.	250K.	
3.5"	MITSUBISHI.	1Mb.	
5.25"	ALPS FDD2124BC1.	250K.	Orig. drive.
5.25"	EPSON SD540.	1Mb.	
5.25"	TOSHIBA MKMO262A002.	1Mb.	(My drive!).
5.25"	MITSUBISHI. M4853-.	1Mb.	

Non-compatible
****

3.5"	EPSON SD110.	250K.	A puzzle here!
3.5"	SONY OA-D30V.	250K.	Non-Shugart.
5.25"	TANDON TM50.	250K.	No ready line.
5.25"	CUMANA (TEK 502).	250K.	No ready line.
5.25"	TEAC FD54.	250K.	

Two of the latter drives, the EPSON and TANDON nearly work but either software written on a TANDON TM50 can't be read by an ALPS (or vice versa) or the EPSON SD110 writes the tracks but can't verify during formatting. As a rough guide, if it works on a BBC, it probably won't work on the LYNX! However a drive which is set up to run on an IBM PC (motor speed 360 rpm), after the revs are slowed to 300rpm, stands a good chance of working under LYNX DOS, which has similarities to IBM System 34. The TOSHIBA was out of an IBM.

As you can see from my investigations, I have spent many futile hours trying to overcome the shortcomings of the LYNX disk operating system but with only partial success. One of the problems, certainly with modern drives, is the lack of technical information which might provide clues as to why only certain drives are suitable. Certainly of the companies I have been in contact with, EPSON are to date the most helpful, and TANDON being totally 'browned off' and thus un-helpful.

I have now built up quite a good library of drive data, including circuit diagrams and now intend to generate a good reference library on as many makes and models as I can.

As I previously mentioned, until someone can 'sort out' the DOS, the LYNX will remain a poor relation in the micro-world.

For a final footnote, it seems that all the micros which adopt a non-standard disk format, e.g. BBC, COMMODORE, ATARI, AMSTRAD have succeeded commercially, whereas those who have tried to follow some sort of industry standard have failed! Perhaps there's a message here?
R B JONES.

SCREEN DUMP

This routine will generate a hard copy of the green screen to a printer. This would be of use to those who may have produced pie charts or histograms together with text and might require a paper copy of a displayed display.

The code can be entered directly or via CODE lines in BASIC.

Program

```
6953              Org              ;Option using &33
                                   ;in place of a REM
6954  21BF61      LD HL,&61BF
6957  5E          LD E,(HL)        ;Save (&61BF) in E
6958  CBAE        RES 5,(HL)       ;Make sure that
695A  CBB6        RES 6,(HL)       ;code &1D and &1E
                                   ;can be passed to a
                                   ;printer
695C  1600        LD D,&0          ;Take first screen
                                   ;row
695E              SCR
695E  2A0062      LD HL,(&6200)
6961  3EC3        LD A,&C3
6963  320162      LD (&6201),A
6966  3E0D        LD A,&0D
6968  CD0162      CALL &6201       ;LPRINT
696B  3E1D        LD A,&1D         ;LPRINT CHR$(29);
696D  CD0162      CALL &6201       ;(PRINT 132 CHARS /
                                   ;LINE)
6970  220062      LD (&6200),HL
6973  0600        LD B,&0          ;Set no of blanks
                                   ;after last non-
                                   ;blank to zero
6975  D9          EXX              ;Switch to alter-
                                   ;native register set
6976  1E00        LD E,&0          ;Take first char in
                                   ;screenrow
6978              ROW
6978  0603        LD B,&3          ;Byte 8 bits wide.
                                   ;Char 2 bit wide.
697A              CBYTE
697A  DD21C661 LD IX,&61C6      ;&61C6 to &61C8 are
                                   ;used as workareas.
697E  0E00        LD C,&0          ;Take first char-row.
6980              CROW
6980  D9          EXX
6981  7A          LD A,D
6982  D9          EXX
6983  81          ADD A,C          ;Total-row to A
6984  6F          LD L,A
6985  2600        LD H,0
6987  1605        LD D,&5
6989              SH
6989  CB25        SLA L
698B  CB14        RL H
698D  15          DEC D
698E  20F9        JR NZ,SH         ;Total-Row*32 in HL
6990                               ;Screen Bank part!
6990  16C0        LD D,&C0
6992  19          ADD HL,DE        ;HL+&C000+Charcount
                                   ;to HL
6993  C5          PUSH BC
6994  CD7000      CALL &70         ;Byte at location HL
6997  C1          POP BC           ;in GREEN bank to L
6998  78          LD A,B
6999  FE00        CP &0
699B              WHILE
699B  2807        JR Z,WEND
699D  CB3D        SRL L
699F  CB3D        SRL L
69A1  3D          DEC A
69A2  18F7        JR WHILE         ;L DIV (B x 4) to L
69A4              WEND
69A4  3E03        LD A,&3
69A6  A5          AND L
69A7  F5          PUSH AF
69A8  E601        AND &01
69AA  17          RLA
69AB  57          LD D,A
69AC  F1          POP AF
69AD  E602        AND &02
69AF  1F          RRA
69B0  B2          OR D
69B1  DD7700      LD (IX+00),A     ;Store row C of char
```

```
69BF  1806        JR BUILD         ;later if char is
                                   ;non-blank
69C0                               ;Code needed to make
                                   ;relative jump longer
69C0              SCRT
69C0  189C        JR SCR
69C2              ROWT
69C2  18B4        JR ROW
69C4              CBYTET
69C4  18B4        JR CBYTE
69C6                               ;End of jump code.
69C6              BUILD
69C6  A7          AND A            ;Clear carry-bit
69C7  3AC861      LD A,(&61C8)
69CA  17          RLA
69CB  17          RLA
69CC  4F          LD C,A
69CD  3AC761      LD A,(&61C7)
69D0  B1          OR C
69D1  17          RLA
69D2  17          RLA
69D3  4F          LD C,A
69D4  3AC661      LD A,(&61C6)
69D7  B1          OR C             ;Charcode with B7
                                   ;reset in A
69D8  281D        JR Z,NXTCHR      ;Jump if char is
                                   ;blank
69DA  F680        OR 80            ;Set B7 of charcode
69DC  F5          PUSH AF          ;Save char-code
69DD                               ;Send all blanks
                                   ;which still must be
                                   ;printed to printer
69DD  2A0062      LD HL,(&6200)
69E0  3EC3        LD A,&C3
69E2  320162      LD (&6201),A
69E5  3E80        LD A,&80
69E7  05          DEC B            ;Correct no of
                                   ;blanks
69E8  2806        JR Z,WE
69EA              WH
69EA  CD0162      CALL &6201       ;LPRINT CHR$(&80);
69ED  05          DEC B
69EE  20FA        JR NZ,WH
69F0              WE
69F0  F1          POP AF
69F1  CD0162      CALL 6201        ;LPRINT CHR$(Char
                                   ;-code);
69F4  220062      LD (&6200),HL
69F7              NXTCHR
69F7  D9          EXX              ;Switch to alter-
                                   ;native register set
69F8  05          DEC B
69F9  3EFF        LD A,&FF
69FB  B8          CP B
69FC  20C6        JR NZ,CBYTET     ;Take next char in
                                   ;byte.
69FE  1C          INC E
69FF  3E20        LD A,&20
6A01  BB          CP E
6A02  20BE        JR NZ,ROWT       ;Take next char in
                                   ;screen row.
6A04  D9          EXX              ;Switch to normal
                                   ;register set
6A05  14          INC D
6A06  14          INC D
6A07  14          INC D
6A08  3EFF        LD A,&FF
6A0A  BA          CP D
6A0B  20B3        JR NZ,SCRT       ;Take next screen
                                   ;row
6A0D  3EC3        LD A,&C3
6A0F  320162      LD (&6201),A
6A12  3E0D        LD A,&0D
6A14  CD0162      CALL &6201
```

Program continues at the foot of page 8.

COMBAT by GORDON CLAY

****

This is a game for two players. It is in two parts and consists of a section of machine code to be entered first, followed by the BASIC program. Instructions to insert the machine code routine follows:-

The machine code is inserted into memory from location &9600 onwards, this must be saved to tape first, before typing in the BASIC program. The following instructions are DIRECT commands and are NOT a program.

After switching on your LYNX, type MON and hit the RETURN key. What will be displayed is the current state of the Z80 registers but unless you understand this ignore the display. Now type S <RETURN> to clear the screen. Unlike the BASIC prompt which starts with a ">" and cursor, you will have a new prompt "#" and cursor. Now type M9600 <RETURN>, this puts you into a MODIFY mode for direct writing into the RAM memory. One essential point is that you must insert a space between each pair of characters,(called a byte). So to start:-

```
ADDRESS:-             TYPE IN:-

9600                  F5 E5 D5 C5 D9 E5 D5 C5     <RETURN>
9608                  06 02 CD CE 00 D9 2A 54     <RETURN>
9610                  62 7D 6C 26 00 44 29 29     <RETURN>
9618                  29 29 29 0E FF 0C D6 03     <RETURN>
9620                  30 FB 09 01 00 C0 09 D9     <RETURN>
9628                  C5 46 23 4E 23 56 23 5E     <RETURN>
9630                  23 E5 D5 D9 01 20 00 E5     <RETURN>
9638                  09 E5 09 54 5D 09 EB D9     <RETURN>
9640                  E1 D1 D9 3E 17 01 FF FF     <RETURN>
9648                  ED 79 C1 3E 40 D3 80 D6     <RETURN>
9650                  20 D3 80 70 EB 71 D9 71     <RETURN>
9658                  EB 70 D6 20 D3 80 AF 01     <RETURN>
9660                  FF FF ED 79 E1 D9 11 20     <RETURN>
9668                  00 19 D9 C1 10 BA C1 D1     <RETURN>
9670                  E1 D9 C1 D1 E1 F1 C9 00     <RETURN>
9678                  00 00 00 00 00 00 00 00     <RETURN>
9680                  .                           <RETURN>
```

Now assuming you have typed the code in correctly, save it to tape, using the following command at the "#" prompt:-

#D 9600 9678 0 "M/C" , set the tape recorder to record and press <RETURN>.

This completes the first part of "COMBAT", now type J <RETURN> to put you back to the normal prompt ">" and cursor. Now you can type in the main BASIC program. After typing in the program, save to tape in the usual way with the title "COMBAT".

```
100 PROC GRAPHICS
110 PROC INTRO
120 CLS
130 DPOKE &620D,&6608
140 WINDOW 0,93,1,251
150 PROTECT 0
160 LET O=0,P=0
170 CLS
180 INK BLUE
190 FOR A=1 TO 100
200   DOT RAND(255),RAND(255)
210 NEXT A
220 PROTECT BLUE
230 DPOKE &62B9,&9600
240 CLS
250 POKE &9600+&0044,&0017
260 PRINT  @ 5,0;"RED :"; @ 44,0;"YELLOW
 :";
270 IF O=0 THEN  GOTO 320
280 FOR o=1 TO O
290   POKE &9600+&0044,&0017
300   PRINT  @ 15+(o#3),0;CHR$(132);
310 NEXT o
320 IF P=0 THEN  GOTO 370
330 FOR p=1 TO P
340   POKE &9600+&0044,&0013
350   PRINT  @ 65+(p#3),0;CHR$(128);
360 NEXT p
370 IF O=5 THEN  PROC PLAYER 1
380 IF P=5 THEN  PROC PLAYER 2
390 LET A=60,B=50,X=30,Y=200
400 LET A$=CHR$(128),B$=CHR$(132)
410 PROC MOVE SECOND
420 IF X>87 THEN  LET X=87
430 IF X<6 THEN  LET X=6
440 IF Y>220 THEN  LET Y=220
450 IF Y<40 THEN  LET Y=40
460 POKE &9600+&0044,&0017
470 PRINT  @ X,Y;B$; @ X-3,Y-10;"     ";
@ X-3,Y+10;"     "; @ X-3,Y;" "; @ X+3,Y;
```

```
1000 NEXT q
1010 PRINT  @ A,B;"  "; @ X,Y;"  ";
1020 GOTO 240
1030 IF A$=CHR$(128) THEN  PROC SHOOT UP
1040 IF A$=CHR$(129) THEN  PROC SHOOT RI
GHT
1050 IF A$=CHR$(130) THEN  PROC SHOOT DO
WN
1060 IF A$=CHR$(131) THEN  PROC SHOOT LE
FT
1070 GOTO 840
1080 IF B$=CHR$(132) THEN  PROC SHOOT UP
1090 IF B$=CHR$(133) THEN  PROC SHOOT RI
GHT
1100 IF B$=CHR$(134) THEN  PROC SHOOT DO
WN
1110 IF B$=CHR$(135) THEN  PROC SHOOT LE
FT
1120 GOTO 920
1130 DEFPROC SHOOT UP
1140 LET b=b-20
1150 PRINT  @ a,b;CHR$(136);
1160 PRINT  @ a,b+5;" ";
1170 SOUND 183,1
1180 IF b<15 THEN  PRINT  @ a,b;" ";
1190 IF b<15 THEN  ENDPROC
1200 LET b=b-10
1210 PROC HIT
1220 GOTO 1150
1230 DEFPROC HIT
1240 IF s=0 THEN  GOTO 1550
1250 IF s=1 THEN  GOTO 1580
1260 ENDPROC
1270 DEFPROC SHOOT RIGHT
1280 LET a=a+3
1290 PRINT  @ a,b;" ";CHR$(138);
1300 IF a>85 THEN  PRINT  @ a,b;"  ";
1310 IF a>85 THEN  ENDPROC
1320 SOUND 183,3
1330 LET a=a+3
```

```
" ";
480 PROC MOVE FIRST
490 IF A>87 THEN  LET A=87
500 IF A<6 THEN  LET A=6
510 IF B>220 THEN  LET B=220
520 IF B<40 THEN  LET B=40
530 POKE &9600+&0044,&0013
540 PRINT  @ A,B;A$; @ A-3,B-10;"     ";
@ A-3,B+10;"     "; @ A-3,B;" "; @ A+3,B;
" ";
550 IF A=X AND B=Y OR A=X+1 AND B=Y OR A
=X-1 AND B=Y THEN  PROC COL
560 GOTO 410
570 DEFPROC GRAPHICS
580 RESERVE HIMEM-120
590 DPOKE GRAPHIC,HIMEM
600 FOR L=0 TO 119
610   READ K
620   POKE LETTER(128)+L,K
630 NEXT L
640 DATA &18,&3C,&7E,&18,&18,&7E,&5A,&42
,&0,&0
650 DATA &0,&E4,&26,&7F,&7F,&26,&E4,&0,&
0,&0
660 DATA &42,&5A,&7E,&18,&7E,&3C,&18,&0,
&0,&0
670 DATA &0,&27,&64,&FE,&FE,&64,&27,&0,&
0,&0
680 DATA &18,&3C,&18,&18,&DB,&FF,&DB,&42
,&0,&0
690 DATA &70,&F0,&22,&7F,&7F,&22,&F0,&70
,&0,&0
700 DATA &42,&DB,&FF,&DB,&18,&18,&3C,&18
,&0,&0
710 DATA &0E,&0F,&44,&FE,&FE,&44,&0F,&0E
,&0,&0
720 DATA &18,&0,&0,&0,&0,&0,&0,&0,&0,&0
730 DATA &0,&0,&0,&0,&0,&0,&0,&18,&0,&0
740 DATA &0,&0,&0,&80,&80,&0,&0,&0,&0,&
0
750 DATA &0,&0,&0,&01,&01,&0,&0,&0,&0,&
0
760 ENDPROC
770 DEFPROC MOVE FIRST
780 IF INP(&0080)=239 THEN  LET A$=CHR$(
128),B=B-10
790 LET s=0,a=A,b=B
800 IF INP(&0280)=239 THEN  LET A$=CHR$(
129),A=A+3
810 IF INP(&0080)=223 THEN  LET A$=CHR$(
130),B=B+10
820 IF INP(&0280)=223 THEN  LET A$=CHR$(
131),A=A-3
830 IF INP(&0380)=223 THEN  GOTO 1030
840 ENDPROC
850 DEFPROC MOVE SECOND
860 LET s=1,a=X,b=Y
870 IF INP(&0880)=223 THEN  LET B$=CHR$(
132),Y=Y-10
880 IF INP(&0780)=223 THEN  LET B$=CHR$(
134),Y=Y+10
890 IF INP(&0980)=251 THEN  LET B$=CHR$(
135),X=X-3
900 IF INP(&0980)=223 THEN  LET B$=CHR$(
133),X=X+3
910 IF INP(&0680)=223 THEN  GOTO 1080
920 ENDPROC
930 DEFPROC COL
940 PRINT  @ A,B;" ";
950 LET b=1
960 FOR q=1 TO 10
970   PRINT  @ A,B;CHR$(150+b);
980   LET b=b+1
990   SOUND 1836,RAND(40)+20
```

```
1340 PROC HIT
1350 GOTO 1290
1360 DEFPROC SHOOT DOWN
1370 LET b=b+20
1380 PRINT  @ a,b;CHR$(137);
1390 PRINT  @ a,b-5;" ";
1400 IF b>230 THEN  PRINT  @ a,b;" ";
1410 IF b>230 THEN  ENDPROC
1420 SOUND 183,4
1430 LET b=b+10
1440 PROC HIT
1450 GOTO 1380
1460 DEFPROC SHOOT LEFT
1470 LET a=a-6
1480 PRINT  @ a,b;CHR$(138);" ";
1490 IF a<6 THEN  PRINT  @ a,b;" ";
1500 IF a<6 THEN  ENDPROC
1510 SOUND 183,1
1520 LET a=a-3
1530 PROC HIT
1540 GOTO 1480
1550 IF a=X AND b=Y OR a=X+1 AND b=Y OR a
=X+2 AND b=Y THEN  PROC KILL SECOND
1560 IF a=X AND b=Y OR a=X AND b=Y+1 OR a
=X AND b=Y+2 OR a=X AND b=Y+5 OR a=X AND
b=Y+6 THEN  PROC KILL SECOND
1570 GOTO 1260
1580 IF a=A AND b=B OR a=A+1 AND b=B OR a
=A+2 AND b=B THEN  PROC KILL FIRST
1590 IF a=A AND b=B OR a=A AND b=B+1 OR a
=A AND b=B+2 OR a=A AND b=B+5 OR a=A AND
b=B+6 THEN  PROC KILL FIRST
1600 GOTO 1260
1610 DEFPROC KILL SECOND
1620 POKE &9600+&0044,&0017
1630 LET R=0
1640 PRINT  @ X-3,Y;"   "; @ X-3,Y+10;"
 "; @ X-3,Y-10;"   ";
1650 REPEAT
1660   LET R=R+1
1670   PRINT  @ X,Y;CHR$(132);
1680   SOUND 1836,10
1690   PRINT  @ X,Y;CHR$(133);
1700   SOUND 1836,30
1710   PRINT  @ X,Y;CHR$(134);
1720   SOUND 1836,20
1730   PRINT  @ X,Y;CHR$(135);
1740   SOUND 1836,4
1750 UNTIL R=10
1760 VDU 20
1770 LET O=O+1
1780 PRINT  @ A,B;"  "; @ X,Y;"  ";
1790 GOTO 1240
1800 ENDPROC
1810 DEFPROC KILL FIRST
1820 POKE &9600+&0044,&0013
1830 PRINT  @ A-3,B;"   "; @ A-3,B+10;"
 "; @ A-3,B-10;"   ";
1840 LET R=0
1850 IF R>10 THEN  GOTO 1970
1860 LET R=R+1
1870 PRINT  @ A,B;CHR$(128);
1880 SOUND 1836,4
1890 PRINT  @ A,B;CHR$(129);
1900 SOUND 1836,20
1910 PRINT  @ A,B;CHR$(130);
1920 SOUND 1836,9
1930 PRINT  @ A,B;CHR$(131);
1940 SOUND 1836,22
1950 SOUND 1836,5
1960 GOTO 1850
1970 PRINT  @ A,B;"  "; @ X,Y;"  ";
1980 LET P=P+1
1990 GOTO 240
```

```
69B4  DD23      INC IX          ;Take next work area
69B6  0C        INC C
69B7  3E03      LD A,&03
69B9  B9        CP C
69BA  20C4      JR NZ,CROW      ;Take next char-row
69BC  B9        EXX             ;Switch to normal
                                ;register set
69BD  04        INC B           ;B will be set back
```

M Perdeck.

```
6A17  3E1E      LD A,&1E
6A19  CD0162    CALL &6201      ;Set printer back to
                                ;80 chars/line.
6A1C  220062    LD (&6200),HL
6A1F  7B        LD A,E
6A20  32BF61    LD (&61BF),A ;Restore contents of
                             ;&61BF.
6A23  C9        RET
```

Lynx Word Processor (cass.)

I responded to the advertisement for a Lynx Word Processor in the last edition of LUG by sending an SAE for further details, and received a manual by return of post. I appreciated this service, short of actually getting your hands on a piece of software to play with, having the manual to read is the best way of evaluating whether it is likely to be worth buying. I may add that it did not take me long to make up my mind to order the program cassette as well, and that I received the cassette just about as quickly.

I had previously acquired the Liontext Word Processor (reviewed by N A Holding in the last issue of LUG magazine). I will say from the outset that the Lynx Word Processor developed by R B Poate at £16.50 represents far better value for money that Liontext, even at £14.95.
Ed. Please see end of review.

The first feature to mention about "LWP", is that it gives you full screen editing. You can use the four cursor control keys to move around the screen in any way you like and overtype or (otherwise correct by insertion or deletion) whatever is on the screen. You are not restricted to the current paragraph or even the current screen, but can move to any part of the text.

The second thing to mention is that although, like Liontext, "LWP" does not scroll the screen, so that when you reach the bottom of the screen, you start again from the top, when this happens in "LWP", the screen is automatically cleared for you, so you do not find yourself typing over what was already there. Even better, "LWP" displays the last couple of lines of the screen you have just completed at the top of the new screen, so that you are not left wondering where on earth you had just got to when the screen just clears.
If you use the cursor control keys to move off the top or bottom of the current screen page, the screen clears and displays the previous or next page in a similar manner. Re-writing of a screen is fairly rapid, so it is practicable to page through a number of screens to find a particular place, although this could naturally become a lengthy process in a long document.

The third area where "LWP" scores heavily over Liontext is in the insertion, deletion and amendment of text. Not only can you move to any part of the text with relative ease in "LWP", but you can amend the text at any point without having to wait an age for the affect to ripple through to the end of text. This is achieved by the use of "nulls". When a character is deleted, it is replaced by a "null" (shown on the screen as a little squiggle { TILDE }, so that no shuffling of text need take place. "LWP" also has true on screen wordwrap, which leaves a generous supply of "nulls" padding out at the end of lines so that insertion of new text involves shuffling existing text only as far as the next group of nulls, not right down to the end of the text file. This makes the insertion of space for new text very rapid indeed (far faster than the BASIC line editor for example). Even better still, there are commands for the insertion of a whole linefull or screenfull of nulls (which you can then overtype with your inserted text), which also seem to work very rapidly. "LWP" does not distinguish between a text and a command mode. Commands are entered by using ESCape key sequences e.g.ESC P for print or ESC D for down page. The control key is used for placing printer control characters into the text; CTRL A =ASCII 01 etc.

When it comes to printing,the text must first be formatted using the ESC G (generate print file) command. This inserts carriage returns in the text at the end of each print line, and pads out lines with extra spaces to produce right justification if required. The purpose of formatting before rather than during print is to allow flexibility. By putting various markers in the text at various points, you can select right justification, ragged right margin, or no formatting at all. The last option allows you to format text, e.g. in a table, manually so that your columns do not get shifted around by attempts at right justification. It does however have some disadvantages however. The first is that on a sizeable document this formatting process can take a disconcertingly long time, so that you might be tempted to think that the program had hung up on you, there is an asterisk which blinks at the bottom of the screen to re-assure you that something is indeed still going on, unfortunately it sometimes remains on or off for rather a long time, which tends to defeat the object. The second is that having printed a draft, although it is easy enough to make minor alterations to the formatted file, such as adding or deleting a single letter in a line, anything more than this, such as adding an extra couple of words, could be more problematic. You would either have to adjust all the line lengths by hand after the insertion, or else reverse the formatting process (a command is provided for this),make your correction, and then re-format the file.The danger is that this will seem like too much bother and you will be tempted to sacrifice the quality of your prose to the convenience of your word processor operation.

The printed line length may be set to between 30 and 250 characters, but there is no means of setting a page length. You can print a selected portion of text by inserting an end of print marker, and you could split your document into pages by inserting several end of print markers where you wanted the page breaks to come. My own solution is to work out from the printed draft where I want the page breaks and to insert form feed printer control characters at appropriate places in the text.

"LWP" includes the normal block move and delete, and search and replace functions. The block move is in fact a block copy, although the text copied can then be deleted by issuing ESC K directly after the move command. This works rapidly on small blocks of text, but I did find the delete command took an alarmingly long time when I tried it on a large 'chunk' (i.e. a couple of printed pages). I am not sure I am entirely happy with the way the find and replace commands have been implemented in "LWP". The find command underlines every occurence of the search string within the text,and you then have to page through the text to find each occurence of underlining (it is no problem to remove the underlining again). I would have preferred a command which paused at each occurence of the string being sought, underlining it and displaying the page where it was found with an option to continue searching or to stop searching and edit that screen page, which would then provide a rapid method of accessing a particular part of a large document. I have a similar reservation about the replace option, which replaces every occurence of the search string with the replacement string, underlining the latter to show what has been done. I would have preferred to see "LWP" display each occurence of the search string and give a "REPLACE (Y/N)" option each time. For example, if

I discover that I have committed my favourite typing error of typing "their" instead of "there", I will probably want to replace some, but not every, occurence of "their" with "there". In my case these are very minor grouses, since in practice I tend to make little use of these functions,even when word processing using Silicon Office on an IBM PC-XT at work.

Having aired my minor reservations, I shall conclude with more good news. "LWP" comes on a cassette (at TAPE 0), which does not auto-load. This means that you can make a backup copy, and better still, make a backup copy which tailors "LWP" to your own requirements,eg. your best tape speed, 48 or 96K, printer features, autoloading "LWP" if you want to, you can even change the graphic characters used for the cursor etc., the manual explains where they are located.

It has not been possible to cover every feature of "LWP" in this review, I have said nothing about tape-handling for example. In spite of the odd reservations I have expressed, "LWP" seems to me to be a good implementation of a tape-based word processor on the LYNX, which overcomes the limitations of the LYNX's screen handling very well. It is certainly a far more useable product than Liontext; I am composing this review using "LWP" and used it to write the final third of a chapter on the LYNX DATA STORE for the ADVANCED MANUAL (Ed. This is now ready for the printers.). I doubt if I would have attempted the former task using Liontext and I would not have used it for the latter. Given the limitations of a 40-column non-scrolling screen for word processing, if you want to use your LYNX for inexpensive word processing, then I can heartily recommend "Lynx Word Processor".
E. Eve.

LWP Update
****

Since the article above was first submitted, there have been important changes relating to "LWP".

1) All rights and copyright have now been transferred to myself (R.B.JONES) and also the rights to NILUG magazine. For those who have missed these magazines,I will be producing a composite version,known as the "NILUG VOLUME".
2) "LWP" is now disk-based and the points mentioned in the article about its deficiencies have been noted.
3) It is also intended to enhance "LWP" even further, to provide auto-saving and loading when a given document file becomes too large to fit in the available RAM space. This latter feature becomes only practicable with disk operation.
4) A further feature will be to provide amending and generation of adjacent files. For example, it may be necessary to repeat a section of text throughout a file, so by using the block defining routine, this could be used to generate a sub-file, which could then be used as many times as required throughout the host file but is only practical via disk operation.

These further features may not be available initially, but return of the original software will of course be up-dated.

It is intended to make "LWP" the definitive word processor for LYNX owners who don't possess CP/M but would still require a high quality word processing program.
R.B.JONES.

TANDY PATTERNS

This is a short program for those who use the TANDY CGP115 printer/plotter. It is not suitable for a conventional printer without the four colour pens. It may require the printer patch provided in Issue one of the magazine.

```
100 CLS
110 VDU 24,1,2
115 LPRINT CHR$(18)
120 PRINT @ 32,60;"STOP THE TAPE"
130 PAUSE 50000
140 VDU 25
150 PROC intro
160 REPEAT
170   PROC box
180   PRINT @ 2,235;"Press S to stop, s
pace bar to run again";
190   LET I$=GET$
200 UNTIL "S"=UPC$(I$)
210 CLS
220 PRINT  @ 2,235;"START THE TAPE"
230 LOAD "ENLARGE"
240 END
250 DEFPROC box
260 RANDOM
270 LET n=0
280 CLS
290 LET Q=RAND(8)+1,W=RAND(8)+1,x=0,y=0,
A=0,B=0
300 WHILE n<3
310   LET X=254-x,Y=247-y
320   IF x>X THEN  LET A=X,B=x
330   ELSE  LET A=x,B=X
340   IF y>Y THEN   LET C=Y,D=y
350   ELSE  LET C=y,D=Y
360   DOT A,C
365   LPRINT "M";INT(A*2/1.06);",";C*2
370   DRAW B,C
375   LPRINT "D";INT(B*2/1.06);",";C*2
380   DRAW B,D
385   LPRINT "D";INT(B*2/1.06);",";D*2
390   DRAW A,D
395   LPRINT "D";INT(A*2/1.06);",";D*2
400   DRAW A,C
405   LPRINT "D";INT(A*2/1.06);",";C*2
410   INK RAND(7)+1
415   LPRINT "C";RAND(4)
420   LET x=x+Q,y=y+W
430   IF y+W>=247 THEN  GOTO  LABEL 1
440   IF x+Q>=254 THEN  GOTO  LABEL 2
450   GOTO  LABEL 3
460   LABEL 4
470   LET n=n+1
480   LABEL 3
490 WEND
500 GOTO  LABEL 5
510 LABEL 1
520 LET y=Y
530 GOTO  LABEL 4
540 LABEL 2
550 LET x=X
560 GOTO  LABEL 4
570 LABEL 5
580 PAUSE 10000
590 ENDPROC
600 DEFPROC intro
610 CLS
620 INK RED
630 PRINT  @ 3,30;"The following program
 draws patterns"
640 PRINT "made from concentric rectangl
es in "
650 PRINT "random colours."
660 VDU 1,7,10,10
670 PRINT "Press any key to continue."
680 LET x=GETN
690 ENDPROC
```

Frank Di Mambro

INPUT ROUTINE

****

ADVANTAGES:- Allows a null input (empty string).
Limits the maximum number of characters that can be input by the user and prevents the corruption of screen formats.
Prevents the accidental escape from the program as could happen if GET or KEY was used to emulate this routine in BASIC.(The [ESC] key is too near the [1] key). [ESC] and [RETURN] must be used together.

DIS-ADVANTAGES:- A steady underline cursor is used to simplify the routine.
Single key entries ([ESC] + letter) or graphics mode ([CONTROL] + [1]) are not disabled.
The only editing feature is the [DELETE] key.

Demo - Program

```
    100 CODE  5D 2A 48 69 16 00 3E 15 CF 3E
    5F CF CD 2F 20 B7 28 FA 47 0E 00 FE 20 3
    8 0D FE 7B 30 09 7A BB 28 11 14 23 70 18
     08 FE 08 20 08 7A B7 28 03 15 28 48 3E
    14 CF 3E 08 CF 79 CF 78 FE 0D 20 C7 23 3
    6 0D C9
    110 DIM A$(30),Z$(30)
    120 LET Y=5,Z=LCTN(100)
    130 LET A$="Hello There"
    140 CLS
    150 PRINT  @ 3,Y;"Input: "A$;  @ 24,Y;
    160 CALL Z,30
    170 IF NOTZ$="" THEN  LET A$=Z$
    180 PRINT  @ 24,Y;A$CHR$(30);
    190 LET Y=Y+10
    200 GOTO 150

    110 DIM A$(30),S$(30),Z$(30)
    130        LET       A$="HELLO       THERE",S$="
    "
    150 PRINT  @ 3,Y;"Input: "A$; @ 117,Y;"‡
    ‡"; @ 24,Y;
    180 PRINT  @ 24,Y;A$LEFT$(S$,30-LEN(A$))
    ;
```

Assembly Routine:

```
 0  LD E,L                 36  JR (Line 48)
 1  LD HL,(&6948)          38  CP &08
 4  LD D,0                 40  JR NZ,(Line 49)
 6  LD A,&15               42  LD A,D
 8  RST &08                43  OR A
 9  LD A,&5F               44  JR Z,(Line 49)
11  RST &08                46  DEC D
12  CALL &202F             47  DEC HL
15  OR A                   48  LD C,B
16  JR Z,(Line 12)         49  LD A,&14
18  LD B,A                 51  RST &08
19  LD C,0                 52  LD A,&08
21  CP &7B                 54  RST &08
23  JR C,(Line 38)         55  LD A,C
25  CP &7B                 56  RST &08
27  JR NC,(Line 38)        57  LD A,B
29  LD A,D                 58  CP &0D
30  CP E                   60  JR NZ,(Line 6)
31  JR Z,(Line 49)         62  INC HL
33  INC D                  63  LD (HL),&0D
34  INC HL                 65  RET
35  LD (HL),B
```

IN USE: This routine can be used to accept input from a blank field or over the top of existing data. The user's input is placed in Z$ which must have previously been dimensioned. The maximum size of the input field must be specified as the argument to the CALL statement.The example program demonstrates the routine being used to amend data in A$. A null input implies, leave the data in A$ as it was.

Note that A$ is printed after input and this must not be omitted even if a null response is detected. The routine blanks out the first character but even if it didn't, the user could overtype with rubbish and delete to the end of line. A null input would be detected implying no change but the screen would obviously show differently. This is a fault which is still found in modern expensive software.

Should the clear to end of line character (30) be inappropriate, because of other input fields or screen display on the same line, the resultant field must be padded out with spaces. Program amendments are therefore given. S$ is a string of 30 spaces - the maximum length of the input string.

K R COOPER.

LYNX MBASIC FUNCTIONS & COMMANDS

============================

As there is no documentation on MBASIC as implemented on the 128K LYNX, to run under CP/M, what follows is purely a list. At the time of compiling this list, the tokenising technique had not been resolved, so de-limiters and quotes for example have yet to be discovered.

```
ABS          GOTO         POKE
AND          GO TO        POS
ASC          .            PRINT
ATN          HEX$         PUT
AUTO         .            .
.            IF           RANDOMIZE
CALL         IMP          READ
CDBL         INKEY$       REM
CHAIN        INP          RENUM
CHR$         INPUT        RESET
CINT         INSTR        RESUME
CLEAR        INT          RESTORE
CLOSE        .            RETURN
COMMON       KILL         RIGHT$
CONT         .            RND
COS          LEFT$        RSET
CSNG         LEN          RUN
CVI          LET          .
CVS          LINE         SAVE
CVG          LIST         SGN
.            LLIST        SIN
DATA         LOAD         SPACE$
DEF          LOC          SPC(
DEFDBL       LOF          SQR
DEFNT        LOG          STEP
DEFSNG       LPOS         STOP
DEFSTR       LPRINT       STRING$
DELETE       LSET         STR$
DIM          .            SWAP
.            MERGE        SYSTEM
EDIT         MID$         .
ELSE         MKD$         TAB(
END          MKI$         TAN
EOF          MKS$         THEN
EQV          MOD          TO
ERASE        .            TROFF
ERL          NAME         TRON
ERR          NEW          .
ERROR        NEXT         USING
EXP          NOT          USR
.            NULL         .
FIELD        .            VAL
FILES        OCT$         VARPT$
FIX          ON           .
FN           OPEN         WAIT
FOR          OPTION       WEND
FRE          OR           WHILE
.            OUT          WIDTH
GET          .            WRITE
GOSUB        PEEK         XOR
```

NB. The full stop (.) represents a "boundary" byte in the command table.

Perhaps someone else could either suggest a manual/book which covers(?) MBASIC or better still, write an article on it covering it in more detail. I do have MBASIC for the ALPHATRONIC and have used it but there are variations between each implementation of the language so what I have would only be a rough guide to that available for the LYNX.

R B JONES.

VID-RAM STORE

This routine can be used to store large arrays of data, in the video RAM. It is only suitable for the 48/96K LYNXes.

BASIC PROGRAM

```
1 CODE  7C FE 00 20 11 7D E6 E0 20 0C 65
 AF CB 14 2E 00 EB 21 00 A0 19 C9 F1 3E
0D C3 88 62
2 CODE  CD 54 69 DD 2A 11 62 DD 7E 07 FE
 00 20 24 DD 7E 06 FE 64 20 1D DD 56 05
DD 5E 04 01 F9 01 E5 C5 D5 CD 69 00 D1 C
1 7D E1 12 23 13 0B 78 B1 20 EE C9 3E 0E
 C3 88 62
3 CODE  CD 54 69 EB DD 2A 11 62 DD 66 05
 DD 6E 04 DD 46 07 DD 4E 06 78 FE 00 28
05 3E 0E C3 88 62 79 FE 64 28 05 3E 0E C
3 88 62 01 F9 01 7E D5 D9 E1 01 00 00 57
 3E 20 08 3E 23 1E 00 CD B6 08 08 D9 23
13 0B 78 B1 20 E5 C9
4 DPOKE LCTN(2)+1,LCTN(1)
5 DPOKE LCTN(3)+1,LCTN(1)
100 TEXT
110 DIM A(100)
120 FOR M=0 TO 31
130   FOR N=0 TO 100
140     LET A(N)=M
150   NEXT N
160   CALL LCTN(3),M
170 NEXT M
180 RUN 190
190 DIM (A(100)
200 FOR M=0 TO 31
210   CALL LCTN(2),M
220   FOR N=0 TO 100
230     PRINT A(N);"    ";
240   NEXT N
250   OUT &0080,4
260   LET A=GETN
270   CLS
280 NEXT M
```

Machine code Analysis

```
Add. Code      Mnemonics      Comment.
               ORG &9600      ;For 48K machine.
               LOAD &9600
     ERRAM:    EQU &6288      ;System calls
9600 181E      JR VFETCH      ;So that call
9602 184D      JR VSTORE      ;addresses are easier
                              ;to remember.
9604           GETADD         ;Find addr. in video
9604 7C        LD A,H         ;Check for offset.
9605 FE00      CP 0           ;Too large
9607 2011      JR NX,FAIL     ;If so then error.
9609 7D        LD A,L         ;Test low byte for
960A E6E0      AND &00E0      ;too large
960C 200C      JR NZ,FAIL     ;If so then error
                              ;again
960E 65        LD H,L         ;Multiply HL by 512
960F AF        XOR A          ;decimal to get
9610 CB14      RL H           ;offset from
9612 2E00      LD L,0         ;start of screen.
9614 EB        EX DE,HL
9615 2100A0    LD HL,&0A00    ;Add &A000 to get
9618 19        ADD HL,DE      ;absolute address.
9619 C9        RET            ;Done.
961A           FAIL
961A F1        POP AF         ;Clear up stack
961B 3E0D      LD A,13        ;Jump to Number out of
961D C38862    JP ERRAM       ;Range error.
9620           VFETCH
9620 CD0496    CALL GETADD
9623 DD2A1162 LD IX,(ATBL)  ;Get pointer to ATBLs.
9627 DD7E07    LD A,(IX+7)    ;Get ms. byte of
                              ;length
962A FE00      CP 0           ;If not zero then fail
962C 203C      JR NZ,FAIL1    ;with error.
962E DD7E06    LD A,(IX+6)    ;Low byte of length
9631 FE64      CP 100         ;Is it 100 decimal? If
9633 2035      JR NZ,FAIL1    ;not then fail with
                              ;error
9635 DD5605    LD D,(IX+5)    ;If OK get addresses
9638 DD5E04    LD E,(IX+4)    ;of start of array.
963B           LSTRT
963B 01F901    LD BC,505      ;No of bytes to
                              ;transfer.
963E           LOOP
963E E5        PUSH HL        ;Save
963F C5        PUSH BC        ;the
9640 D5        PUSH DE        ;environment.
9641 CD6900    CALL INBLUE    ;Fetch byte
9644 D1        POP DE         ;Restore part
9645 C1        POP BC         ;of environment
9646 7D        LD A,L         ;Put data byte into a
                              ;register.
9647 E1        POP HL         ;Complete environment.
```

```
Add. Code      Mnemonics      Comment.
9648 12        LD (DE),A      ;Save data
9649 23        INC HL         ;Bump
964A 13        INC DE         ;pointers
964B 0B        DEC BC         ;Dec count
964C 78        LD A,B         ;Is count
964D B1        OR C           ;zero
964E 20EE      JR NZ,LOOP     ;if not then repeat.
9650 C9        RET            ;Finish.
9651           VSTORE
9651 CD0496    CALL GETADD
9654 EB        EX DE,HL       ;Video is destination
9655 DD2A1162 LD IX,(ATBL)  ;now see similar
9659 DD6605    LD H,(IX+5)    ;section above.
965C DD6E04    LD L,(IX+4)    ;-Scope for
965F DD4607    LD B,(IX+7)    ;further
9662 DD4E06    LD C,(IX+6)    ;compression
9665 78        LD A,B         ;here!
9666 FE00      CP 0           ;Sorry-lack
9669 2805      JR Z,OK1       ;of time!
966A           FAIL1
966A 3E0E      LD A,14        ;Subscript out of
                              ;range.
966C C38862    JP ERRAM       ;Make error jump
966F           OK1
966F 79        LD A,C         ;More duplication
9670 FE64      CP 100         ;see above again.
9672 2805      JR Z,OK2
9674 3E0E      LD A,14
9676 C38862    JP ERRAM
9679           OK2
9679 01F901    LD BC,505      ;Byte count again!
967C           LOOPO
967C 7E        LD A,(HL)      ;Byte to store
967D D5        PUSH DE        ;Save dest. address.
967E D9        EXX
967F E1        POP HL         ;Fetch into HL'
9680 010000    LD BC,0        ;Set up registers.
9683 57        LD D,A         ;for late
9684 3E20      LD A,&20       ;entry into
9686 08        EX AF,AF       ;OUTBYTE.
9687 3E23      LD A,&23       ;See text
9689 1E00      LD E,0
968B CDB608    CALL &08B6     ;Late entry to OUTBYTE.
968E 08        EX AF,AF       ;Restore
968F D9        EXX            ;environment
9690 23        INC HL         ;Bump
9691 13        INC DE         ;pointers,
9692 0B        DEC BC         ;decrement count
9693 78        LD A,B         ;and test
9694 B1        OR C           ;for finish
9695 20E5      JR NZ,LOOPO    ;If not done repeat
9697 C9        RET            ;else finish
```

S.Roberts, (Formerly of Camputers)
Ed. Please note this routine may be untried, so save before running it!

REVIEW OF TIMESTEP ELECTRONICS WEATHER SATELLITE SYSTEM

At last - a really good, practical use of the LYNX's high resolution screen capabilities.

TIMESTEP ELECTRONICS have been marketing a weather satellite receiving system for the BBC micro for some time now and it certainly produces very good results. However, the LYNX screen being somewhat superior should produce better weather pictures and indeed it does!

TIMESTEP market their products either as kits, built modules or built and boxed units. The cheapest are of course the kits, but you MUST know both ends of a soldering iron before even thinking of building one. The radio receiver will require some test gear to align it, so if you don't possess this facility, DON'T build a kit, buy it ready built. I built the kits and excellent they are too.

The basic system needs a chimney mounted aerial, an aerial pre-amp, a radio receiver and an A/D interface. The signal from the aerial is amplified by the pre-amp and this is then fed to the receiver. The output of this then feeds the LYNX via the interface. Software held within the LYNX then decodes the information and the result is displayed on the screen. This set-up will receive weather pictures from the orbiting satellites such as NOAA. A more complex system requiring a dish aerial and a down/converter will give access to the METEOSTAT geo-stationary satellites and will inter-connect with the above system. My local technical college has this full system driving a 128K LYNX and TIMESTEP can provide it all.

The software can currently be supplied on disk for either the 96K or the 128K and also on tape for the 96K. When loaded, it auto-runs and produces a menu of 10 options on the screen. The software supports both NOAA and the METEOSTAT weather satellites.

Options available allow control over the height and width of a picture thereby allowing the user to magnify or reduce any part of a picture as it is received, either live from a satellite or more usually after it has first been recorded as an audio tone onto tape. The resulting weather picture can be made "negative" or have various colours manipulated to make it look different before storing on tape, disk or on the 128K, to a printer. The software also allows a title to bewritten on the picture before storage.

And the results of all this ----- quite frankly they are amazing!

The 96K produces a picture that is better than its BBC equivalent, and the 128K produces a picture that is even better than commercially available weather picture framestores!

Fine cloud structure and swirls are often seen and with a clear day, when looking at the land masses, many features such as the Isle of Wight or Isle of MAN, London, Birmingham, the rivers Thames and Medway estuaries and even Mount Etna on Scicily can be clearly picked out.

To be honest, some pictures look better to me than those often seen on the TV weather bulletins. Unfortunately, a printer dump does not do justice to the pictures as it does not show the shades of grey that make up the clouds and so the best idea is to take a black and white photograph of the monitor screen. Nevertheless, I have supplied the editor with some printer dumps of the land masses of BRITAIN and ITALY, which at least shows how the software can "blow-up" a picture. Ed. As shown opposite.

Ratings:- For kit quality 10/10
For software quality ... 9/10

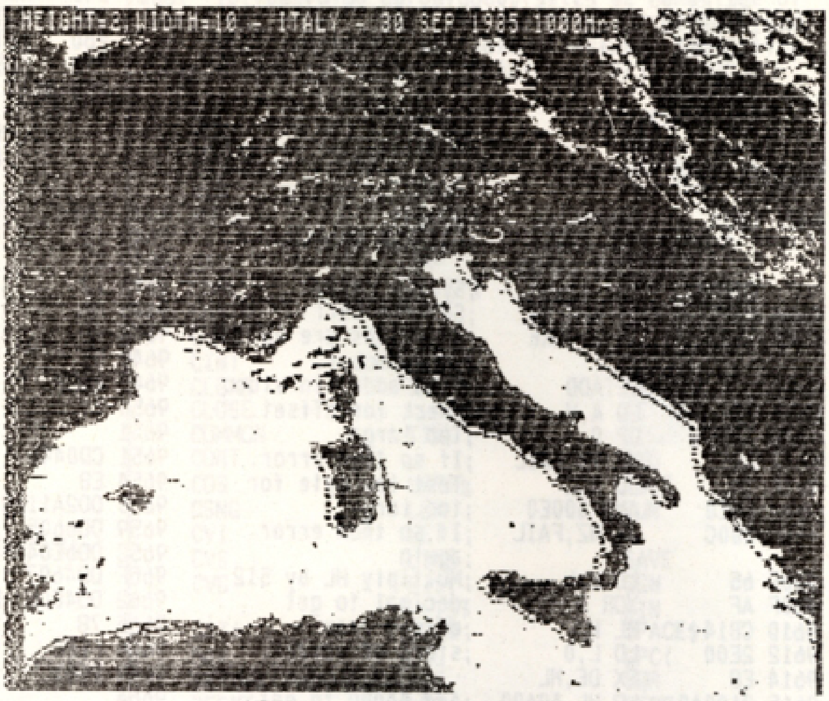
Address:- TIMESTEP ELECTRONICS
WICKHAMBROOK,
NEWMARKET,
SUFFOLK.

Tel:- 0440 820040

JOHN DOERR.

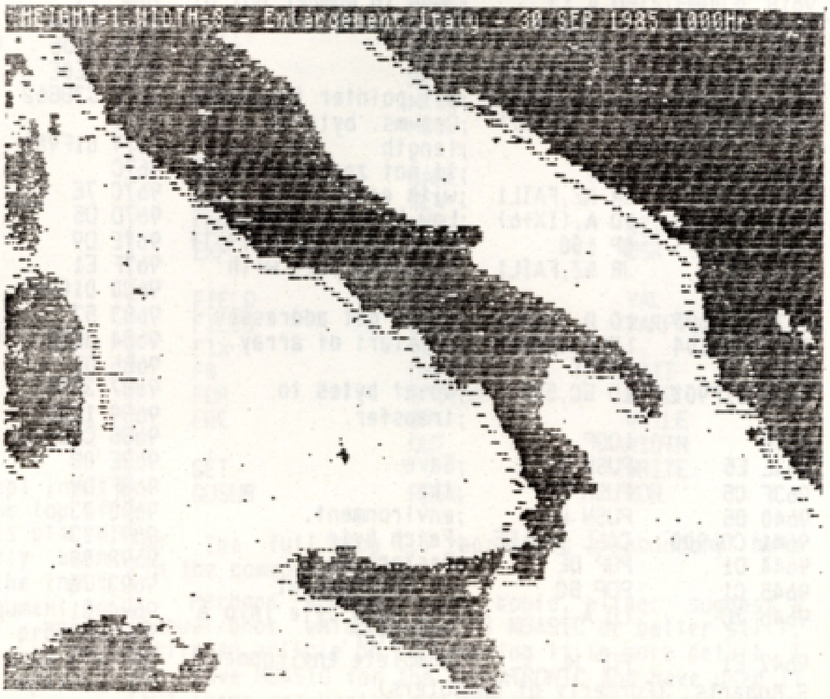
Editor's Note.

Since this article was received, there have been changes to the situation concerning TIMESTEP. Please contact the editor for the latest information.



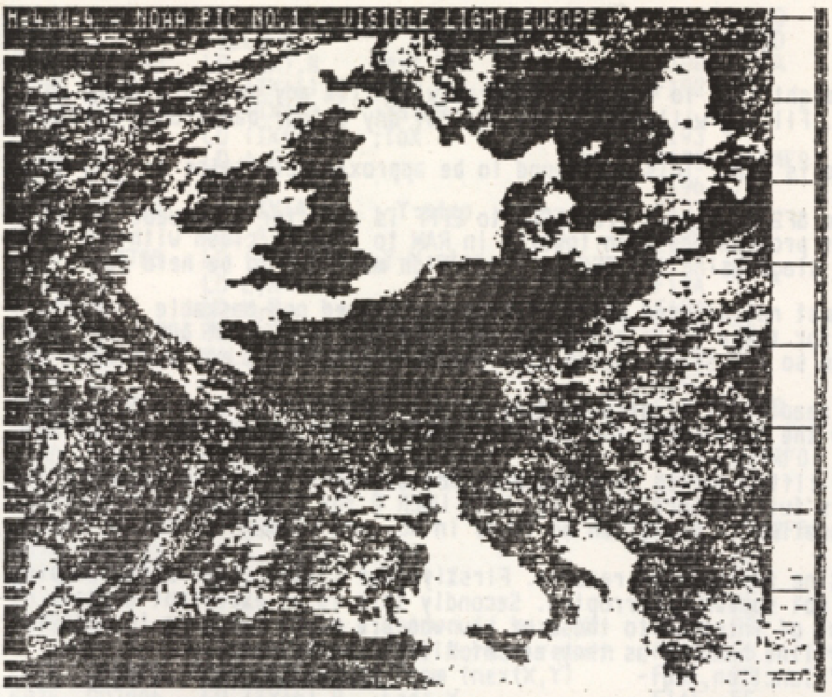
METEOSTAT 2 TRANSMISSION

Covering ITALY etc. The country boundary, latitude and longtitude marks are put in by the satellite.

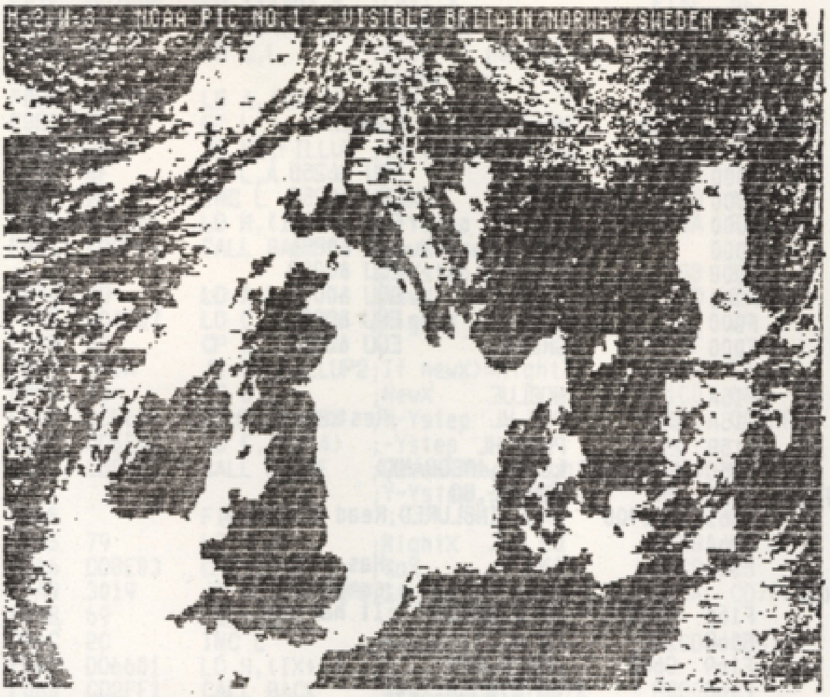


Close up of ITALY

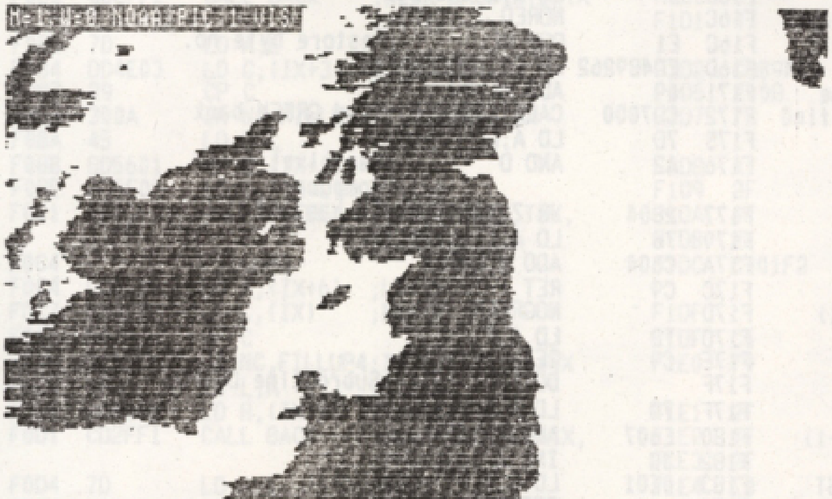
Note Mt. ETNA in Scicily and some lakes near the YUGOSLAV coastline. The boundary put in by the satellite is notoriously inaccurate!



General view of EUROPE
This shot is modified by the software, to reduce the height & increase the width of the visible light picture.



NOAA TRANSMISSION
An enlargement of that above.



Close up of BRITAIN
This shot is a software blow-up, which is too big to fit on the screen. Even further magnification!

LINE FILL

This is another fill facility that users might like to try. The program will fill any suitable shape with horizontal lines in the current ink colour. Filling will take place against any colour background or paper colour. The protection levels work as usual.

A special horizontal line filling routine is used. This was found to be approximately twice as fast than by using the normal line drawing.

The routine occupies memory from &EC00 to &F264. Memory from EC00 to EFFF is used for the queue and must not be used by the user's program. (Ed. This program has been lowered in RAM to avoid a clash with the disk code). The filling can be halted at any stage using the ESCape key, which may have to be held for a few seconds, this is due to the algorithm used.

It should be noted that the BREAK key must not be used while filling nor shared non-maskable interrupts to be made to occur by the user. The reason for these restrictions is that the ROM routine at &0836 switches out the RAM while accessing the video RAM and so the stack (being part of the user RAM) is not available for access.

The machine code can be relocated fairly readily (I've already done this!). However the base of the queue in RAM should be put at a similar address to the above, and the variable QMASK (which determines the maximum queue size) adjusted accordingly.

The routine "DOTCOLOUR" at &F145 can be lifted out to form the basis of a general pixel reading routine to be put in a BASIC code line. This is fully relocatable, see LYNX USER 2 for a suitable application. "DOTCOLOUR" can read the screen at any location (x,y), given as (y,x) in HL, and return the colour of the pixel at that location in A.

Ed. I have included this second fill routine for several reasons. Firstly, due to my rather sloppy typing some of the code in Issue 2 of the mag. got somewhat corrupted. Secondly this is an excellent example of fully detailed and noted code, this will be of interest to those of you who are maybe dabbling in code for the first time. I have left out the assembler line numbers as they are of little significance.

The program is accessed through a 'USER' function, eg:-

```
DPOKE &627D,&F000
LET P=USER0(X+256*Y)
PRINT P
```

```
Addr. Code      Mnemonics      Comments

F000            INBLURED       EQU &0069
F000            INGREEN        EQU &0070
F000            FPINT          EQU &3497
F000            BLUEBANK       EQU &628E
F000            REDBANK        EQU &6290
F000            GREENBANK      EQU &6292
F000            QBASE          EQU &EC00
F000            QMASK          EQU &00F3
F000            ORIGIN         EQU &F000
F000  180C      JR FILLFROM
F004            FIRST          ;DEFS 2
F006            LAST           ;DEFS 2
F00E            WKAREA         ;DEFS 8
F00E            FILLFROM       ;Subroutine FILLFROM
F00E  2100EC    LD HL,QBASE    ;
F011  2202F0    LD (FIRST),HL
F014  2204F0    LD (LAST),HL   ;Set up queue
F017  CD9734    CALL FPINT     ;Posn to fill from
                               ;=(X,Y)
F01A  E5        PUSH HL
F01B  CD3CF1    CALL RDSCRN
F01E  E1        POP HL
F01F  C0        RET NZ         ;Return if not paper
                               ;coloured pixel.
F020  E5        PUSH HL
F021  CD04F1    CALL FILLALONG;Fill first line
                               ;from (X,Y) and find
                               ;L & R.
F024  4D        LD C,L
F025  E1        POP HL
F026  E5        PUSH HL
F027  54        LD D,H
F028  14        INC D
F029  1E01      LD E,1
F02B  CD93F1    CALL QUEUE     ;QUEUE(L,R,Y+1,1)
F02E  E1        POP HL
F02F  54        LD D,H
F030  15        DEC D
F031  1EFF      LD E,-1
F033  CD93F1    CALL QUEUE     ;QUEUE(L,R,Y-1,-1)
F036            REPEAT         ;Repeat
F036  CDA6F1    CALL UNQUEUE   ;until FIRST=LAST
F039  CD49F0    CALL FILLUP
F03C  2A02F0    LD HL,(FIRST)
F03F  ED5B04F0  LD DE,(LAST)
F043  CDB9F1    CALL CPHLDE
F046  20EE      JR NZ REPEAT
F048  C9        RET
```

```
Addr. Code      Mnemonics      Comments

F000            INK            EQU &625B
F000            PAPER          EQU &625C
F000            PROTECT        EQU &626B
F000            ROMOUT         EQU &0886
F000            BLACK          EQU &0000
F000            BLUE           EQU &0001
F000            RED            EQU &0002
F000            GREEN          EQU &0004

F15A            NOBLUE
F15A  E1        POP HL         ;Restore byte no.
F15B  E5        PUSH HL
F15C  ED4B9062  LD BC,(REDBANK)
F160  09        ADD HL,BC
F161  CD6900    CALL INBLURED;Read RED bank
F164  7D        LD A,L
F165  A2        AND D          ;Has pixel red
                               ;component
F166  2804      JR Z,NORED     ;It has not
F168  7B        LD A,E
F169  C602      ADD A,RED
F16B  5F        LD E,A
F16C            NORED
F16C  E1        POP HL         ;Restore byte no.
F16D  ED4B9262  LD BC,(GREENBANK)
F171  09        ADD HL,BC
F172  CD7000    CALL INGREEN   ;Read GREEN bank
F175  7D        LD A,L
F176  A2        AND D          ;Has pixel green
                               ;component
F177  2804      JR Z,NOGREEN   ;It has not
F179  7B        LD A,E
F17A  C604      ADD A,GREEN
F17C  C9        RET
F17D            NOGREEN
F17D  7B        LD A,E
F17E  C9        RET
F17F            DOTPOS         ;Subroutine DOTPOS
F17F  7D        LD A,L
F180  E607      AND 7
F182  3C        INC A
F183  0E01      LD C,1
F185            DOTPOS1
F185  CB09      RRC C
F187  3D        DEC A
F188  20FB      JR NZ,DOTPOS1;Find bit posn in C
F18A  0603      LD B,3
F18C            DOTPOS2
```

```
F049  DD2106F0 LD IX,WKAREA
F04D           FILLUP0       ;
F04D  DD7000   LD (IX),B     Next X
F050  DD7201   LD (IX+1),D   ;Y
F053  DD7302   LD (IX+2),E   ;YSTEP
F056  DD7103   LD (IX+3),C   ;ToX
F059  7B       LD A,E
F05A  ED44     NEG
F05C  DD7704   LD (IX+4),A   ;-Y step
F05F  82       ADD A,D
F060  DD7705   LD (IX+5),A   ;Y-Y step
F063  68       LD L,B
F064  62       LD H,D
F065  E5       PUSH HL
F066  CD3CF1   CALL RDSCRN   ;At (next X,Y)
F069  E1       POP HL
F06A  2808     JR Z,FILLUP1
F06C  E5       PUSH HL
F06D  CD2FF1   CALL BACK
F070  DD7500   LD (IX),L     ;Next X=Back(NextX,Y)
F073  E1       POP HL
F074           FILLUP1
F074  DD7E03   LD A,(IX+3)   ;ToX
F077  DD6E00   LD L,(IX)     ;Next X
F07A  BD       CP L
F07B  D8       RET C         ;Return if next X>
                             ;ToX. Normal return
F07C  CD04F1   CALL FILLALONG;From (nextX,Y)
F07F  DD7006   LD (IX+6),B   ;Left X
F082  DD7507   LD (IX+7),L   ;Right X
F085  4D       LD C,L        ;Save if needed for
                             ;queue.
F086  DD7E03   LD A,(IX+3)   ;ToX
F089  BD       CP L
F08A  3019     JR NC,FILLUP  ;If toX>=Right X
F08C  6F       LD L,A
F08D  2C       INC L         ;ToX+1
F08E  DD6605   LD H,(IX+5)   ;Y-Ystep
F091  CD2FF1   CALL BACK     ;NewX=Back(ToX+!,
                             ;Y-Ystep).
F094  7D       LD A,L        ;NewX
F095  DD4E07   LD C,(IX+7)   ;RightX
F098  B9       CP C
F099  300A     JR NC,FILLUP2;If newX>=RightX
F09B  45       LD B,L        ;NewX
F09C  DD5605   LD D,(IX+5)   ;Y-Ystep
F09F  DD5E04   LD E,(IX+4)   ;-Ystep
F0A2  CD93F1   CALL QUEUE    ;Queue(NewX,RightX,
                             ;Y-Ystep,-Ystep).
F0A5           FILLUP2
F0A5  79       LD A,C        ;RightX
F0A6  DDBE03   CP (IX+3)     ;ToX
F0A9  3019     JR NC,FILLUP3;If rightX>=ToX
F0AB  69       LD L,C
F0AC  2C       INC L         ;RightX+1
F0AD  DD6601   LD H,(IX+1)   ;Y
F0B0  CD2FF1   CALL BACK     ;NewX=BACK(RightX
                             ;+1,Y)
F0B3  7D       LD A,L
F0B4  DD4E03   LD C,(IX+3)   ;ToX
F0B7  B9       CP C
F0B8  300A     JR NC,FILLUP3;If newX>=ToX
F0BA  45       LD B,L        ;NewX
F0BB  DD5601   LD D,(IX+1)   ;Y
F0BE  DD5E02   LD E,(IX+2)   ;Ystep
F0C1  CD93F1   CALL QUEUE    ;Queue(NewX,ToX,
                             ;Y,Ystep)
F0C4           FILLUP3
F0C4  DD7E06   LD A,(IX+6)   ;LeftX
F0C7  DD4E00   LD C,(IX)     ;NextX
F0CA  B9       CP C
F0CB  3018     JR NC,FILLUP4;If leftX>=NextX
F0CD  6F       LD L,A        ;LeftX
F0CE  DD6605   LD H,(IX+5)   ;Y-Ystep
F0D1  CD2FF1   CALL BACK     ;NewX=BACK(LeftX,
                             ;Y-Ystep)
F0D4  7D       LD A,L        ;NewX
F0D5  DD4E00   LD C,(IX)     ;NextX
F0D8  B9       CP C
F0D9  300A     JR NC,FILLUP4;If newX>=NextX
F0DB  45       LD B,L        ;NewX
F0DC  DD5605   LD D,(IX+5)   ;Y-Ystep
F0DF  DD5E04   LD E,(IX+4)   ;-Ystep
```

```
F18C  CB3C     SRL H
F18E  CB1D     RR L
F190  10FA     DJNZ DOTPOS2
F192  C9       RET           ;Byte no in HL, bit
                             ;posn in C
F193           QUEUE         ;Subroutine QUEUE
F193  2A04F0   LD HL,(LAST)
F196  70       LD (HL),B
F197  23       INC HL
F198  71       LD (HL),C
F199  23       INC HL
F19A  72       LD (HL),D
F19B  23       INC HL
F19C  73       LD (HL),E
F19D  23       INC HL
F19E  7C       LD A,H
F19F  E6F3     AND QMASK
F1A1  67       LD H,A
F1A2  2204F0   LD (LAST),HL
F1A5  C9       RET
F1A6           UNQUEUE       ;Subroutine UNQUEUE
F1A6  2A02F0   LD HL,(FIRST)
F1A9  46       LD B,(HL)
F1AA  23       INC HL
F1AB  4E       LD C,(HL)
F1AC  23       INC HL
F1AD  56       LD D,(HL)
F1AE  23       INC HL
F1AF  5E       LD E,(HL)
F1B0  23       INC HL
F1B1  7C       LD A,H
F1B2  E6F3     AND QMASK
F1B4  67       LD H,A
F1B5  2202F0   LD (FIRST),HL
F1B8  C9       RET
F1B9           CPHLDE        ;Subroutine CPHLDE
F1B9  AF       XOR A         ;Zero flags
F1BA  E5       PUSH HL       ;Dummy subtraction
                             ;of HL and DE
F1BB  ED52     SBC HL,DE
F1BD  E1       POP HL
F1BE  C9       RET
F1BF           FILLINE       ;Subroutine FILLINE
F1BF  E5       PUSH HL       ;Save (R,Y)
F1C0  68       LD L,B        ;Get (L,Y)
F1C1  CD7FF1   CALL DOTPOS   ;L bit in C,
                             ;L byte in HL
F1C4  CD5CF2   CALL LEFTMASK;L mask in C,
                             ;L byte in HL
F1C7  EB       EX DE,HL      ;L byte in DE
F1C8  E1       POP HL        ;Restore (R,Y)
F1C9  C5       PUSH BC       ;Save L mask
F1CA  CD7FF1   CALL DOTPOS   ;R bit in C,
                             ;R byte in HL
F1CD  CD61F2   CALL RITEMASK;R mask in A,
                             ;R byte in HL
F1D0  C1       POP BC        ;Restore L mask
F1D1  47       LD B,A        ;L mask in C,
                             ;R mask in B
F1D2  CDB9F1   CALL CPHLDE
F1D5  2008     JR NZ,FILLEFT;L byte<>R byte
F1D7  78       LD A,B        ;Calculate screen
                             ;byte and mask
F1D8  A1       AND C
F1D9  2F       CPL
F1DA  4F       LD C,A
F1DB  2F       CPL
F1DC  C301F2   JP OUTBYTE    ;O/P byte to screen
                             ;and return
F1DF           FILLEFT
F1DF  C5       PUSH BC       ;Save masks
F1E0  79       LD A,C        ;Calculate screen
                             ;byte and mask
F1E1  2F       CPL
F1E2  4F       LD C,A
F1E3  2F       CPL
F1E4  EB       EX DE,HL      ;Get L byte in HL
F1E5  CD01F2   CALL OUTBYTE ;O/P byte to screen
F1E8  EB       EX DE,HL      ;L byte back in DE
F1E9  C1       POP BC        ;Restore masks
F1EA           FILLMID
F1EA  13       INC DE        ;L byte=L byte+1
F1EB  CDB9F1   CALL CPHLDE   ;Any (more) middle
```

```
F0E2  CD93F1    CALL QUEUE     ;Queue(NewX,NextX,
                               ;Y-Ystep,-Ystep)
F0E5            FILLUP4
F0E5  DD4606    LD B,(IX+6)    ;LeftX->NextX
F0E8  DD4E07    LD C,(IX+7)    ;RightX->ToX
F0EB  DD5E02    LD E,(IX+2)    ;Ystep
F0EE  DD7E01    LD A,(IX+1)    ;Y
F0F1  83        ADD A,E
F0F2  57        LD D,A         ;Y+Ystep
F0F3  DB80      IN A,(&80)
F0F5  CB77      BIT 6,A        ;Test for ESCape key
F0F7  C24DF0    JP NZ,FILLUP0;Not pressed
F0FA  2100EC    LD HL,QBASE    ;Make FIRST=LAST so
F0FD  2202F0    LD (FIRST),HL;FILLFROM terminates
F100  2204F0    LD (LAST),HL   ;when it is returned
F103  C9        RET            ;to special return
F104            FILLALONG      ;Subroutine FILLALONG
F104  E5        PUSH HL        ;Save (X,Y)
F105  45        LD B,L         ;Max. no. of posns to
                               ;test
F106  4C        LD C,H         ;Save Y
F107            FINDL
F107  C5        PUSH BC
F108  68        LD L,B         ;X posn to test at
F109  61        LD H,C         ;Restore Y
F10A  CD3CF1    CALL RDSCRN    ;Read pixel at (X,Y)
F10D  C1        POP BC
F10E  2002      JR NZ,FOUNDL
F110  10F5      DJNZ FINDL
F112            FOUNDL
F112  04        INC B          ;Do not test at X=0
F113  E1        POP HL         ;Restore (X,Y)
F114  C5        PUSH BC        ;Save left
F115            FINDR
F115  E5        PUSH HL        ;
F116  CD3CF1    CALL RDSCRN
F119  E1        POP HL
F11A  2006      JR NZ,FOUNDR
F11C  2C        INC L
F11D  7D        LD A,L
F11E  FEFF      CP 255
F120  38F3      JR C,FINDR     ;Do not test at X=255
F122            FOUNDR
F122  2D        DEC L
F123  C1        POP BC         ;Restore left
F124  7D        LD A,L
F125  90        SUB B
F126  D8        RET C          ;Do nothing if right
                               ;<left
F127  C5        PUSH BC        ;Save left in B
F128  E5        PUSH HL        ;Save right in L
F129  CD8FF1    CALL FILLINE   ;Fill line between
                               ;left and right
F12C  E1        POP HL         ;Restore right
F12D  C1        POP HL         ;Restore left
F12E  C9        RET
F12F            BACK           ;Subroutine BACK
F12F  E5        PUSH HL        ;Save (X,Y)
F130  CD3CF1    CALL RDSCRN    ;Read pixel at (,Y)
F133  E1        POP HL
F134  C8        RET Z          ;Black pixel found ?
F135  2C        INC L
F136  7D        LD A,L
F137  FEFF      CP 255         ;Is pos extreme right
F139  38F4      JR C,BACK
F13B  C9        RET
F13C            RDSCRN         ;Subroutine RDSCRN
F13C  CD45F1    CALL DOTCOL
F13F  5F        LD E,A         ;Pixel colour
F140  3A5C62    LD A,(PAPER)
F143  93        SUB E          ;Is pixel the same
                               ;colour as the paper
F144  C9        RET            ;A=0 If it is, A<>0,
                               ;if not
F145            DOTCOL         ;Subroutine DOTCOLour
F145  CD7FF1    CALL DOT POS
F148  51        LD D,C         ;Save bit posn in D
F149  1E00      LD E,BLACK
F14B  E5        PUSH HL        ;Save byte number
F14C  ED4B8E62  LD BC,(BLUEBANK)
F150  09        ADD HL,BC
F151  CD6900    CALL INBLURED;Read BLUE bank
F154  7D        LD A,L         ;Save byte from
```

```
                               ;bytes to fill?
F1EE  280D      JR Z,FILLRITE;No (more)
F1F0  C5        PUSH BC        ;Protect masks
F1F1  3EFF      LD A,&FF       ;Screen byte
F1F3  0E00      LD C,0         ;Screen mask
F1F5  EB        EX DE,HL       ;L byte in HL
F1F6  CD01F2    CALL OUTBYTE   ;O/P screen byte
F1F9  EB        EX DE,HL       ;L byte back in DE
F1FA  C1        POP BC         ;Restore masks
F1FB  18ED      JR FILLMID     ;Next middle byte
F1FD            FILLRITE
F1FD  78        LD A,B         ;Screen byte and
                               ;mask
F1FE  2F        CPL
F1FF  4F        LD C,A
F200  2F        CPL
F201            OUTBYTE
F201  D5        PUSH DE
F202  57        LD D,A         ;Byte to be O/P
F203  59        LD E,C         ;Mask
F204  3A6B62    LD A,(PROTECT)
F207  0F        RRCA
F208  3816      JR C,TESTRED   ;Blue protected
F20A  D5        PUSH DE
F20B  3A5B62    LD A,(INK)
F20E  0F        RRCA
F20F  3802      JR C,RITEBLUE;Write blue
                               ;Component if
                               ;present
F211  1600      LD D,0         ;Else blank out
F213            RITEBLUE
F213  3EE8      LD A,&E8
F215  08        EX AF,AF'
F216  3E63      LD A,&63
F218  ED4B8E62  LD BC,(BLUBANK)
F21C  CDB608    CALL ROMOUT    ;O/P blue component
                               ;or blank
F21F  D1        POP DE
F220            TESTRED
F220  3A6B62    LD A,(PROTECT)
F223  CB4F      BIT 1,A
F225  2017      JR NZ,TESTGRN;Red protected
F227  D5        PUSH DE
F228  3A5B62    LD A,(INK)
F22B  CB4F      BIT 1,A
F22D  2002      JR NZ,RITERED;Write red component
                               ;if present
F22F  1600      LD D,0         ;Else blank out
F231            RITERED
F231  3EE8      LD A,&E8
F233  08        EX AF,AF'
F234  3E63      LD A,&63
F236  ED4B9062  LD BC,(REDBANK)
F23A  CDB608    CALL ROMOUT    ;O/P red component
                               ;or blank out
F23D  D1        POP DE
F23E            TESTGRN
F23E  3A6B62    LD A,(PROTECT)
F241  CB57      BIT 2,A
F243  2015      JR NZ,RESTORE;Green protected
F245  3A5B62    LD A,(INK)
F248  CB57      BIT 2,A
F24A  2002      JR NZ,RITEGRN;Write green
                               ;component if
                               ;present
F24C  1600      LD D,0         ;Else blank out
F24E            RITEGRN
F24E  3EE4      LD A,&E4
F250  08        EX AF,AF'
F251  3E65      LD A,&65
F253  ED4B9262  LD BC,(GRNBANK)
F257  CDB608    CALL ROMOUT    ;O/P green component
                               ;or blank out
F25A            RESTORE
F25A  D1        POP DE
F25B  C9        RET
F25C            LEFTMASK       ;Subroutine LEFTMASK
F25C  79        LD A,C
F25D  0D        DEC C
F25E  81        ADD A,C
F25F  4F        LD C,A         ;LEFTMASK=C+(C-1)
F260  C9        RET
F261            RIGHTMASK      ;Subroutine
```

```
                            ;screen
F155  A2        AND D       ;Has pixel blue
                            ;component
F156  2802      JR Z,NOBLUE ;It has not
F158  1E01      LD E,BLUE
```

```
                                ;RIGHTMASK
F261  0D        DEC C
F262  79        LD A,C
F263  2F        CPL             ;RIGHTMASK=CPL(C-1)
F264  C9        RET
```

List of Routines & Pointers

0000	BLACK
0001	BLUE
0002	RED
0004	GREEN
0069	INBLURED
0070	INGREEN
00F3	QMASK
08B6	ROMOUT
3497	FPINT
628E	BLUBANK
6290	REDBANK
6292	GREENBANK
625B	INK
625C	PAPER
626B	PROTECT
EC00	QBASE
F002	FIRST
F004	LAST
F006	WKAREA
F00E	FILLFROM
F036	REPEAT
F049	FILLUP
F04D	FILLUP0
F074	FILLUP1
F0A5	FILLUP2
F0C4	FILLUP3
F0E5	FILLUP4
F104	FILLALONG
F107	FINDL
F112	FOUNDL
F115	FINDR
F122	FOUNDR
F12F	BACK
F13C	RDSCRN
F145	DOTCOL
F15A	NOBLUE
F16C	NORED
F17D	NOGREEN
F17F	DOTPOS
F185	DOTPOS1
F18C	DOTPOS2
F193	QUEUE
F1A6	UNQUEUE
F1B9	CPHLDE
F1BF	FILLINE
F1DF	FILLEFT
F1EA	FILLMID
F1FD	FILLRIGHT
F201	OUTBYTE
F213	WRITEBLUE
F220	TESTRED
F231	WRITERED
F23E	TESTGREEN
F24E	WRITEGREEN
F25A	RESTORE
F25C	LEFTMASK
F261	RIGHTMASK

David Peter.

SINE-WAVE

Another interesting program for the TANDY !

```
10 LPRINT CHR$(18)
20 LPRINT "C1"
30 LPRINT "M0,-100"
40 LPRINT "I"
50 LPRINT "M100,0"
60 FOR A=0 TO 360 STEP 5
70   LPRINT "D";A+100;",";SIN(RAD(A))*10
0
80 NEXT A
90 LPRINT "M100,-100"
100 LPRINT "C0"
110 LPRINT "X0,20,10"
120 LPRINT "M100,0"
130 LPRINT "X1,90,4"
140 LPRINT "M300,90"
150 LPRINT "J90,0"
160 LPRINT "C3"
170 LPRINT "S0"
180 FOR A=-1 TO 1 STEP 0.2
190   LPRINT "M60,";A*100
200   LPRINT "P";A
210 NEXT A
220 FOR A=90 TO 360 STEP 90
230   LPRINT "M";A+90;",-20"
240   LPRINT "P";A
250 NEXT A
260 LPRINT "Q3"
270 LPRINT "S1"
280 LPRINT "M30,-30"
290 LPRINT "C2"
300 LPRINT "PY AXIS"
310 LPRINT "Q0"
320 LPRINT "M360,15"
330 LPRINT "PX AXIS"
340 LPRINT "M300,100"
350 LPRINT "PY=SIN(X)"
360 LPRINT "H"
370 LPRINT "C0"
380 LPRINT "A"
```

MOD and DIV

Two maths functions which are poorly explained in either of the LYNX manuals are MOD and DIV. They can be quite useful when used in a loop (eg. FOR - NEXT), to select one item within the loop from the other values. They are in real terms quite simple to understand as they are both related to the division of numbers.

When any number is divided by any other number, (except zero, which the LYNX will not divide by), two results occur; 1) the number of times the denominator goes into the numerator and 2) a remainder (which may be zero!).

The DIV command gives the first result, for example :-

10 DIV 5 gives the result 2

This is because 5 'goes into' 10 twice with no remainder whereas :-

10 DIV 6 gives the result 1

because 6 'goes into' 10 once with a remainder of four (although this is not shown in the answer).

The MOD command, on the other hand, only shows the remainder when the two numbers are divided.

10 MOD 5 gives the result 0

because as stated before, 5 'goes into' 10 twice with no remainder, whereas,

10 MOD 6 gives the result 4

because 6 'goes into' 10 once with a remainder of four.

Here is a short program to demonstrate both functions which I hope will also help to explain the operations of the two commands.

```
100 CLS
110 PRINT "A","B","A MOD B","A DIV B"
120 LET A=0,B=0
130 FOR A=1 TO 4
140   FOR B=1 TO 4
150     PRINT A,B,,A MOD B,,A DIV B
160   NEXT B
170 NEXT A
```

G. CHRISTOFI

KEY-VALUE

With the LYNX keyboard, as with other machines, it is possible to obtain other outputs by multiple key-presses. Some of these actions are already familiar to those who use CP/M for example. One of the prime weaknesses of the current LYNX keyboard is that it is rather small when compared with the trends which have appeared in recent years. So that for example, PERFECT WRITER under CP/M becomes an onerous task to use due to the multiple key-presses involved with its use. This makes for very un-friendly usage.

Because of the above comments, an investigation has been carried out, to see what would be involved in implementing a larger key-board complete with a numeric keypad and others. Part of this investigation was looking into the aspect of multiple key strokes, either double (ESC.+L=LIST) or even triple key actions. A program was therefore devised to illustrate these figures. This is listed below. It may be of interest to those of you who wish to protect a piece of software by a triple key action, only known to you.

```
100 CLS
110 DIM N(7)
120 REM LINK ON
130 PRINT "  ";
140 FOR V=7 TO 0 STEP -1
150   LET N(V)=2**V
160   PRINT CHR$(32*(V<7));CHR$(32*(V<7)
);CHR$(32*(V<7));CHR$(32*(V<4));N(V);CHR
$(19*(V=0));
170 NEXT V
180 PRINT "------------------------------
-----------";
190 REM LINK OFF
200 INPUT "SINGLE+DOUBLE/TRIPLE KEYPRESS
 (S/T)";K$
210 IF K$="S" THEN  PROC S/D
220 IF K$="T" THEN  PROC T
230 REM LINK OFF
240 END
250 DEFPROC S/D
260 REM * SINGLE KEYPRESS ROUTINE *
270 REM LINK ON
280 PRINT  @ 3,25;"S";
290 FOR X=7 TO 0 STEP -1
300   IF N(X)>0 THEN  LET N=255-N(X)
310   IF N(X)=0 THEN  GOTO 350
320   PROC HEX(N)
330   PRINT CHR$(32*(X<7));"  ";A$;
340   GOTO 360
350   PRINT CHR$(32*(X<7));"  ==";
360 NEXT X
370 PRINT "------------------------------
-----------";
380 REM * DOUBLE KEYPRESS ROUTINE *
390 FOR X=7 TO 0 STEP -1
400   LET L$=" DOUBLE "
410   PRINT MID$(L$,8-X,1);
420   FOR Y=7 TO 0 STEP -1
430     IF N(X)<>N(Y) THEN  LET N=255-(N
(X)+N(Y))
440     IF N(X)=N(Y) THEN  GOTO 480
450     PROC HEX(N)
460     PRINT CHR$(32*(Y<7));"  ";A$;
470     GOTO 490
480     PRINT CHR$(32*(Y<7));"   =";
490   NEXT Y
500   IF X=0 AND Y=0 THEN  PRINT CHR$(19
);
510 NEXT X
520 PRINT "------------------------------
-----------";
530 ENDPROC
```

R B JONES

```
540 DEFPROC HEX(N)
550 REM * DEC. TO HEX. ROUTINE *
560 IF N>255 THEN  ENDPROC
570 LET n=N DIV 16,m=N MOD 16
580 FOR i=0 TO 9
590   IF n=i THEN  LET A$=CHR$(48+i)
600   IF m=i THEN  LET B$=CHR$(48+i)
610   IF n=10+i*(i<7) THEN  LET A$=CHR$(
65+i*(i<7))
620   IF m=10+i*(i<7) THEN  LET B$=CHR$(
65+i*(i<7))
630 NEXT i
640 LET A$=A$+B$
650 ENDPROC
660 DEFPROC T
670 REM * TRIPLE KEYPRESS ROUTINE *
680 REM LINK ON
690 LET C=6,D=7
700 PRINT  @ 3,25;
710 FOR Z=D TO C STEP -1
720   FOR X=7 TO 0 STEP -1
730     LET L$=" TRIPLE "
740     PRINT MID$(L$,8-X,1);
750     FOR Y=7 TO 0 STEP -1
760       IF N(X)+N(Y)+N(Z)>255 THEN  GO
TO 820
770       IF N(X)+N(Y)+N(Z)<=255 THEN  L
ET N=255-(N(X)+N(Y)+N(Z))
780       IF N(Y)=N(X) OR N(Y)=N(Z) THEN
  GOTO 840
790       PROC HEX(N)
800       PRINT CHR$(32*(Y<7));"  ";A$;
810       GOTO 850
820       PRINT CHR$(32*(Y<7));"  **";
830       GOTO 850
840       PRINT CHR$(32*(Y<7));"  ==";
850     NEXT Y
860     IF X=0 AND Y=0 AND Z=C THEN  PRI
NT CHR$(19);
870   NEXT X
880 NEXT Z
890 PRINT "------------------------------
-----------";
900 IF (Z=-1) THEN  ENDPROC
910 IF (Z<>-1) THEN  LET D=D-2,C=C-2
920 PRINT  @ 3,195;"PRESS SPACE FOR NEXT
 BLOCK OF KEYTRIPLES";
930 LET G=GETN
940 PRINT "
          ";
950 GOTO 700
960 ENDPROC
```

REMOTE CASSETTE CONTROL

These circuits were accidently left out of Issue 3 of the magazine:-

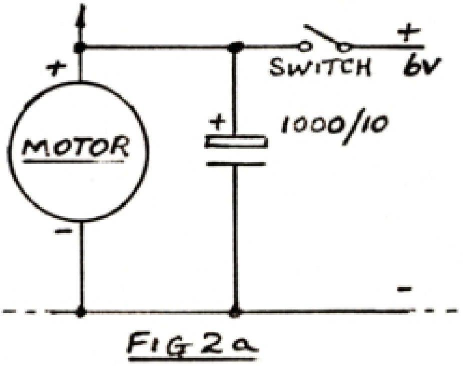


FIG 2a

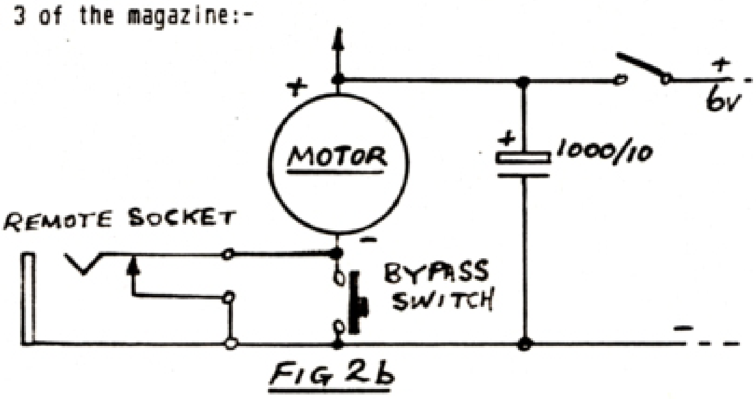


FIG 2b

"WAVES"

****

This is a program for the 128K LYNX. It enables the user to define ENVELOPES for use with the SOUND command.

CONTROLS

```
RIGHT ARROW KEY = LINE FORWARD
UP ARROW KEY    = DIAGONAL UP
DOWN ARROW KEY  = DIAGONAL DOWN
[ KEY           = LINE DOWN
] KEY           = LINE UP
```

FIRST MENU

OPTION

1. Gives wave 1 with wave 2 as a base.

2. Adds wave 1 and wave 2 together.

3. Adds wave 1 and wave 2 together with wave 1 as a base.

4. Wave 1.

5. Wave 2.

SECOND MENU

OPTION

1. This allows you to test notes with keys:- QWERTYIOPASDFGHJKL Space bar to exit.
2. This allows you to test the duration by inputing a number.
3. Return to Menu 1.

4. This allows you to input 2 new waves.
If zero is typed for origin, then previous wave can be used again.

```
100 DIM A(230),B(230),C(230)
110 RESERVE &E000
120 PROC SCREEN
130 PROC INPUT WAVE ONE
140 PROC INPUT WAVE TWO
150 INK 6
160 PROTECT 0
170 PRINT  @ 150,110;CHR$(1)(6);" 1 - WA
VE 2 AS BASE"; @ 150,120;" 2 - COMBINE 1
 & 2"; @ 150,130;" 3 - COMBINE 2 AS BASE
"; @ 150,140;" 4 - WAVE 1 "; @ 150,150;"
 5 - WAVE 2 ";
180 PAUSE 1000
190 LET A$=GET$
200 LET x=0
210 IF A$="1" THEN  LET x=1
220 IF x=1 THEN  PRINT  @ 150,110;CHR$(1
8);" 1 - WAVE 2 AS BASE ";CHR$(18);
230 IF A$="2" THEN  LET x=2
240 IF x=2 THEN  PRINT  @ 150,120;CHR$(1
250 IF A$="3" THEN  LET x=3
260 IF x=3 THEN  PRINT  @ 150,130;CHR$(1
8);" 3 - COMBINE 2 AS BASE ";CHR$(18);
270 IF A$="4" THEN  LET x=4
280 IF x=4 THEN  PRINT  @ 150,140;CHR$(1
8);" 4 - WAVE 1 ";CHR$(18);
290 IF A$="5" THEN  LET x=5
300 IF x=5 THEN  PRINT  @ 150,150;CHR$(1
8);" 5 - WAVE 2 ";CHR$(18);
310 IF x=0 THEN  GOTO 190
320 IF x=2 THEN  PROC EVAL WAVES
330 IF x=1 THEN  PROC POKE TWO
340 IF x=3 THEN  PROC POKE THREE
350 IF x=4 THEN  PROC POKE WAVE ONE
360 IF x=5 THEN  PROC POKE WAVE TWO
370 FOR T=110 TO 150 STEP 10
380   PRINT  @ 150,T;"
     ";
390 NEXT T
400 IF x=2 THEN  PROC DRAW RES
410 IF x=1 THEN  PROC DRAW
420 IF x=3 THEN  PROC DRAW THREE
430 IF x=4 THEN  PROC DRAW WAVE ONE (270
,137)
440 IF x=5 THEN  PROC DRAW WAVE TWO (270
,137)
450 PROTECT 0
460 INK 7
470 PRINT  @ 150,180;" 1 - TEST NOTES ";
 @ 150,190;" 2 - TEST DURATION"; @ 150,2
00;" 3 - NEW MODE"; @ 150,210;" 4 - NEW
WAVES";
480 IF INP(&0080)=254 THEN  PROC SOUND N
OTES
490 IF INP(&0280)=254 THEN  PROC TEST SO
UND
```

```
1570 FOR T=1 TO 5
1580   SOUND &E000,D
1590   PAUSE 10000
1600 NEXT T
1610 GOTO 1520
1620 PROTECT 0
1630 PRINT  @ 140,220;"
      ";
1640 EXT WRESET
1650 ENDPROC
1660 DEFPROC SOUND NOTES
1670 VDU 3,0,1,7
1680 PRINT  @ 150,230;"   SOUND NOTES";
1690 REPEAT
1700   LET X$=GET$
1710   IF X$="A" THEN  SOUND &E000,100
1720   IF X$="S" THEN  SOUND &E000,140
1730   IF X$="D" THEN  SOUND &E000,180
1740   IF X$="F" THEN  SOUND &E000,220
1750   IF X$="G" THEN  SOUND &E000,260
1760   IF X$="H" THEN  SOUND &E000,300
1770   IF X$="J" THEN  SOUND &E000,340
1780   IF X$="K" THEN  SOUND &E000,380
1790   IF X$="L" THEN  SOUND &E000,420
1800   IF X$="Q" THEN  SOUND &E000,460
1810   IF X$="W" THEN  SOUND &E000,500
1820   IF X$="E" THEN  SOUND &E000,540
1830   IF X$="R" THEN  SOUND &E000,580
1840   IF X$="T" THEN  SOUND &E000,620
1850   IF X$="Y" THEN  SOUND &E000,660
1860   IF X$="U" THEN  SOUND &E000,700
1870   IF X$="I" THEN  SOUND &E000,740
1880   IF X$="O" THEN  SOUND &E000,780
1890   IF X$="P" THEN  SOUND &E000,820
1900 UNTIL X$=" "
1910 PRINT  @ 150,230;"                  ";
1920 ENDPROC
1930 DEFPROC SCREEN
1940 EXT VRESET
1950 CCHAR 256*95+32
1960 VDU 2,0,4,1,2
1970 MOVE 20,123
1980 DRAW 20,200
1990 DRAW 250,200
2000 MOVE 20,400
2010 DRAW 20,470
2020 DRAW 250,470
2030 VDU 27
2040 PRINT  @ 3,53;"63"; @ 6,95;"0"; @ 3
,190;"63"; @ 6,230;"0";
2050 MOVE 270,123
2060 DRAW 270,200
2070 DRAW 500,200
2080 PRINT  @ 127,53;"63"; @ 130,95;"0";
2090 VDU 26,1,7,2,1
2100 PRINT  @ 21,0;" WAVE 1 "; @ 21,135;
```

```
500 IF INP(&0180)=254 THEN  GOTO 170
510 IF INP(&0180)=253 THEN  GOTO 120
520 GOTO 480
530 DEFPROC EVAL WAVES
540 FOR T=1 TO 230
550   LET C(T)=(A(T)+B(T))
560 NEXT T
570 PROC POKE
580 ENDPROC
590 DEFPROC DRAW RES
600 WINDOW 125,249,20,130
610 PROTECT RED
620 EXT CLW
630 INK 7
640 LET c=271,d=137+(C(1)/2)
650 MOVE c,d
660 FOR T=2 TO 230
670   DRAW c,137+(C(T)/2)
680   LET c=c+1
690 NEXT T
700 ENDPROC
710 DEFPROC INPUT WAVE TWO
720 EXT TRAP F
730 WINDOW 76,95,170,180
740 PRINT  @ 15,170;" ORIGIN 1 - 63 ";
750 LET F=0,f=0
760 INPUT O$
770 LET O=VAL(O$)
780 IF O=0 THEN  f=1,S=B(5)
790 IF F=21 THEN  EXT KLAXON
800 IF F=21 THEN  GOTO 730
810 IF f=1 AND F=0 THEN  PROC DRAW WAVE
TWO (20,400)
820 IF f=1 AND F=0 THEN  GOTO 1060
830 PRINT  @ 15,170;"
 ";
840 IF O>63 OR O<0 THEN  GOTO 740
850 LET c=20,d=470-O
860 EXT NOTRAP
870 EXT WRESET
880 REPEAT
890   IF INP(&0080)<>255 THEN  GOTO 930
900   IF INP(&0880)<>255 THEN  GOTO 930
910   IF INP(&0980)<>255 THEN  GOTO 930
920   GOTO 890
930   IF INP(&0080)<>239 THEN  LET d=d-1
940   IF INP(&0080)<>223 THEN  LET d=d+1
950   IF INP(&0980)<>223 THEN  LET d=d
960   IF INP(&0880)<>251 THEN  c=c-1,d=d
+1
970   IF INP(&0980)<>253 THEN  c=c-1,d=d
-1
980   IF d<404 THEN  LET d=d+1
990   IF d>470 THEN  LET d=d-1
1000   LET c=c+1
1010   LET B(c-20)=d-404
1020   IF B(c-20)>63 THEN  LET B(c-20)=63
1030   IF B(c-20)<1 THEN  LET B(c-20)=1
1040   DOT c,d
1050 UNTIL c=250
1060 ENDPROC
1070 DEFPROC INPUT WAVE ONE
1080 EXT TRAP F
1090 VDU 2,0,1,4,3,RED
1100 FAST ON
1110 WINDOW 76,95,30,40
1120 PRINT  @ 15,30;" ORIGIN 1 - 63 ";
1130 LET F=0,f=0
1140 INPUT O$
1150 LET O=VAL(O$)
1160 IF O=0 THEN  LET f=1,S=A(5)
1170 IF F=21 THEN  EXT KLAXON
1180 IF f=1 AND F=0 THEN  LET O=200-A(1)
1190 IF f=1 AND F=0 THEN  PROC DRAW WAVE
 ONE (20,0)
1200 IF f=1 AND F=0 THEN  GOTO 1460
1210 PRINT  @ 15,30;"
 ";
1220 IF VAL(O$)=0 THEN  GOTO 1120
1230 IF O<1 OR O>63 THEN  GOTO 1120
1240 MOVE 20,200-O
1250 LET c=20,d=200-O
1260 EXT NOTRAP
```

```
" WAVE 2 "; @ 84,0;" RESULT ";
2110 ENDPROC
2120 DEFPROC POKE
2130 LET c=1
2140 PROC SET FOR WAVE
2150 FOR T=1 TO 460 STEP 2
2160   POKE &E000+T,C(c)
2170   LET c=c+1
2180 NEXT T
2190 POKE &E000+461,0
2200 ENDPROC
2210 DEFPROC POKE TWO
2220 LET c=1
2230 FOR T=1 TO 460 STEP 2
2240   LET C(c)=A(c)
2250   POKE &E000+T,A(c)
2260   POKE &E000+T+1,B(c)
2270   LET c=c+1
2280 NEXT T
2290 POKE &E000+461,0
2300 ENDPROC
2310 DEFPROC POKE THREE
2320 FOR T=1 TO 460 STEP 2
2330   LET C(T)=A(T)+B(T)
2340 NEXT T
2350 LET c=1
2360 FOR T=1 TO 460 STEP 2
2370   POKE &E000+T,C(c)
2380   POKE &E000+T+1,B(c)
2390   LET c=c+1
2400 NEXT T
2410 POKE &E000+461,0
2420 ENDPROC
2430 DEFPROC DRAW
2440 INK 7
2450 WINDOW 125,249,20,130
2460 PROTECT RED
2470 EXT CLW
2480 LET c=270,d=137+A(1)
2490 MOVE c,d
2500 FOR T=2 TO 230
2510   DRAW c,137+A(T)
2520   LET c=c+1
2530 NEXT T
2540 LET c=270,d=137+B(1)
2550 MOVE c,d
2560 FOR T=2 TO 230
2570   DRAW c,137+B(T)
2580   LET c=c+1
2590 NEXT T
2600 ENDPROC
2610 DEFPROC DRAW THREE
2620 INK 7
2630 WINDOW 125,249,20,130
2640 PROTECT RED
2650 EXT CLW
2660 LET c=270,d=137+(C(1)/2)
2670 MOVE c,d
2680 FOR T=2 TO 230
2690   DRAW c,137+(C(T)/2)
2700   LET c=c+1
2710 NEXT T
2720 LET c=270,d=137+B(1)
2730 MOVE c,d
2740 FOR T=2 TO 230
2750   DRAW c,137+B(T)
2760   LET c=c+1
2770 NEXT T
2780 ENDPROC
2790 DEFPROC POKE WAVE ONE
2800 LET c=1
2810 PROC SET FOR WAVE
2820 FOR T=1 TO 460 STEP 2
2830   LET C(c)=A(c)
2840   POKE &E000+T,A(c)
2850   LET c=c+1
2860 NEXT T
2870 POKE &E000+461,0
2880 ENDPROC
2890 DEFPROC DRAW WAVE ONE (c,d)
2900 INK 7
2910 PROTECT RED
2920 PRINT  @ 15,30;"
```

```
1270 EXT WRESET
1280 REPEAT
1290    IF INP(&0080)<>255 THEN  GOTO 133
0
1300    IF INP(&0880)<>255 THEN  GOTO 133
0
1310    IF INP(&0980)<>255 THEN  GOTO 133
0
1320    GOTO 1290
1330    IF INP(&0080)<>239 THEN  LET d=d-
1
1340    IF INP(&0080)<>223 THEN  LET d=d+
1
1350    IF INP(&0980)<>223 THEN  LET d=d
1360    IF INP(&0880)<>251 THEN  LET c=c-
1,d=d+1
1370    IF INP(&0980)<>253 THEN  LET c=c-
1,d=d-1
1380    IF d<137 THEN  LET d=d+1
1390    IF d>200 THEN  d=d-1
1400    LET c=c+1
1410    LET A(c-20)=d-137
1420    IF A(c-20)>63 THEN  LET A(c-20)=6
3
1430    IF A(c-20)<1 THEN  LET A(c-20)=1
1440    DOT c,d
1450 UNTIL c=250
1460 ENDPROC
1470 DEFPROC TEST SOUND
1480 PAUSE 1000
1490 INK BLUE
1500 PROTECT GREEN+RED
1510 WINDOW 210,240,220,230
1520 PRINT  @ 140,220;"DURATION 1 - 2000
 ";
1530 INPUT D$
1540 IF D$=" " THEN  GOTO 1620
1550 IF VAL(D$)=0 THEN  GOTO 1520
1560 LET D=VAL(D$)
```

Gordon Clay.

```
 ";
2930 WINDOW 125,249,20,130
2940 EXT CLW
2950 LET d=D+A(1)
2960 MOVE c,d
2970 FOR T=2 TO 230
2980   DRAW c,D+A(T)
2990   LET c=c+1
3000 NEXT T
3010 ENDPROC
3020 DEFPROC DRAW WAVE TWO (c,D)
3030 INK 7
3040 WINDOW 125,249,20,130
3050 PROTECT RED
3060 EXT CLW
3070 PRINT  @ 15,170;"
 ";
3080 LET d=D+B(1)
3090 MOVE c,D
3100 FOR T=2 TO 230
3110   DRAW c,D+B(T)
3120   LET c=c+1
3130 NEXT T
3140 ENDPROC
3150 DEFPROC POKE WAVE TWO
3160 LET c=c+1
3170 PROC SET FOR WAVE
3180 FOR T=1 TO 460 STEP 2
3190   LET C(c)=B(c)
3200   POKE &E000+T,B(c)
3210   LET c=c+1
3220 NEXT T
3230 POKE &E000+461,0
3240 ENDPROC
3250 DEFPROC SET FOR WAVE
3260 FOR T=&E000 TO &E000+460
3270   POKE T,1
3280 NEXT T
3290 ENDPROC
```

PRINTER TRANSLATION
****

This article describes a short routine for translating the character values output by the LYNX to different values in cases where the printer requires values to a different standard. It will work for parallel or serial printing (PPRINT or SPRINT), but as given here it assumes that one value translates to one value. If one value needs to be translated to two or more values (or vice versa) then the routine will need modification. The method will also work for terminal communication using the printer ports.

1) Draw up a translation table, LYNX to PRINTER, for all LYNX values 0 to 127 decimal (ie. 128 bytes worth), using hexadecimal values :-

LYNX VALUE	PRINTER VALUE
**********	*************
00	9A
01	37
02	4C
03	18
.	.
20	63
.	.
126	AA
127	FF

Thus in this example, the SPACE character, &20 or 32 decimal is generated by &63 (99 decimal) on the printer.

2) Enter the printer values as a table into memory, where you choose to put it, using the LYNX monitor M command. Note the start address of the table in hexadecimal and save it using the monitor D command or, if you have disks, the EXT MSAVE command.

3) Enter the following Basic and SAVE it :-

```
10 CODE E5 C5 21 yy xx 4F 06 00 09 7E C1
E1 C3 CB 48
20 DPOKE &6202,LCTN(10)
```

Where xx yy are the low and high order bytes of your table start address in hexadecimal. Also note that the last two instructions of code are for the 96K LYNX.

4) RUN. When you use your printer it should now print characters according to the translation table you have set up.

5) NB. Be careful that neither the translation table nor the CODE line gets overwritten. To avoid needing the CODE line you could use the monitor M command to enter the buffer after CODE into memory (eg. just after the translation table -- then you can save it with the table) and then DPOKE &6202 with the address of the first byte of the routine.

6) In Z80 assembler the routine is :-

```
E5          PUSH HL
C5          PUSH BC
21xxyy      LD HL,yyxx
4F          LD C,A
0600        LD B,0
09          ADD HL,BC
7E          LD A,(HL)
C1          POP BC
E1          POP HL
C3CB48      JP NORMLPRINT
```

C.Mathews.

GRAPH PLOT

```
100 PROC GRAPHICS
110 PROTECT 0
120 CCHAR 256*95+32
130 VDU 1,YELLOW,2,0,4
140 PRINT  @ 30,0;"TITLE OF GRAPH ";
150 INPUT L$
160 IF LEN(L$)>12 THEN  GOTO 140
170 PRINT  @ 30,20;"THICKNESS 1 OR 2 ";
180 INPUT a
190 IF a=1 OR a=2 THEN  GOTO 210
200 GOTO 170
210 IF a=1 THEN  LET A$=CHR$(128),B$=" "
220 IF a=2 THEN  LET A$=CHR$(129)+CHR$(1
30),B$="  "
230 PRINT  @ 30,50;"HOW MANY BARS ";
240 IF a=1 THEN  PRINT  @ 70,50;"(MAX 10
)";
250 IF a=2 THEN  PRINT  @ 70,50;"(MAX 8)
";
260 INPUT A
270 IF a=1 AND A>10 THEN  GOTO 230
280 IF a=2 AND A>8 THEN  GOTO 230
290 INK GREEN
300 DIM X(A),C(A),K$(30)(30),P(A)
310 PROTECT MAGENTA
320 FOR W=1 TO A
330   PRINT  @ 10,100;"HEIGHT OF BAR ";W
;" (MAX 20) ";
340   INPUT X(W)
350   IF X(W)>20 THEN  GOTO 330
360   PRINT  @ 10,120;"COLOUR OF BAR ";W
;" ";
370   INPUT C(W)
380   PRINT  @ 10,140;"KEY (MAX 6 LETTER
S) ";
390   INPUT K$(W)
400   IF LEN(K$(W))>6 THEN  GOTO 380
410   FOR d=80 TO 140 STEP 20
420     PRINT  @ 10,d;"
                          ";
430   NEXT d
440 NEXT W
450 PRINT  @ 10,200;"SPACING OF BARS (MA
X 4)";
460 INPUT j
470 IF j>4 THEN  GOTO 450
480 IF a=2 THEN  LET j=j+2
490 PRINT  @ 10,230;"SCALE: 1 BLOCK=";
500 INPUT H$
510 IF LEN(H$)>4 THEN  GOTO 490
520 PROTECT 0
530 INK 7
540 CLS
550 MOVE 25,20
560 DRAW 25,220
570 DRAW 185,220
580 INK 7
590 FOR T=25 TO 190 STEP 10
600   MOVE T,20
610   DRAW T,220
620   INK BLUE
630 NEXT T
640 FOR T=20 TO 210 STEP 10
650   MOVE 25,T
660   DRAW 185,T
670 NEXT T
680 PROTECT BLUE
690 LET E=1
700 LET D=15
710 LET R=X(E)*10
720 PRINT  @ D,222;CHR$(1)(7);E;
730 INK C(E)
740 FOR Q=210 TO 220-R STEP -1
750   PRINT  @ D,Q;A$;
760 NEXT Q
770 IF E=A THEN  GOTO 800
780 LET E=E+1,D=D+4+j
790 GOTO 710
800 LET p=0
810 PAPER 0
820 INK 7
830 FOR P=205 TO 10 STEP -10
840   LET p=p+1
850   IF p<10 THEN  LET Z=6
860   ELSE  LET Z=3
870   PRINT  @ Z,P;p;"-";
880 NEXT P
890 PRINT  @ 0,5;"SCALE";
900 VDU 24
910 INK RED
920 PRINT  @ 30,0;L$;
930 VDU 25
940 LET U=0
950 VDU 1,7,21
960 PRINT  @ 93,30;"KEY"; @ 93,35;"---";
CHR$(20);
970 FOR Y=1 TO A
980   LET U=U+1
990   PRINT  @ 93,(Y*12)+30;U;"=";K$(U);
1000 NEXT Y
1010 INK YELLOW
1020 VDU 21
1030 PRINT  @ 96,195;"SCALE"; @ 96,200;"
-----";
1040 VDU 20
1050 PRINT  @ 93,210;A$;" =";H$
1060 INK YELLOW
1070 PRINT  @ 0,230;"A TO ALTER OR I TO
INPUT NEW GRAPH.";
1080 IF INP(&0680)=253 THEN  RUN 110
1090 IF INP(&0280)=223 THEN  PROC ALTER
1100 GOTO 1080
1110 DEFPROC ALTER
1120 PRINT  @ 0,230;"
                       ";
1130 LET d=15
1140 PRINT  @ 0,230;"WHICH BLOCK 1-";A;
1150 LET v=GETN,v=v-48
1160 IF v=1 THEN  GOTO 1210
1170 IF v>A THEN  GOTO 1150
1180 FOR b=2 TO v
1190   LET d=d+4+j
1200 NEXT b
1210 FOR y=210 TO 220-(X(v)*10)STEP -1
1220   PRINT  @ d,y;B$;
1230 NEXT y
1240 PRINT  @ 0,230;"
                      ";
1250 PRINT  @ 0,230;"HEIGHT 1-20";
1260 INPUT V
1270 IF V>20 THEN  GOTO 1260
1280 LET X(v)=V,w=X(v)*10
1290 PRINT  @ 0,230;"
                      ";
1300 PRINT  @ 0,230;"COLOUR OF BAR ";v;
1310 INPUT C(v)
1320 INK C(v)
1330 FOR Q=210 TO 220-w STEP -1
1340   PRINT  @ d,Q;A$;
1350 NEXT Q
1360 INK YELLOW
1370 PRINT  @ 0,230;"A TO ALTER OR N TO
RUN";
1380 ENDPROC
1390 DEFPROC GRAPHICS
1400 RESERVE HIMEM-30
1410 DPOKE GRAPHIC,HIMEM
1420 FOR J=0 TO 29
1430   READ A
1440   POKE LETTER(128)+J,A
1450 NEXT J
1460 DATA &1E,&2E,&30,&37,&37,&37,&37,&3
7,&37,&37
1470 DATA &3F,&1F,&2F,&37,&38,&3B,&3B,&3
B,&3B,&3B
1480 DATA &30,&38,&3C,&3E,&00,&3E,&3E,&3
E,&3E,&3E
1490 ENDPROC
```

Gordon Clay.

CURSOR CHANGING

As stated in the LYNX USER MANUAL, the cursor can be redefined to anything the user decides upon. Myself, being of the modest type, decided to change the cursor to a monogram of my initials! To do this I first defined my monogram on a 6x10 grid and then, by using a modified version of the program on page 64 of the manual, managed to print it as a character on the screen. I then used the same grid to define the inverse of the monogram and also printed that on the screen. I redefined the cursor itself using the CCHAR command and the ASCII values of my two characters. I also altered the flash rate of the cursor to make the alternating effect greater using the CFR command. To round off the program so that it would not affect any other program that would be loaded I put a NEW command at the end. To alter the characters to your own monogram, all you have to change are the contents of the DATA statements in lines 160 and 170.

```
100 RESERVE HIMEM-30
110 DPOKE GRAPHIC,HIMEM
120 FOR J=0 TO 19
130   READ A
140   POKE LETTER(128)+J,BIN(A)
150 NEXT J
160 DATA 000000,111000,100000,101000,101
111,111100,000100,000100,000111,000000
170 DATA 111111,000111,011111,010111,010
000,000011,111011,111011,111000,111111
180 CLS
190 FOR J=1 TO 9
200   READ A
210   LET A$=A$+CHR$(A)
220 NEXT J
230 DATA 24,1,2,128,129,28,25,1,7
240 CCHAR 128*256+129
250 CFR 1500
260 CLS
270 NEW
```

It is advisable to leave line 270 out until the program is fully working, otherwise you will have to keep on typing in the whole program!! The cursor will not change after running the program until it is either redefined by other software or the LYNX is turned off, but the flash rate can be altered at any time by the CFR instruction. Happy flashing!!
G.CHRISTOFI

HOUSE

This is rather a novel program for fast drawing and is supplied by one of our more recent members.

```
100 DIM A$(100)
110 VDU BLUE,BLUE,RED,BLACK,GREEN
120 PROTECT YELLOW
130 FOR N=0 TO 250 STEP 10
140   MOVE N,0
150   PLOT 3,0,250
160   MOVE 0,N
170   PLOT 3,250,0
180 NEXT N
190 LET A$="90R90R30L90U90L30L90D90U10LA
50R50U40RA50R50D10L"
200 PROC DRAW (YELLOW,20,180)
210 LET A$="40R25D40L25U"
220 PROC DRAW (a,70,90)
230 PROC DRAW (a,70,140)
240 LET A$="30R25D30L25U"
250 PROC DRAW (a,130,90)
260 LET A$="40U20R40D"
270 PROC DRAW (a,135,180)
280 LET A$="15U20R15D15U02L05U05L05D06L0
5U05L05D"
290 PROC DRAW (a,100,40)
900 PRINT  @ 3,200;
999 END
1000 DEFPROC DRAW (a,b,c)
1010 INK RED
1020 PROTECT WHITE-INK
1030 MOVE b,c
1040 FOR n=1 TO LEN(A$) STEP 3
1050   IF ASC(MID$(A$,n,1))=65 THEN  GOT
O 1100
1060   LET d=VAL(MID$(A$,n,2)),e=ASC(MID
$(A$,n+2,1))
1070   PLOT 3,d*(e=82)-d*(e=76),d*(e=68)
-d*(e=85)
1080 NEXT n
1090 ENDPROC
1100 LET d=VAL(MID$(A$,n+1,2)),e=ASC(MID
$(A$,n+3,1)),f=VAL(MID$(A$,n+4,2)),g=ASC
(MID$(A$,n+6,1))
1110 PLOT 3,d*(e=82)-d*(g=68)-f*(g=85)
1120 LET n=n+4
1130 GOTO 1080
```

TANDY COLOUR PLOT

Yet another ! This routine provides a colour screen dump to the TANDY.

```
10 PROTECT 0
20 CLS
30 LPRINT CHR$(18)
40 PRINT  @ 0,0;CHR$(1)(1);"HE";CHR$(1)(
2);"LL";CHR$(1)(4);"O"
50 LPRINT "C1"
60 PROC CALL(&0069,&A000)
70 LPRINT "C"
80 PROC CALL(&0070,&C000)
90 LPRINT "C3"
100 PROC CALL(&0069,&C000)
110 LPRINT "A"
120 STOP
130 DEFPROC CALL(C,D)
140 FOR Y=0 TO 10
150   FOR X=0 TO 5
160     CALL C,D+Y*32+X
170     LET A=HL
180     PROC P(A>127)
190     LET A=A-128*(A>127)
200     PROC P(A>63)
210     LET A=A-64*(A>63)
220     PROC P(A>31)
230     LET A=A-32*(A>31)
240     PROC P(A>15)
250     LET A=A-16*(A>15)
260     PROC P(A>7)
270     LET A=A-8*(A>7)
280     PROC P(A>3)
290     LET A=A-4*(A>3)
300     PROC P(A>1)
310     LET A=A-2*(A>1)
320     PROC P(A>0)
330   NEXT X
340   LPRINT "R-96,-2"
350 NEXT Y
360 LPRINT "H"
370 ENDPROC
380 DEFPROC P(B)
390 IF B THEN  GOTO 420
400 LPRINT "R2,0"
410 LPRINT "J2,0"
420 LPRINT "J-2,-1"
430 LPRINT "J2,0"
440 LPRINT "J0,1"
450 ENDPROC
```

AUTO-LOADING CP/M on the 128K
****

Boot up CP/M as normal using a copy of a system disk. Then follow this procedure:-

```
A>DDT SYSGEN.COM  <RET>    (Running SYSGEN under DDT
DDT VERS 2.2               control).
NEXT  PC
0600  0100
-S5D
005D  53 20       <RET>    (Clear the command line).
005E  59 .        <RET>
-G100,0                     (Execute SYSGEN the break
                back to DDT).
Source Drive name, (or return to skip) A
Source on A, then type return. <RET>
                           (Read system tracks).
‡‡Function complete‡‡
Dest. Drive name,(or return to reboot) <RET>
                           (Break to DDT)
‡0000                      (The system is now in
                           memory starting at &B00)
-ICOMMANDLINE     <RET>    (Command line is usually
                           a file name eg. DDT, but
                           the command must be less
                           than 16 characters)
-FB08,B18,20      <RET>    (Clear CCP's command line)
-M5D,6C,B08       <RET>    (Move command into the
                           CCP)
-SB07             <RET>    (Set No. of letters in
                           command line.)
0B07  00 XX       <RET     (XX=length of the command
                           line)
0B08  ‡‡.         <RET
-S5D
005D  ‡‡ 20       <RET     (Set Clear)
005E  ‡‡ .        <RET
-S2138
2138  01 3        <RET     (Set the auto flag in the
                           BIOS)
2139  ‡‡ .        <RET
-G100                      (Now complete SYSGEN)
Source Drive name, (or return to skip) <RET>
                           (Already done)
Destination Drive name, (or return to reboot) A
Destination on A, then type return  <RET>
                           (Write SYSTEM back)
‡‡Function complete‡‡
Dest. Drive name, (or return to reboot) <RET>
                           (All done)
```

Note, ‡‡ denotes don't care.

However, having done all of this, my version of CP/M 2.2 is set up to execute a SUBMIT provided the AUTOFLAG at &E738 is set. This provides a more powerful, but less secure procedure. To simply set this flag, follow this procedure above but don't touch the CCP command line. Then write a suitable SUBMIT.SUB file to execute your command(s).

Incidently it's much easier to use DISKED to implement these mods. On a 200K disk, the start of the CCP is on sector 2 of track 0, and start of BIOS on sector 3 of track 1.
Ray Albone.

ELLIPSE ROUTINE
****

When listing Forth routines, it's a good idea to leave a space on the right hand side of the page for notes. It is often useful to make a note of the stack effects there. I also like to draw a line connecting each DO to the corresponding LOOP and each IF to its ELSE and THEN to check that the structure is correct. When entering the routine however you start at the beginning of the line and type through to the end. There is no need to enter the comments (the bits in brackets).

I believe the general formula for an ellipse is:-

$$X^2\left(\frac{a^2-u^2}{a^2}\right)+2.X.Y\left(\frac{u.v}{a^2}\right)+Y^2\left(\frac{a^2-v^2}{a^2}\right)+u^2+v^2=a^2$$

This quadratic equation can be solved by the well-known formula:-

$$X = \frac{(-B \pm SQR(B^2- 4.A.C))}{2.A}$$

which is what the ellipse routine does. I have chosen the names of the subroutines to illustrate this. B -4AC couldn't be evaluated all in one go, in fact the 3 "EXP" words had to be designed quite carefully to keep their values within the range of unsigned single-length integer numbers (any value less than one becomes zero in Forth so you have to avoid that pitfall too!!). I haven't learnt to use double length numbers yet!

PROGRAM
****

```
: VAR   VARIABLE ; (To save space)
0 VAR a 0 VAR u 0 VAR v 0 VAR x0 0 VAR y0 0 VAR Y
: PARAMS (x , y , x , y , a --) (co-ordinates
          of the 2 focii and major radius)
                   a ! ROT SWAP 2DUP
                   + 2 / y0 ! - 2 / v !
                   2DUP + 2 / x0 !
                   SWAP - 2 / u !      ;
: A       a @ DUP ‡ DUP u @ DUP ‡ - SWAP ;
: EXP1  u @ DUP ‡ v @ DUP ‡ a @ DUP ‡ ‡/ 4 ‡ ;
        (note ‡/ is a single word - NO space)
: EXP2  a @ DUP ‡ v @ DUP ‡ - A ‡/ 4 ‡   ;
: EXP3  a @ DUP ‡ u @ DUP ‡ - v @ DUP ‡ -
        A ‡/ 4 ‡                         ;
: BB-4AC  EXP1 EXP2 - Y @ DUP ‡ a @ DUP ‡
          ‡/ EXP3 +                      ;
: -B    u @ v @ ‡ Y @ -2 ‡ a @ DUP ‡ ‡/ ;
: 1/2A‡  a @ DUP ‡ DUP u @ DUP ‡ - 2 ‡ ‡/ ;
: ELLIPSE ( x ,y , x ,y , a -- )
                   PARAMS a @ DUP 1+ MINUS DO
                   I Y ! BB-4AC DUP 0< IF
                                 DROP ELSE
                   SQR DUP -B DUP
                   ROT - 1/2A‡ x0 @ +
                   ROT ROT + 1/2A‡ x0 @ +
                   Y @ y0 @ + MOVE Y @ y0 @ +
                   DRAW           THEN
                                     LOOP ;
```

Note. SQR as defined in Issue 1.
A.L. SHAW

RECLAIM ROUTINE
****

It is sometimes desirable to reclaim the space taken by the use of the RESERVE command. This routine will move the stack to a higher memory location and is basically the opposite of the RESERVE ROM routine. It is relocatable and may be placed in a CODE line. The DE register must be primed with with the HIMEM value required, in this example &F51E to allow for the disk routines of a 96K LYNX.

```
...0 01 00 01  LD BC,&0100          ..17     EB          EX DE,HL
...3 11 1E F5  LD DE,&F51E          ..18     2A EE 61    LD HL,(&61EE)
...6 2A EE 61  LD HL,(&61EE)        ..21     19          ADD HL,DE
...9 23         INC HL              ..22     22 EE 61    LD (&61EE),HL
..10 23         INC HL              ..25     EB          EX DE,HL
..11 ED B8      LDDR                ..26     39          ADD HL,SP
..13 B7         OR A                ..27     F9          LD SP,HL
..14 EB         EX DE,HL            ..28     C9          RET
..15 ED 52      SBC HL,DE
```

K R Cooper.

CAMFORTH ERROR MESSAGES

The FIG-FORTH standard error messages are given below, with notes on their usage in Camsoft FORTH.

MSG0 UNDEFINED. The offending word is not in the current vocabulary and is not a number. Used by COMPILE, NUMBER and ' (tick).

MSG1 EMPTY STACK. Used by INTERPRET. The error message only appears after the instruction that causes the stack-empty has been executed, i.e. after your program has run to completion.
Not very helpful!

MSG2 DICTIONARY FULL. Not implemented in Camsoft FORTH.

MSG3 INCORRECT ADDRESS MODE Whatever that may mean,not implemented.

MSG4 NOT UNIQUE. You already have a definition of the same name in the current vocabulary. Used by CREATE (called by :), does not call ABORT. This means you can ignore the error message if you wish and compile two or more definitions with the same name.

MSG6 DISK RANGE ?. The address of the disk block you have asked for is outside the range available, specified by LO and HI (see memory map). The error check is performed by R/W which is called by many words.Wrong arguments for LIST and LOAD are the usual sources of error.

MSG7 FULL STACK. Like MSG1,used by INTERPRET and again, only checked after execution of all instructions in the input message buffer. The stack is deemed full if the top of the stack is within 128 bytes of the top of the dictionary. In practice what tends to happen is that a rogue routine fills up the stack and then overwrites part of the dictionary, preventing further instructions being processed.The system then hangs and you never see the error message to know what went wrong. (Bitter experience!).

MSG8 DISK ERROR. Not implemented.

MSG9 Screen 0 is reserved for comments in this version. The phrase 0 LOAD produces the error message instead of compiling your code. This one is not in the FIG-FORTH standard.

MSG17 COMPILATION ONLY, USE IN DEFINITION. This reminds you that compiling words, like BEGIN ENDIF,CASE etc,can only be used within a colon-definition.

MSG18 EXECUTION ONLY. Used by : and X, these can't be used inside a definition.

MSG19 CONDITIONALS NOT PAIRED. Used by compiling words such as LOOP, UNTIL, AGAIN, ENDOF etc to indicate structures that are not correctly nested. For example:- :NONSENSE BEGIN 1 IF AGAIN THEN ; would produce the response AGAIN ? MSG h19 and NONSENSE would not be compiled; whereas : RUBBISH BEGIN 1 IF THEN AGAIN ; is perfectly acceptable to the compiler - it doesn't check whether your program makes sense, as long as it is structured correctly.

MSG20 DEFINITION NOT FINISHED. Used by ; and ;CODE to detect omissions of structuring words. E.g. : GARBAGE BEGIN 1 IF THEN ; would be rejected. BEGIN has opened a structure line which needs the corresponding UNTIL, AGAIN or REPEAT to close it.

MSG21 IN PROTECTED DICTIONARY. You are trying to FORGET something protected by the FENCE.

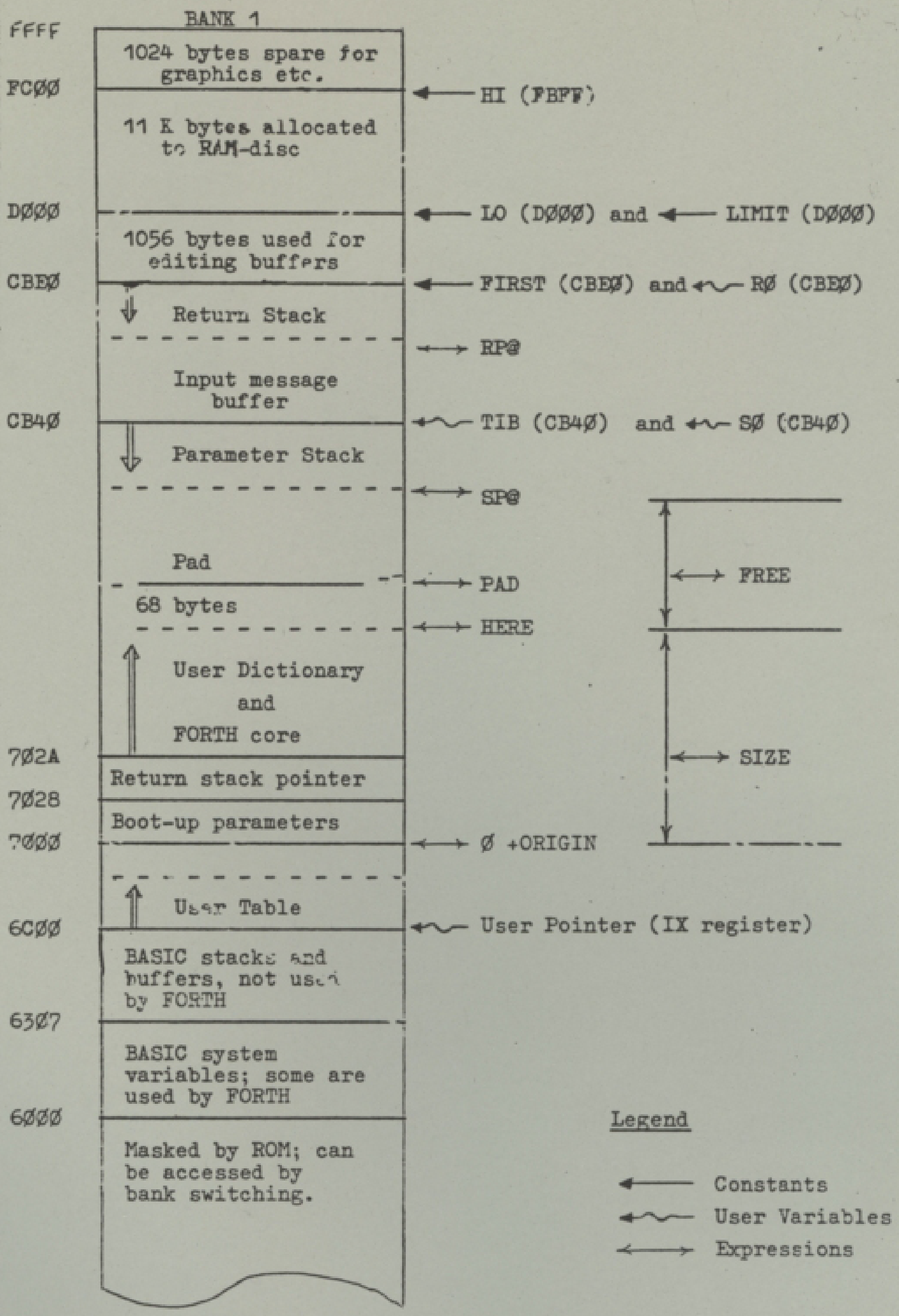
MSG22 USE ONLY WHEN LOADING. Used by --> (next-screen) to continue the LOAD instruction on to the next screen; would be meaningless in any other context.

MSG23 OFF CURRENT EDITING SCREEN. Used by LINE - indicates user has specified a line number outside the range 0-15. 16 lines per screen is the arrangement in most FORTH systems.

MSG24 DECLARE VOCABULARY. Used by FORGET, indicates current and context vocabularies are not the same. State which you want to FORGET from. The choice is FORTH or EDITOR, unless you've created any other vocabularies.

A.L.SHAW.

CAMSOFT FORTH MEMORY MAP FOR THE 96K LYNX



BANK 1
FFFF
1024 bytes spare for graphics etc.
FCØØ
HI (FBFF)
11 K bytes allocated to RAM-disc
DØØØ
LO (DØØØ) and LIMIT (DØØØ)
1056 bytes used for editing buffers
CBEØ
FIRST (CBEØ) and RØ (CBEØ)
Return Stack
RP@
Input message buffer
CB4Ø
TIB (CB4Ø) and SØ (CB4Ø)
Parameter Stack
SP@
Pad
PAD
68 bytes
HERE
FREE
User Dictionary and FORTH core
SIZE
7Ø2A
Return stack pointer
7Ø28
Boot-up parameters
7ØØØ
Ø +ORIGIN
User Table
6CØØ
User Pointer (IX register)
BASIC stacks and buffers, not used by FORTH
63Ø7
BASIC system variables; some are used by FORTH
6ØØØ
Masked by ROM; can be accessed by bank switching.
Legend
Constants
User Variables
Expressions

LYNX ADVANCED MANUAL
****

PRELIMINARY NOTICE
==================

As has been seen at recent shows, the LYNX ADVANCED MANUAL is now close to readiness for printing. It will however be a limited edition and reprints will not be undertaken. So if you are interested in obtaining a copy, then you have only one chance. Currently at the time of this notice, Nov '89, there is still some work to be done in two areas:

Certain corrections and additions to existing chapters and,

completion of several diagrams and illustrations pertaining to the 96K.

As it will be limited, some confirmation is required now concerning the quantity needed. For this purpose, I am defining a decision period, both for UK and members abroad. If a minimum requirement is not reached, then I can only assume that the overall interest is not present, so the manual will not be printed.

The "DECIDE BY" date is 31 Mar '90.

If you decide to have a copy and would like to know whether the interest has been adequate, either 'phone or send an SAE for a reply. The estimated price for the manual is £30.00 incl. or £32.00 Sterling for sending abroad.

The Manual is in "A5 ring-binder" format, allowing for either corrections or additions or even your own notes!

Chapters include: KEYBOARD info, DISK OPERATING SYSTEM, DATA STORE MANIPULATING, DUMB TERMINAL EMULATION, BASICs CONVERSIONS, 96K DETAILED HARDWARE ANALYSIS,

appendices include: COMPLETE TOKEN TABLES (96K), NUMERIC Z80 LISTINGS, DIY DISK MAP, and KEYBOARD DATA TABLES.