NILUG NEWS

NATIONAL INDEPENDENT LYNX USER GROUP

Magazine for **LYNX** Users

Volume 1. Issue 6.

NILUG NEWS

VOLUME 1. ISSUE 6.

EDITORIAL

This is the last Issue of NIULG NEWS. No I don't mean the last Issue of Volume 1 I mean the last Issue of NILUG NEWS. Unfortunately NILUG has not had the support it needs and hence I am going to close it. I had hoped that NILUG would get a thousand members in the first six months. Here we are a year later and membership is 500+. Its just not enough for me to consider a second year. Everything hinges on membership. It is as easy to produce a mag for a thousand members as it is for 500 but the economics are vastly different. Secondly the more members you have the more chance there is that you will get copy. I must say I have had a fair amount of copy but very little of it has been suitable for immediate publication. The vast majority needs more than a little work.

The final deciding factor is that I feel I can't go into the second year of NILUG without supporting the 128K Lynx and disks. In effect I would have to pay about £1000 out of my pocket to support the group - thats a subsidy of £2.00 per member.

I am pleased to be able to tell you that another Lynx User Group is going to start and you will find an application form enclosed. If NILUG couldn't succeed why should the new group do any better? There are two main factors. The first is that LUG is not starting from scratch as NILUG did. The second is that it is going to have a committee to run it and share the work. As I understand it the mag is going to be similar to NILUG NEWS so I hope you will support LUG.

Some members still have outstanding subscriptions. UK members will be paid off (£1.50 per outstanding issue) in the coming month. If you would prefer to have any back issues to form a complete set of NILUG NEWS please write and let me know.

Unfortunately the Bank charges associated with overseas payment prevent me from refunding small amounts of money. Overseas members will have their subscriptions transferred to LUG. I feel that this is the best I can do. Fortunately it only affects about a dozen members.

I would like to express my thanks to all of you who took the trouble to write to me. I am sorry that I couldn't help all of you.

Last but not least I would to thank my mother for proof-reading all the copies of NILUG NEWS. I couldn't have had a better proof-reader. I bet your mother wouldn't have spotted that 'PBC' should have been 'PCB' - mine did along with a lot of other mistakes that I made. Many many thanks mum.

R.B.Poate
EDITOR.

COMMUNICATIONS

5.Acacia Road, Redstock, Bath, Avon

Dear Mr Poate,

I teach science in a secondary school and spend quite a lot of time writing science programs on my Lynx for use in school. Some of these are for use by the teacher, eg an explanation of electrolysis for fourth year Chemistry, the production of hydrogen spectrum for sixth form Chemistry and Physics, and a genetics program for fifth year Biology. Others are for pupil use eg. practice at writing chemical formulas, practice at identifying inorganic and organic chemicals, ray diagrams for convex lenses, and a balancing see-saw program for the first years.

Whilst these have not had the rigid testing which may be necessary for fully commercial programs, they have been used and criticised by other science teachers and by pupils in school.

If anyone is interested in finding out more about these programs I would be happy to send details on receipt of a stamped addressed envelope.

Yours faithfully, Linda Hencher (Miss).

The following members would like to contact local users:-
Mr A.R. Bristow 22,Lawrence Rd.,Tilehurst,Reading,RG3 6BH
Stuart Higgins 97, Green Lane, Eastwood, Leigh-on-Sea,ESSEX. SS9 5QU

Are all NILUG members receiving Lynx User? If not then send your name and address to Rebecca Hamsley at Camputers.

REVIEWS

Snowball by Level 9; Price £9.99 By Alan Hutchison

This is the fourth adventure to come from the Level 9 stable and once again they have managed to achieve what must be 'state of the art' adventure programming. Most people must have heard of Level 9's previous adventures, which are famous for cramming over 200 locations, each wonderfully described in superb detail, into a relatively small 32K of program space. Well this time they have managed to include over 7000 locations, YES 7000! again with marvellous lengthy descriptions to really set the mood.

The object of the adventure is as follows:- You take the role of Kim Kimberly, a space agent, whose mission is to see the giant transporter Snowball IX safely arrive on the new space colony called Eden. Unfortunately something has gone wrong, and you are awoken from your hibernating state to return the ship to a safe operating state.

You find yourself in a level of one of the many passenger disks and you must use your wits and logic to find your way to the control room on the five mile long ship. Many problems are encountered the first being the huge "nightingale" guavctions (? I couldn't read the writing Ed.) who patrol the ship and destroy anything that moves. This was a problem that had me stumped, but once again, when the solution is discovered it is so obvious you kick yourself. I am still trying to get out of the huge passenger disk, but I have managed to use the lifts, and get access to all the levels on the disk. Seemingly a spacesuit must be worn to proceed further, but I have yet to find it.

The documentaion included is superb, and includes a full background history to the game and also a "taster" for the forthcoming sequel to this adventure "Return to Eden". Level 9's programmers can only be congratulated for this superb implementation on the Lynx micro, involving themselves in complicated bank-switching techniques to get a full 32K out of the micro, a task most software houses would have a nightmare even thinking about.

To sum up, if you like adventures buy it. If you don't like them, buy this one, it may change your mind.

Grid Attack by Shards Software; Price £4.95 By David Shaw.

This program is a mixture of BASIC and machine code but is nevertheless a fast game. It bears a resemblance to other games of the grid variety available on other machines. You must steer your light beam around the grid collecting energy pods, but you must avoid the grid mines, your own trail and also the computer controlled laser tracker which is hell bent on catching you.

By collecting all of the energy pods you move onto the next levels which become increasingly harder.

This program makes good use of the computer sound capabilities, colour and graphics and should keep the games player amused for many hours.

One small gripe, I would like to have seen a joystick option in this otherwise good program. Value for money seven and a half out of ten.

Atom Smasher from Romik Price £9.99 By David Shaw.

This is one of four programs offered by Romik. The program loads quickly and is entirely in machine code. On loading you get a title page giving the nine skill levels, it also asks if you wish to use a keyboard or joystick.

From here you launch into the game - a nuclear reactor. You have to shoot protons to slow down the melt-down of the reactor, but shooting a proton releases an electron which must be dodged. Shooting an electron speeds up the game and melt-down. The game is extremely fast with lots of things happening on the screen complete with many sound effects.

At £9.99 (available mail order from Romik) this software is not cheap, but this arcade standard game is one that every Lynx owner should have, if only to see what speed is possible with the Lynx.

I think this software deserves nine out of ten and I hope Romik will consider writing some more excellent software to supplement their four present titles.

Lynx Disks By Martin Livesey

First the disk - this is a smart half-height drive in a metal case painted the same colour as the Lynx. It is relatively deep since the PSU is located behind the drive mechanism and the large heatsink at the back becomes almost as alarmingly hot as the Lynx PSU. The drive connects to the Lynx via a 26-way ribbon cable and the disk interface box. The interface box is an external ROM sized box but without the on/off switch and slots into the parallel expansion port of the Lynx "piggy-back" style with the joystick interface.

Once powered-up (disk first!) the external DOS ROM must be activated with the XROM command. The reason that only the 96K RAM Lynxes and larger can use the disk now becomes apparent, the ROM is copied into the data store RAM and high RAM moving down the stack pointer. The 48k does not contain the data store RAM and hence can't be used. An initialisation message "Lynx DOS Vers. 1.0 HIMEM=&F51E" appears on the screen and the disk operates for a second or so. All disk commands are preceded by EXT (yuk) and there are nineteen of them, which offer basically the same functions as for tape storage, some new disk management commands and a few extras of varying use. Programs, data and machine code can be loaded and saved in much the same way as by tape including appending a BASIC program. The disk management commands provided are:-

FORMAT - Sets up a disk for use.
CHECK - Checks whether a disk is formatted.
DIR - Displays the current default drive directory
DRIVE - Changes the current default drive (A-D).
ERASE - Removes a specified file from the drive.
UNERASE - Replaces an erased file (if possible).
RENAME - Changes the name of a specified file to a different specified name.

Now for the others.

INFO - Displays information relating to a specified file, (the file type, lock status, no of blocks occupied, load address for MC programs, execution address for MC programs, and displacement from start of BASIC for auto run BASIC programs, the size in bytes of the file and the file comment.)

COMMENT - Allows a predefinded comment to be added to all files.

LOCK - A software read and/or write and/or erase protection.

BOOT - Boots Lynx from current default drive (main use for booting CP/M).

The manual provided with the drive (should be with

the controller) was only provisional and apart from the essential information and a list of error codes provided very little extra information. The DOS also has a few shortcomings possibly allowing CP/M to fill in the gaps here

There is no wildcard specifiers (as far as I know) when specifiying filenames which means that when erasing a large number of similar programs for example requires a program or a fair amount of time.

There are no sequential or random access files for storing data in a reasonably accessibly form without having to resort to data store juggling.

Overall fair value for money but a bit on the pricey side.

Joystick Interface. Price £14.95 By Martin Livesely

The joystick interface is packaged in the same matching grey plastic box as the disk controller and fits into the rear expansion socket. Any Atari compatible joystick will fit themselves into two sockets on either side of the box. The software to run the joysticks (extended ROM required) from BASIC gives a number between between one and eight rather like the numbers on a clock. 1 for top right, 2 for right, round to 8 for top. The leaflet provided with the interface gives enough information to access the joysticks from BASIC using the INP command. The pin outs are given to allow the interface to be used as a general input device with the correct D-type connector. Overall good value due to a reasonable amount of information.

YNXVADERS from BUSTECH price £7.00 By A.C. Karsten

The game loads on tape 0 in about one minute. Then you get a picture of the different invader types and the points you score for blasting them out. There's no on-screen explanation only a brief explanation on the cassette leaflet Hitting the spacebar starts the game. It looks similar to the original arcade game. Only the invaders are a bit smaller. There are a few bugs in the program.

First: sometimes your bullets stop.

Second: the high score facility doesn't work. It changes between 4966 and 4995. This is very high and I think impossible to get to. You would have to destroy the invaders five times and they start lower every time.

Third: after you destroy all invaders you start all over again. But the first ten seconds everything on the screen works very slowly.

Despite the few bugs I enjoyed the game for many hours. It has nice graphics and bearable sounds. 7 OUT OF 10

GAMES VOL1 from WILLOWSOFT £4.50 By A.C. Karsten.

Four games for just over one pound each,that is quite cheap so I bought this one. Loading is on TAPE 0 and worked first time. There is no protection in the games so you can resave them at TAPE 5 for faster loading. The first game is 'TOWER OF RINGS' a classic thinking game usually called 'TOWERS OF HANOI' There are three towers and 7 rings of different sizes. You move the rings from tower 1 to tower 3 so that no 'big' ring comes above a smaller one. This is quite difficult and needs more than 500 moves of the rings.

The second game is MASTERMIND also a well known thinking game you have to guess four colours in the right sequence. The computer tells you whether they were in the right position, the right colour or neither of them. The screen layout is very nice. With some clever deducing and some luck you will be able to guess the right combination within the allowed seven turns.

The third game is 'BLOCKER' the best of the four I think. You control a line which grows in the direction you want. You have to lead this line to coloured blocks and so score points. The blocks have different values for each colour. You may not hit your own line or the borders. The speed of your line increases and the coloured blocks flash away sometimes. There is a high score.

The fourth game is 'YAHTZEE'. This game has instructions on the cassette leaflet. The other three have instructions in the game which I prefer. This is a dice game for one to five players. You have to form certain combinations with the dice. You have three tries for each combination. Each is scored in a different way and sometimes you get a bonus. The games on this cassette are usually sold separately for other machines and for a higher price than this one SO VALUE FOR MONEY 10 OUT OF 10.

WORD PROCESSOR from Camsoft Price £24.95 By M. Cheetham

Released by Camsoft for the 48K and 96K Lynx, at £24.95 this word processor package is remarkably good value. Written in 3K of machine code it leaves more than 10K of memory available for the text in the 48K Lynx and more than 34K in the 96K. As usual with Camsoft programs it is recorded at two speeds - TAPE 3 and TAPE 5.

Most of the main word processing features are available except for on-screen formatting due to the obvious limitations imposed by a 40-column screen. Not that this proves a great hardship as the on-screen formatting markers are easy to understand.

Text may be centered, ranged to the right, justified to the right-hand margin, or just plain printed out. A tabulation command provides for fixed indentation of paragraphs. Block move, erase and insert are all provided along with a powerful search and amend facility. Printer control codes may be inserted in the text but care must be taken when using condensed or enlarged print as it is likely to upset the formatting routines.

Good printer control-including line length (up to 126 characters), margin width, number of lines to each page, line formatting as previously described, and automatic perforation skip, is provided. There are additional commands for saving and loading blocks of text to and from tape, either to work on at a later date, keep for reference, or to append to other texts.

Bearing in mind the cost and small size of the package, the programmers have done a masterly job. The only drawback is due more to the screen display of the Lynx than the program. Given the easy text review facilities and having mastered the extensive editing facilities, this ceases to be a problem. The error trapping provided is adequate, the only weak point being with the cassette reading command. If you enter an incorrect name and find yourself trapped in the load routine, the act of escaping will corrupt the program. This can also also happen if you forget to place quotes around the name in the save command and then try to read or verify it. Instructions on how to recover from this potentially catastrophic situation are detailed in the very good documentation making it more of a nuisance than a disaster.

This package is not "Wordstar". It doesn't purport to be. With a little care and practice, very professional results can be easily obtained. All things considered, it is a very good implementation of a basic word processor, and one that I have no hesitation in recommending.

Siege Attack from Quazar Price £5.95 By Simon Brookes

This is an all machine code game based on Thorn EMI Video's 'ORC ATTACK' for Atari machines. It loads in about three minutes at TAPE 0 with no difficulties, and is keyboard or joystick controllable.

You are the lone defender of the city walls, guarding against attackers who climb up the wall with scaling ladders only to be knocked down again when you drop boulders on them. The title page is a nice piece of animated graphics with your 'man' walking across the battlements and dropping letters down to form the title. Nice but hardly original, rather reminiscent of ancient PET space invader titles.

You are then presented with the options to select keyboard or joystick, read instructions and select level of difficulty (1-3). The game then begins, you run to either end of the wall to collect one of the limitless supply of boulders while a few attackers place their ladders down and start to climb them. You then position your man over an attacker and drop your boulder on it by pressing the space bar, this knocks the attacker off and you score 10 points.

As time goes by the number of attackers and their speed increases. When you reach 500 points a rock thrower appears in the bottom left hand corner and promptly starts to catapult rocks at you. If hit, these stun rather than kill you, giving the attackers an extra chance to reach the top of the wall and take one of your three initial lives.

The game proceeds and at 1000 points another rock thrower appears but in the opposite corner, when you reach 2000,3000 ... points you gain extra lives. When your three lives are lost the game goes back to the title page and starts all over again. Having to watch the title page over and over again however is very, very annoying.

In conclusion then, a good game that soon loses its appeal, due to seeing the title page again and again as much as anything. It is sadly lacking on TEV's original, but has colourful graphics and reasonable sound nevertheless. Seven out of ten.

Superchess II from CP Software Price £8.95 By Eric Morris

Superchess II by C.P. Software is described on the inlay card as currently the strongest chess program for the LYNX. Since it is, as far as I know, the only chess playing program available this claim might seem to be empty. This however is not so it is indeed a powerful program and worthy of the attention of all chess playing LYNX owners.

All legal moves, including castling and pawn capture en passant, are accepted. Pawns are promoted to queens on reaching the back rank. The program can give recommended moves or play against itself if S is pressed after the program has made its normal move. Chess problems can be set up and played through.

The board and piece display is first class with the LYNX colour graphics. The move history is displayed at the side in algebraic notation. Instructions are printed on the fold-out inlay card, a very convenient way of ensuring that they are always available.

Seven levels of play (0 to 6) are available. The highest levels are beyond my own chess capacity so I cannot comment on their strength but the response time seems to increase to hours so these levels may be of limited use. Level 2 gives a reasonable game and gives a response in a few minutes.

LYNX 128K By Kym Wilson

The Lynx 128K machine has the same appearance as the 48K or 96K model from the outside. Only the 128 on the Lynx badge indicates that it is the latest model in the Lynx range. However if you open it up, several changes inside are apparent. First of all the board has been redesigned. The new board gives the impression that it's been designed rather than thrown together and there are no "last minute add-ons". The CPU has been upgraded as well as the 128 has a Z80B running at 6MHz. When the machine is switched on another difference is seen. At last we have a full 80 columns and the display looks like it belongs to a real computer.

The 128K of RAM is made up of 64k user RAM in bank 1 and a full bank of 64K video RAM in bank 2. The video RAM consists of four blocks of 16k for each colour. This rearrangement of the video RAM means that any 96K programs that accesses the screen directly will not work on the 128K model. The extra video RAM gives a graphics resolution of 512 by 256 pixels of text. The latter is needed for CP/M and for serious applications such as word processing or spreadsheets. The 80 columns can be read with some difficulty (and eye strain) on a television set but really a monitor is needed. The 128K can emulate the 96K screen in both text and graphics modes. The new VDU commands No. 26 and 27 change text mode and LOW ON/OFF the graphics. A new "colour" has been added, No 8 which corresponds to alternate green and VDU 11 is used to switch the display between the two greens.

VDU 3 has been implemented as PROTECT with the same format as VDU 1 and 2. The graphics commands have been changed as follows. The high res. screen is considered to be 512x512 pixels, although the actual vertical resolution is only half that number. This is for MOVE, DRAW DOT, CIRCLE and TRIANGLE commands. PRINT@ and WINDOW still work on half the number of horizontal positions. An extra command FILL has been added. This defines the bit pattern for filled triangles, circles and rectangles (using CLW). The ink and paper colours are assigned according to the 0s and 1s in the parameter of the FILL comand. For example &5555 would alternate ink and paper (binary 01010101 01010101). This gives rise to a whole set of new colours since you can effectively put two pixels in the place of a single one in the 96K screen. These include orange and several shades of red, green and blue.

The BREAK key can be enabled and pressing ESC on BREAK at the same time acts as a warm boot. All programs etc are lost but at least you no longer need to switch the Lynx off and on again after its hung.

The manual is mostly a copy of the 48K manual with most of the errors corrected and a whole lot of new ones added. Some more explanation of the USER commands and SOUND is given this time and there is informaton on bank switching, the 6845 CRT controller and interrupts in the hardware section.

Some ROM addreses have been changed and this may also result in programs which were written for the 48/96 machines not working on the 128. I find that I cannot read any of my old tapes in any case. I had a problem reading commercial software on my 96K. Now thats all I can read!

The micro press led me to believe that CP/M would be included with the 128K model. However this was not the case when I bought my upgrade although it may change in the future.

In summary, the 128 takes the Lynx into the serious business class at last. The higher resolution graphics and 80 columns of text, CP/M and the faster Z80B CPU give this machine some of the best specifications in the price range. Be warned, however, that existing software will probably not run on the new model.

PUZZLE PACK from QUAZAR Price £4.75 By R.B.Poate

This tape has four type programs on it. The first is PYTHONS and as its name suggests is a game in which you steer a python around the screen. You must avoid the walls, your own trail and the Lynx's python. Sounds simple but not for the likes of me. Just to complicate the issue red prowling stray cats appear at random. The graphics are reasonable especially the end when you or the Lynx has lost.

Unfortunately I couldn't load the second game MATCH IT. I suspect the tape because I had no problem loading the other three games (or any of Quazar's other tapes for that matter) . However I did manage to LIST the program (although all the programs are protected they are not protected very well!). The idea is that the Lynx will flash a colour at you and you have to press the initial letter of the colour. Then the Lynx will flash two colours at you, then three etc. The game ends when you get the sequence wrong.

In INSIGHT you choose a 4 by 4 template with holes cut in it and the Lynx does likewise. You then take it in turns to place a marker on a 4 by 4 grid, your markers are white the Lynx uses black. The aim is to put your template on the grid such that three of your colours show and one of your opponents. I rather liked this game. It really makes you think in 3D.

MUFTYMIND is a variation of Master Mind. The Lynx will generate a secret number combination which you have to determine. You may make a guess at the number and you are given a red marker if you have correctly put one of the digits in the correct place or a white marker for a correct digit in the wrong place.

The games/puzzles are nothing spectacular but I quite enjoyed them. Overall good value for money at £4.75 for four games (10% discount for NILUG members).

Centipede from PLAY IT Price By R.B.Poate

Once in a while a really good game comes in for review. Centipede from PLAY IT is just such a game. It is a Lynx version of the familiar Centipede and mushroom game where you try and zap everything which moves and everything which doesn't. If you hit the centipede it breaks up into two (or more) separate centipedes which continue to descend upon you.

The first thing that strikes you about this game is that you will think that your colour TV is on the blink. You then realise that the characters on the screen are multi-coloured. This gives them a very classy sheen and greatly

enhances the appearance of the game. The game starts with alternating HI-SCORE table and instructions. Fortunately there is no music (on these two screens) to drive you crazy.

The game starts when you press the FIRE control (either keyboard or joystick). Then there is a short tune and the game starts with amazing speed and sounds. If you hit the centipede the zapped segment turns into a mushroom which may also be destroyed to gain more points. Just to add more fun there is a spider like object which flies in a zig-zag manner across the bottom of the screen. This is normally the thing which gets me. The final touch is a snail-like object which lays green mushrooms across the screen.

To conclude, if you want a good game for your Lynx this may be it. I couldn't stop playing with it and since I don't normally play games that is quite a compliment. I would like to give it ten out of ten but I am reserving that so it will have to settle for nine and a half. An excellent piece of software.

FORTH from CUBSOFT Price £15.95 By R.B.Poate

The first draft of this review was easy to write. After all I have written several reviews over the last year. However when it was finished I realised that I had slipped in so many Forth technicalities that I doubted whether any of you would understand what I had written. Consequently this is going to be a be more of an explanation of Forth than a review.

In case you don't know Forth is a computer language. Informed people tend to either love or hate Forth. On the plus side Forth is very compact, very fast, is structured, extendable and highly portable. On the minus side Forth is hard to read, it uses reverse Polish notation, it encourages programmers to use 'tricks' and it lacks many programming constructs such as arrays, strings and floating-point numbers.

The hub of Forth is called a dictionary which I am sure you know is a list of word with meanings. BASIC has a dictionary (LIST, PRINT, MID$ etc) but the difference between the dictionaries is that the user may extend the Forth dictionary to suit his own purposes. Furthermore when Forth is executing code it looks for words in the dictionary backwards. This means that you can re-define existing words to have a new meaning (ie perform a new operation). So the power of Forth is not in is predefined operators but in user defined operators.

Forth is written in reverse Polish notation which enables Forth to use the stack for passing most of its data around although you can use variables as in BASIC. So to add two numbers together you would type:-

3 4 + . [return] and Forth would give you

7 ok

The '3' and the '4' are pushed onto the stack by Forth. The '+' word takes two numbers off the stack and adds them together leaving the result on the stack. The '.' word takes one number off the stack and then prints it.

Why accept such a bizarre way of doing maths? Well the simple answer is speed. It may surprise you to find out that BASIC operates in this manner. What you enter as a BASIC line is translated into reverse polish before it is execut- ed. Since Forth does not have this translation overhead it will run much faster. As an example I timed 30,000 empty DO LOOPs. On Forth it took 4 seconds. On BASIC it took 30! Now I appreciate that its an unfair test because Forth is integer whereas BASIC is real (ie floating point) but it does give an indication of the speed of Forth.

The reverse Polish may give Forth a speed advantage but it makes it very user-unfriendly. I found it very difficult to think in Forth although I am sure that it would come in time. To illustrate the point there is a word in Cubsofts' Forth RND which will put a 32-bit word on the stack. I wanted to produce a word that would give a random number within a specified range (ie like RAND(4) in BASIC). Now to use it you would give it the argument on the stack such as 20 RAND and it would leave the number on the stack. It it probably very simple if you know Forth but I couldn't work it out. In the end I set up and used two variables rather than comply with the idea of forth which is that you use the stack as much as possible. I am sure that there is a very neat way to do the task but it empahasized to me how you need to change your method of thinking to be able to use Forth.

New words like RAND are set up using colon definitions. This simply mean that you type a colon followed by the name of the new word and then the definition of the new word. As an example : COUNT 0 DO I . LOOP ; could be entered. The DO and LOOP are similar to BASIC and the 'I .' will print the counter. The zero in front of the DO is the value to count down to. So now you may enter 300 COUNT [return] and COUNT will display the numbers from 0 to 299 on the screen. It is the ability to define new words that gives Forth its power. Each programmer can set up his own dictionary for his particular purposes.

Cubsoft Forth comes on a cassette with a professionally printed manual of 40 pages. I understand that there are 48K and 96K versions available. The 96K version allows you to access the RAM hidden behind the EPROMS. My version was for the 48K and it loaded first time without any problems. The first thing you notice is that the cursor doesn't blink. I find this most disconcerting - I like cursors to blink as it lets me know that the machine is still running.

Once loaded you may start developing your Forth program. This may be done directly from the keyboard or you may edit what are known as 'screens'. To edit 'screen' 3 you would type 3 EDIT [return]. The screen clears and enters text mode. Screen editing is adequate. As I understand it in the original Forth you edited a 'screen' (ie 24 by 80 characters) of definitions. The screens in Cubsofts' version consist of only 240 characters composed as eight lines of 30. In actual fact the eight lines are really one long line as in basic. When you call them up the text doesn't appear until you press escape which seems rather pointless.

Although the manual is professionally printed it will not be enough for a noivce user to be able to get going with Forth. It is a technical manual which describes what Cubsofts' version of Forth does. It does not describe how to use the language. For that you will have to buy a book on the subject.

I found it very easy to 'crash' Forth. I am not saying that Cubsoft's Forth has bugs (it may have but I didn't find any). The problem is that unlike BASIC there is no built-in escape facility so if you put it into an infinite loop then it just 'hangs'. You have to turn the power off and start again.

I didn't find the help messages very helpful. 'Syntax error' at least tells you what the problem is but 'MSG 2' etc tells you nothing and you need to refer to the manual each time. Cubsoft also supply some utilities with Forth. These are predefined Forth word definitions to make using Forth on the Lynx easier. I can't see why they were not included in the Forth core. Having them as extras means that you have to load them as well as Forth before you can start.

To sum up then if you want to face the challenge of Forth then Cubsofts version could be just what you have been looking for. I couldn't recommend it to anyone struggling with BASIC. If you want to find out more about Forth I can recommend BYTE August 1980.

Interfacing Non-Standard Printers

There is no such thing as a standard printer and consequently interfacing between micros and printers can be a trying affair - you literally have to try and then try and then try some more. Give credit where credit is due. Camputers have made attaching a printer as simple as possible because you can buy one of the recommended printers and the standard interface, plug them in and away you go. Unfortunately not everyone can use one of the recommended printers. Like me, you may already have a printer, or perhaps you want a better printer or have some other reason for not wanting a recommended one. Whatever the reason the fact remains that if you opt for a non-standard printer you may well have trouble interfacing it to your Lynx.

Both the serial and parallel print routines work in the same way except that they output their characters to different ports. This is of no consequence when it comes to

understanding the routines with a view to modifying them. Consequently I will explain how the routine works and then note any differences.

The routines are split into two parts. First comes the routine for the control characters in the range 0-1FH. Then there is the code for the normal characters in the range 20H and upwards. The normal characters do not usually cause any problems. It is with the control char- acters that most people will have difficulty. The print routines are quite clever in how they handle the control codes. Unfortunately the accompanying documentation doesn't betray this. There are four bytes from 61BCH to 61BFH whos bits are used to flag whether any action is taken for each of the control codes.

If action is required then another routine is called. This deals with TAB (09H) and CR (0DH) separately. The tabs are expaned into blanks and the CR is changed to a linefeed (0AH) before being sent.

This substitution of 0AH for 0DH may cause problems since some printers require the 0DH control character. The substitution is made at 9FB4 in the parallel print routine and 9F5BH in the serial print routine. In both cases it is done by the code 3E 0A. This could be changed to 3E 0D if the 0DH (ie carriage return) control is required.

The remaining values (ie 0-8, 0A,0B and 0E-1FH) are indexed out of a table of substitution values. This is where the user may modify the routine to suit his own printer. As an example many printers require the ESCAPE character (1BH). The Lynx has bit 3 of the byte at 61BFH reset (ie 0) so that it will not be sent. Furthermore the substitution value in the table is zero so even if you set bit 3 the print routine will not send the 1BH value. You need to set bit 3 and change the value in the substitution table. For the parallel print the substitution table starts at 9FBAH. The table I have looks like:-

00 00 00 00 0C 00 00 00 00 09 00 00 0A 0A 00 00 0C 00 00 13 00 00 08 00 00 00 00 00 00 00 00 0A

For the serial printer it starts at 9F61H.

It may well be the case that the table varies both in content and in location so you may have to find your own table. The best way to find it is to locate the 0A0A bytes. Load your software, enter the montior and look for the 0A0AH word with the W (word search) command. W 0 0A0A [return] will display all the locations of the word 0A0AH.

The software auto runs and at first sight it may seem impossible to modify it. The routines are in two parts. First there is a BASIC routine which sets up the address of the drive routine at 6202H and sets up the bytes from 61BCH onwards. The BASIC routine then MLOADs a machine code drive routine. It is at this stage when 'CODE' appears on the screen that you can press ESCAPE and list the BASIC program. Hence you may modify it and save yourself a modified copy. Note that when you save a copy it should be saved to auto run with the first line of the program as the start line number.

In the BASIC program you will find a line which looks like 'DPOKE &6202,xxxx' where xxxx is the start of the print routine. You can use this information to save yourself a copy of the machine code part. The program ends at 9FF7H so to save a copy you use the monitor Dump command as follows:-

D xxxx 9FF7 0000 "CODE" [return].

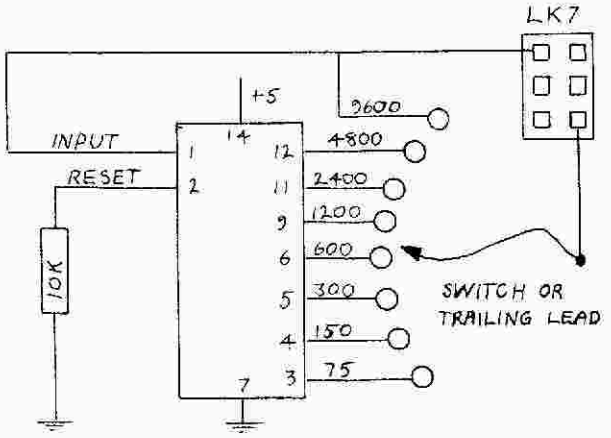
BAUD RATE MODIFICATION by Peter Collingridge

One of the draw-backs of the LYNX's 2400 baud rate is that cheaper printers can't be driven from the UART. Here is a small hardware modification which allows the UART to transmit baud rates down to 75 BAUD. The circuit works by intercepting the clock signal to the UART and dividing by the required number to reduce the baud rate, and then re-applying this clock signal to the UART.

There is a small plug/socket assembly labelled LK7 close to the UHF modulator. This can be used to select the baud rates 2400, 4800 and 9600. You can select 2400 by having the plug on the bottom two pins, 4800 with it on the middle two pins and 9600 with it on the top two pins. With the computer facing you the three pins on the right are linked together and go to the UART. The three pins on the left act as the sources of the clock signals. By removing the small black plug (and not losing it as I did!), it is possible to "tap off" the clock signal, divide it down and re-apply the new signal to the right-hand side pins. The simplest way I have found to do this is using a CMOS 4024 chip costing about 55p. Using this chip it is possible to genterate the following baud rates:- 4800, 2400, 1200, 600, 300, 150 and 75.

The supply voltages can be taken from anywhere, I soldered two wires to the connections on the back of the power socket beside the speaker. The 4024 can be put on a small piece of veroboard and outputs either fed to a multi-way switch or taken to pins on the veroboard to which wires can be attached as required. I stuck the board to the underside of the top half of the case so that it would not foul the other circuit components.

The circuit is as follows:-



SOFTWARE

HARMONOGRAPH By Chris Cytera

The harmonograph is a mechanical device which can draw a variety of intricate and complex patterns. It consists of two pendulums that are free to swing in any direction, with the pivot point some way down from the top. A table is attached to the top of one pendulum and a pen is attached to the top of the other. A drawing is made by fixing a piece of paper to the table, resting the pen on the table and setting the two penulums swinging.

The program simulates the operation of a frictionless harmonograph, drawing the patterns on the screen. To decrease the execution time of the program, the SIN calculation is done by a look-up table rather than use the SIN function each time. The generation of this table is the reason for the delay when the program is run.

The drawing can be stopped by pressing the space bar, it can be continued with "C" or a new pattern started by pressing "D". The parameters for the program are chosen at random, and you may have to start several patterns before you get an interesting one.

```
100 RANDOM
110 TEXT
120 PRINT @ 3,45;"PLEASE WAIT"
130 LET C=255,Q=C DIV 4,H=128,K=124,A=80
140 DIM S(C)
150 FOR S=0 TO C
160   LET S(S)=SIN(S*2*PI/C)
170 NEXT S
180 REPEAT
190   CLS
200   LET R=RAND(C),r=(R+Q) MOD C
210   LET S=RAND(C),s=(S+Q) MOD C
220   LET D=RAND(A),a=RAND(A)
230   LET B=RAND(A),c=RAND(A)
240   LET F=RND*7,f=RND*9,b=RAND(C)
250   LET X=S(R)*D,x=S(r)*a
260   LET W=S(S)*B,w=S(s)*c
270   LET Y=S((Q+R) MOD C)*D
280   LET y=S((Q+r) MOD C)*a
290   LET Z=S((Q+S) MOD C)*B
300   LET z=S((Q+s) MOD C)*c
310   LET M=0
320   LET J=0
330   REPEAT
340     REPEAT
350       J=J+1
360       T=F*J BNAND C,t=b+f*J BNAND C
```

```
370       U=Q+T BNAND C,u=Q+t BNAND C
380       PLOT M,X*S(T)+x*S(U)+W*S(t)+w*
S(u)+H,Y*S(T)+y*S(U)+Z*S(t)+z*S(u)+K
390       LET M=2
400     UNTIL KEYN=32
410     REPEAT
420       LET A$=GET$
430     UNTIL A$="D" OR A$="C"
440   UNTIL NOTA$="C"
450 UNTIL FALSE

100 REM PLANETS By Chris Cytera
110 RANDOM
120 PROTECT 0
130 CLS
140 REPEAT
150   LET E=RAND(165)+40,F=RAND(165)+40
160   LET L=RAND(6)+1
170   LET S=RAND(35)+1
180   LET s=S*S
190   FOR Y=-S TO S
200     LET X=SQR(s-Y*Y)
210     LET x=2*X
220     FOR I=-X TO X
230       IF RAND(x)-X<I THEN  INK L
240       ELSE  INK BLACK
250       DOT I+E,Y+F
260     NEXT I
270   NEXT Y
280 UNTIL FALSE

ROTATING SPHERE by Bill Walton.
100 CLS
110 INK CYAN
120 DPOKE &6292,&A000
130 CLS
140 LET X=80,A=0
150 MOVE 120,100
160 REPEAT
170   LET A=A+10
180   DRAW 120+X*COS(RAD(A))*SIN(RAD(A)*
0.95),100+X*SIN(RAD(A))
190   IF NOTA>1810 AND A<3620 OR A>4530
AND A<7240 THEN  DPOKE &6292,&A000
200   ELSE  DPOKE &6292,&C000
210 UNTIL A=7240
220 DPOKE &6292,&C000
230 PAUSE 1000
240 OUT &0080,2
250 PAUSE 1000
260 OUT &0080,16
270 GOTO 230

TRON BLOCKER by Stephen Sawyer.
5 GOTO 90
10 CODE  EB 21 F6 61 E5 21 00 82 E5
        21 00 00 C3 FF 3E
20 REM "TRON BLOCKER"
30 CALL LCTN(10),LCTN(20)
90 PROC INST
95 LET Y=0,G=0
99 REPEAT
100   VDU 2,0,1,7,4
110   FOR A=1 TO 40
120     PRINT CHR$(239);
130   NEXT A
140   FOR A=15 TO 225 STEP 10
150     PRINT  @ 3,A;CHR$(239); @ 120,A;
CHR$(239);
160   NEXT A
170   FOR A=1 TO 40
180     PRINT CHR$(239);
190   NEXT A
195   VDU 18
196   ?  @ 15,5;"YELLOW : ";Y; @ 60,5;
"GREEN : ";G;
198   IF Z$="Y" THEN  PROC BLOCK
200   CALL &8000
202   FOR A=10 TO 100 STEP HL*10
205     BEEP 100*HL,A,63
207   NEXT A
210   IF HL=2 THEN  PRINT  @ 50,100;
"GREEN LOST";
215   IF HL=1 THEN  PRINT  @ 50,100;
"YELLOW LOST";
216   PAUSE 40000
220   IF HL=1 THEN  LET G=G+1
225   ELSE  LET Y=Y+1
227   DPOKE &900A,S
250 UNTIL Y>9 OR G>9
260 VDU 1,6,2,2
300 IF Y>9 THEN  PRINT  @ 5,200;"YELLOW
WINS ";Y;" TO ";G;
310 ELSE  PRINT  @ 5,200;"GREEN
WINS ";G;" TO ";Y;
315 VDU 1,3,2,0
320 PRINT  @ 20,220;"Do you want to play
again ?";
330 LET G$=GET$
340 IF G$="Y" THEN  RUN
350 END
500 DEFPROC INST
505 PROTECT 0
506 LET A$=CHR$(1)+CHR$(2),B$=CHR$(1)+
CHR$(4)
510 VDU 1,6,2,0,4,2,2
520 PRINT  @ 40,5;CHR$(24);"TRON
BLOCKER";CHR$(25);
525 VDU 1,4,2,0
526 PROTECT 3
527 PRINT  @ 5,50;"The object of TRON
BLOCKER is to force  your opponent into
the outer wall,your trace ,his trace .
                              or the
obstructions [IF SELECTED]. ";
527.5 PROTECT 2
527.6 INK 5
528 PRINT  @ 5,100;"PLAYER 1 [YELLOW
TRACE .Start at top.]";
529 VDU 1,6,21
529.1 PROTECT 1
529.2 PRINT  @ 5,100;"_____
                    ";
_____
530 PRINT  @ 8,112;"Press the following
keys to move.";
530.4 INK 2
531 ? @ 9,126;A$"V";B$" to move down ";
532 ? @ 9,136;A$"}";B$" to move up ";
533 ? @ 9,146;A$"A";B$" to move left";
534 ? @ 9,156;A$"S";B$" to move right";
535 PROTECT 2
536 INK 5
540 PRINT  @ 5,170;"PLAYER 2 [GREEN
TRACE .Start at bottom.]";
540.5 PROTECT 1
541 INK 6
542 PRINT  @ 5,170;"_____
                    ";
_____
550 PRINT  @ 8,182;"Press the following
keys to move.";
551 ? @ 9,196;A$"[";B$" to move down ";
552 ? @ 9,206;A$"]";B$" to move up ";
553 ? @ 9,216;A$;CHR$(124);B$" to
move left";
554 ? @ 9,226;A$"(";B$" to move right";
560 PROTECT 4
561 INK 3
562 ?@12,240;"PRESS A KEY TO  CONTINUE";
565 LET A$=GET$
570 PROTECT 0
580 VDU 1,6,2,0,4,2,2,20
585 PRINT  @ 40,5;CHR$(24);"TRON
BLOCKER";CHR$(25);
590 VDU 1,4,2,0
592 PRINT  @ 6,50;"Enter speed
required [ 1 to 20 ]";
593 INPUT S
593.5 BEEP 100,100,63
594 IF S<1 OR S>20 THEN  GOTO 592
595 INK 5
595.4 LET S=S*350
595.5 DPOKE &900A,S
596 ? @ 6,65;"Speed selected ";S/350;
596.5 INK 4
597 ? @ 3,80;" The speed will increase
as the game     progresses.";
599 ?@ 6,110;"Do you want obstructions
[ Y or N ]";
600 LET Z$=GET$
620 BEEP 100,100,63
630 IF Z$="N" THEN  ENDPROC
640 IF NOTZ$="Y" THEN  GOTO 600
650 ENDPROC
700 DEFPROC BLOCK
705 VDU 1,5,2,0
710 FOR A=1 TO 20
720   ? @ RAND(110)+6,RAND(210)+15;"";
730 NEXT A
740 VDU 18
750 ENDPROC
```

The following machine code is required for TRON BLOCKER.
Enter it with the MONITOR M command.

```
8000  21 90 C2 22 00 90 21 AF  8070  2A 00 90 CD 7B 81 ED 43  80E0  00 CD 00 81 C3 1B 80 00  8150  ED 4B 06 90 7B B5 47 48
8008  DC 22 02 90 21 30 30 22  8078  04 90 22 00 90 21 09 90  80E8  21 01 00 C9 21 02 00 C9  8158  C9 C5 E5 21 FF FF 19 22
8010  04 90 22 06 90 21 02 01  8080  7E ED 4B 06 90 2A 02 90  80F0  FF FF 00 00 FF FF 00 00  8160  0A 90 D5 1B 7A B3 20 FB
8018  22 08 90 21 08 90 56 01  8088  CD 7B 81 ED 43 06 90 22  80F8  FF FF 00 00 FF FF 00 00  8168  D1 21 0A 00 CB 3A CB 3A
8020  80 00 ED 78 FE FF 2B 0A  8090  02 90 2A 00 90 ED 5B 0A  8100  C5 01 FF FF ED 79 C1 3E  8170  3E 01 D3 80 14 CD 90 09
8028  16 01 FE EF 28 14 16 02  8098  90 00 CD 70 00 00 01 04  8108  40 D3 80 D6 20 D3 80 70  8178  E1 C1 C9 CB 47 28 05 11
8030  18 10 06 02 ED 78 FE FF  80A0  90 0A A5 C2 E8 80 2A 02  8110  EB 71 AF D3 80 01 FF FF  8180  C0 FF 19 C9 CB 4F 28 05
8038  28 08 16 04 FE EF 28 02  80A8  90 CD 59 81 00 00 CD 70  8118  ED 79 C9 FF FF 00 C5 01  8188  11 40 00 19 C9 CB 57 28
8040  16 08 72 23 56 01 80 09  80B0  00 00 01 06 90 0A A5 C2  8120  C5 01 FF FF ED 79 C1 3E  8190  10 3E 03 B8 20 05 23 01
8048  ED 78 FE FF 28 10 16 04  80B8  EC 80 CD 3B 81 00 2A 00  8128  40 D3 80 D6 20 D3 80 70  8198  C0 C0 C9 CB 38 CB 38 48
8050  FE DF 28 14 16 08 FE FB  80C0  90 11 20 00 19 EB 2A 00  8130  EB 71 AF D3 80 01 FF FF  81A0  C9 3E C0 B8 20 05 2B 01
8058  28 0E 16 01 18 0A 06 08  80C8  90 3E 17 00 CD 00 81 CD  8138  ED 79 C9 2A 00 90 CD 70  81A8  03 03 C9 CB 20 CB 20 48
8060  ED 78 FE FF 28 02 16 02  80D0  4A 81 00 2A 02 90 11 20  8140  00 ED 4B 04 90 78 B5 47  81B0  C9
8068  72 72 2B 7E ED 4B 04 90  80D8  00 19 EB 2A 02 90 3E 15  8148  48 C9 2A 02 90 CD 70 00
```

LYNX EPROMS

Several members have written in to request a 'complete disassembly of the LYNX EPROMS with an explanation of each routine'. It is very easy to request it but I doubt that anyone has given their request very much thought. I have such a listing - 180 pages worth! To publish it in a newsletter of the current size would take about TEN YEARS.

The next best thing I can do is to publish a list of the routines and what they do. This is not as easy as it may seem. Firstly the routines don't always start and stop at convenient places but quite often they jump all over the place. This makes it difficult to allocate areas in the EPROM to particular routines so you will have to be broad-minded with the addresses I have given. Secondly some of the routines are still a mystery to me - I am not a mind reader so I don't know what the author was thinking about when he wrote them. Thirdly some small utility routines defy description. This is especially true when they are called by several different subroutines.

Finally there are two known versions of the BASIC EPROMS and I suspect that there may be more. I know of no identification byte(s) in the EPROMs to indicate which version you have. This means that the addreses I have given may not be the same as the ones in your EPROM.

To help you crack your version I will give you a few hints. Firstly at 6217H and 6219H you will find the addresses of the start of two lists of BASIC words. I call them PRIMARY and SECONDARY words because some words may appear as the first word on a BASIC line (PRINT, LET etc) ie primary words and some words may only appear in the middle of a BASIC line (CHR$,MID$, etc) ie secondary words. To separate the words the seventh bit of the first letter of each word in the list is set. The lists end with an 80H byte.

For the SECONDARY words (6217H) you will find an address stored at 6215H that points to the start of a list of addresses for each routine. The first secondary word is MEM and the first address is 1E69H which is where you will find the code for MEM. Note that the addresses are back to front so 1E69H will be found as 691E in the list.

The PRIMARY word list (6219H) is followed by two lists of addresses. The first is a list of validation routines. The address for the start of this may be found at 621BH. The second list is a list of execution addresses for the routines. The start of this list may be found at 621DH.

Although you don't NEED a disassembler to have a look inside the EPROMS they are very cheap and they do take a lot of the hard work out of the task. QUAZAR (see their ad.) sell one for £4.75 which is quite adequate. If you quote your membership number they will sell it to you for £4.27. You can then produce a disassembled listing of the EPROMs (or the part you are interested in).

Once you have a listing and found the addresses for the various words it is straightforward to divide up the listing. The hard work then begins. You have to trudge through the listing to fill in the gaps. If you want to have a go it shouldn't be too bad for you because I've broken the back of the work. If you find that you can fill in any gaps that I haven't cracked yet or can correct any of my mistakes please drop me line.

If you are interested in learning machine code I would recommend that you have a go at disassembling the EPROMs. It is a very good way of learning how to write machine code. I don't suggest that you try to crack all of it but just pick out a few routines and have a look at them. Have fun!

0000 Start. Disable interrupts. Jump to 003B.
0008 Output character in the A register.
0010 Find next non-blank character. DE points to where search is to start.
0018 Evaluate reverse polish expression and put into HL.
0020 Check if byte after RST 20 is the same as in the A register. Display SYNTAX ERROR if not.
0028 Evaluate reverse polish expression.
0030 Jump to monitor trap routine.
0038 Jump to 6297H.
003B Initialise the 6845 CRT controller.
0056 0065 Data for CRT controller.
0066 0068 NMI jump.
0069 006E Read a byte from red or blue bank
0070 0088 Read a byte from green or alt. green bank.
0089 0099 I don't think that this is used.
009A 00CD Output character in A register.
00CE 00E6 Find 10 bytes of data for characters.
00E7 01D3 Output bytes to screen.
01D4 0697 Bytes to form the characters.
0698 06CD Output routine.
06CE 06E9 Special/normal character routine
06EA 0729 List of addresses for the VDU routines.
072A 072F VDU 24 Set double height characters on.
0730 0733 VDU 25 Set single height character on.
0734 VDU 20 Overwrite on.
0736 073B VDU 21 Overwrite off.
073C 0745 VDU 18 Reverse video.
0746 0761 Move cursor over 1 character
0762 VDU 4 Clear screen
0765 VDU 23 Home cursor.
0768 VDU 16 Move cursor to top of the window.
076F 077A VDU 19 CR+LF if cursor is not at start of line.
077B 077D VDU 13 (and 31) CR+LF and clear to end of the line.
0787 078A VDU 06 Move cursor down one pixel.
078B 0799 VDU 29 Move cursor down three pixels.
079B 07A1 Move cursor to left hand column of the WINDOW.
07A2 VDU 09 Tab cursor to next field.
07C1 07DC VDU 30 Clear to end of line.
07DE 07E5 VDU 22 Backspace cursor.
07E6 080B Called by backspace routine.
080C VDU 05 Move cursor up one pixel.
0810 VDU 28 Move cursor up three pixels.
0820 0826 VDU 08 Backspace and delete character.
0828 082D VDU 14 Turn cursor on.
0831 0843 WINDOW routine.
0844 085D A routine to decide which bits need to be set. Used in the output of characters to the screen.
08E5 0885 Normal character output routine.
086B 08CE Output a byte to vdu ram.
08CF 08E4 Called by clear screen routine.
08E5 0925 Called by clear screen routine.
0926 09C2 Returns D set to FF or 00. Used by CLS.
092D 093C VDU 07 'BEEP'.
093D 0977 Basic BEEP routine.
0978 0980 Basic BEEP delay routine.
0981 09A3 SOUND routine.
09A4 09BC Test for escape.
09BD 0A75 Keyboard input routine
0A76 0B1A Keyboard data table.
0B1B 0B5A Find word in a list.
0B5B 0B64 Scan for a key.
0B65 0BB4 Read sync from tape.
0B85 0B92 Read a byte from tape.
0B93 0BB3 Write sync to tape

0BC5 0BCA Part of SAVE routine.
0BCB 0C59 SAVE routine.
0C5A 0C84 VERIFY.
0C87 0C8E APPEND.
0C8F 0CCB LOAD.
0CCE 0CD3 Escape route for read bit routine.
0CD4 0CF1 Read bit routine.
0CF2 0CFA Turn cassette circuitry on.
0CFB 0D08 Turn cassette circuitry off.
0D09 0D66 Data for wave. Pointed to by 62A1.
0D67 0D91 Write byte to tape.
0D92 0DA2 TAPE.
0DAD 0DC4 Data for the TAPE command.
0DC5 0DCA VDU 12 Move cursor one character to the right.
0DCB 0E04 All set to FF ie not used.
0E05 0E17 ERROR.
0E18 0E3B MOD.
0EF3 0E43 DIV.
0E44 0E4B Return INK value.
0E4C 0E50 Return PAPER value.
0E51 0E55 POS.
0E56 0E5A VPOS.
0E5B 0E5F GRAPHIC.
0E60 0E64 ALPHA.
0E65 0E6A INF.
0E6B 0E6F Data for INF.
0E70 0E7E Utility routine caled by BNOR.BNXOR & BNAND.
0E7F 0EAA BNOR.
0E8B 0E94 BNXOR.
0E95 0E9E BNAND.
0E9F 0EAD RAND.
0EAF 0EB2 CFR.
0EB3 0EB3 CCHAR.
0EB8 WHITE.
0EBB YELLOW.
0EBE CYAN.
0EC1 GREEN.
0EC4 MAGENTA.
0EC7 RED.
0ECA BLUE.
0ECD BLACK.
0E4D Validation for VDU.
0EE3 0EEE VDU routine.
0EF0 Validation for WINDOW.
0EF4 Validation for PLOT:BEEP:PAINT.
0EF6 0F03 Validation. Words of the form:- WORD a,b,c,d.
0F04 0F1C Validation for '?'.
0F1D 0F24 Part of CODE validation.
0F25 0F2B Validation for CODE.
0F2C 0F4A Part of CODE validation.
0F4B 0F4F Set INK routine.
0F50 0F54 Set PAPER routine.
0F55 0F5E MOVE.
0F5F 0F6A DRAW.
0F6B 0F7D PLOT subroutine.
0F7E 0F92 PLOT subroutine.
0F93 0FA4 PLOT subroutine.
0FA5 1016 PLOT.
1017 1045 PLOT subroutine.
1046 1050 PLOT subroutine.
1051 105D PLOT subroutine.
105E 1065 Output a line of characters terminated by a CR.
1066 106F Flash cursor and scan keyboard.
1070 108C Display buffer.
108D 1094 Loop until CR.
1095 10A6 Output the cursor.
10A7 112D Character input routines
11B9 11C6 Data. 'Line number?'
11E1 134C First list of BASIC words.
134D 1367 Set to B0. Is it used ?
136B 1415 Data. Routine addresses for the list of words.
1416 155F Second list of BASIC words.
1560 15F6 Validation routine adresses for BASIC words.
15F8 168B Execution routine addresses for BASIC words.
168C 16C3 Main routine.
16C4 1714 Primary routine - called from MAIN routine.
1715 17EB Data used in initialisation.
17E9 17EC Data used for BASIC prompt.
17ED 17F8 Data used to display the 'LYNX LOGO' at power on.
17FC 18E1 BASIC line analysis routine.
181F 1826 Line analysis subroutine.
1827 182C Line analysis subroutine.
182D 1856 Line analysis subroutine.
1857 186C Line analysis subroutine.
186D 1888 Line analysis subroutine.
188C 1892 Pick up address of a list of words.
1893 189A Loop down a list of words to find an address.
189B 1942 Line analysis subroutine.
1943 1948 Pick up address from a list of BASIC words.
1949 197D Find word in list,return address if pos.
197E 1986 Search for start of BASIC word in a list.
1987 1B53 Normal to internal language routine.
1B54 1B6F This looks like data ? No idea!
1B70 1B90 Line analysis subroutine.
1B91 1B97 This looks like data ? No idea!
1B98 1C3C Numeric (hex/decimal) input routine
1C3D 1DD7 Number analysis.
1DD8 Decimal point (not sure).
1E4F 1E56 Data 'Stopped'.
1E57 1E68 RND.
1E69 1E74 MEM.
1E76 1E9E Called by 'J' command.
1E9F 1EA8 Data 'in line'.
1EA9 1F04 Main process loop for BASIC
1F05 1F0C Read from port B0.
1F0D 1F12 Routine to print 'Ready!'.
1F13 1F1B Data 'Ready!'.
1F1C 1F2E Test for a full return stack
1F2F 1F3D Push HL onto the return stack.
1F3E 1F46 Push A onto the return stack.
1F47 1F54 Pop HL off the return stack.
1F55 1F5D Pop A off the return stack.
1F5E 1F69 LLIST subroutine: "LINK ON".
1F6A 1F6D LLIST subroutine: "LINK OFF".
1F6E 1F9C Called from MAIN routine. Zeros memory etc.
1F9D 1FA7 LOTN.
1FA8 1FB0 RANDOM.
1FB1 1FBD CLS.
1FBE 1FC3 END.
1FC4 1FD2 WEND.
1FD3 1FF9 WHILE.
1FFA 2012 UNTIL.
20013 201D REPEAT.
201E 2025 GETN.
2026 202E KEYN.
202F 203B Print or don't print routine.
203C 2058 Display character.
2059 205E Check and then evaluate an expression.
205F 206F CALL.
2070 2073 Save the value in HL at 61F4.

2074 2082 Check for stack full.
2083 2093 NEW.
2094 20B4 INPUT.
20B5 2107 Input buffer analysis.
2108 2121 Test what DE points to for a comma, a " or a CR.
2122 212C PAUSE.
212D 2131 PROTECT.
2132 2147 TEXT.
2148 217C LET.
217D 218F RESTORE.
2190 219F Search BASIC file for token in C.
21A1 21C4 WHILE subroutine.
21C5 21E9 READ.
21EA 21F3 Test for a valid variable.
21F4 2217 Variable validation routine.
2218 2233 Load A with variable number (ie A=1, B=2 etc) and HL will return with the address of the variable.
2234 224D Print HL in hex.
224E 2256 LPRINT.
2257 225E Prints a string.
225F 226E SWAP.
226D 227F CONT.
2280 2295 RUN.
2296 22D2 GOSUB
22D3 22EF Compare two strings for compatibility.
22E0 2337 LIST (in part)
2338 233E LLIST.
233F 2362 DEL.
2363 2379 Picks up a Basic argument ?
2371 237C Called by 2363-2370
237D 238B OUT.
238C 2396 DPOKE.
2397 23DA FOR
23DB 242B NEXT.
242C 2454 IF
2455 245F TRAIL.
245F is exectution address for LABEL:ELSE:DATA:REM.
2460 246D SPEED.
246E 2472 TRACE.
2473 247D ROUND.
247E 2488 LINK.
2489 24C1 Called from 252D.(part of LIST)
24C2 245F IF.
24F6 2509 TRACE handling routine.
250A 2523 SPEED handling routine.
2524 255B Input line analysis routine.
255E 2567 Add line length to IX.
2568 256D Test HL against DE.
256E 25C9 Load line to memory.
25CA 25E1 Called from MAIN. Zeros memory.
25E2 25F0 Read ASCII line, convert to internal language and store where DE points to.
25F1 2603 Validation utility routine.
2606 2615 Validation for ROUND,TRAIL,LINK,TRACE.
2616 261C Validation for LOAD,MLOAD,APPEND,VERIFY.
261D 262A Validation for SAVE.
262D 265C Validation for INPUT.
265D 2663 Validation for RESTORE.
2664 2667 Validation for LIST,LLIST,RENUM.
2668 2673 Validation for CALL.
2767 2678 Part of the 'NOT FOUND' logic.
267C 26BC Validation routine for LET.
26BD 26C2 Validation routine for DPOKE,POKE,DOT,OUT,MOVE, DRAW,SOUND.
26C3 26D7 Validation routine for GOSUB,GOTO.
26D8 Validation routine for NEXT.
26DB 26DE Validation for MON,ENDPROC,RANDOM,END,STOP,RETURN, CLS,CONT,NEW,WEND,REPEAT,DISK.TEXT.
26DF 26FB Validation for FOR.
26FD 2706 Validation for IF.
2702 270C Validation for INK,PAPER, PAUSE, WHILE, UNTIL. RESERVE,CFR,CCHAR,SPEED,PROTECT,ERROR, TAPE.
270D 2714 Convert to internal language and check for a comma.
2715 2720 Test for a variable and find next non-blank.
2721 273B 2739 is validation routine for READ.
273C 2744 Validation for REM,DATA,EXT,LABEL.
2745 275B Validation for SWAP.
275C 2764 Copy WRA1 to where DE points.
2765 2776 Line analysis subroutine.
2779 2791 Line analysis subroutine.
2792 27D1 Part of the find logic for looking at words in a list. 27A9 Validation for ELSE.
27D2 27E4 Print a word from a list (&219').
27E5 284B Display coded line. Pointed to from 62B2.
284C 285E Test for FOR,WHILE and REPEAT,
285F 2876 Test for NEXT,WEND and UNTIL.
2877 2889 Print spaces for indentation.
288A 28CE AUTO.
28CF 28D8 Data for AUTO. The numbers 100 and 10.
28D9 2903 BIN.
2904 294D PROC.
294E 295C Load (DE) to (HL) testing for CR and(.
295D 296E Validation for PROC.
296F 2987 Validation for DEFPROC.
2988 2992 ENDPROC.
2993 29A8 Part of the PRINT routine.
29A9 2A10 PRINT.
2A11 2A4D Validation for PRINT.
2A4E 2A7B RESERVE.
2A7E 2B1E RENUM
2B1F 2B4A Find line with line number in WRA1.
2B4B 2B60 Line analysis subroutine.
2B61 2B6F Line analysis subroutine.
2B70 2B75 Load A to where IX points. Increment IX.
2B76 2B92 STR$
2B93 2B98 KEY$
2B99 2B9F Part of GET$
2BA0 2BA7 GET$
2BA8 2BDC UPC$
2BDD 2C27 String execution routine.
2C28 2C31 CHR$
2C32 2C40 LEFT$
2C41 2C62 MID$
2C63 2C8E RIGHT$
2C8F 2D0D Validation routine.
2D0E 2D13 VAL(. Note left bracket is part of the word.
2D14 LEN(.
2D26 ASC(.
2DCB 2E65 DIM.
2E66 2E92 Validation for DIM.
2E93 2EBA Numeric output routine.
2EBB 2EC0 Validate SWAP and multiply.
2EC1 2EE1 Multiply and add routine.
2EEA 2EFE (IY) to WRA1.
2EFF 2F0D WRA1 to (IY).
2F0E 2F1C (IY) to WRA2
2F1D 2F2B (IX) to WRA2
2F2C 2F39 Push WRA1 and jump to (IX).
2F3A 2F4B Pop WRA2 and jupt to (IX). 2F4C 2F57 LN.
2F58 2F84 ARCSIN.

```
2F85 2FA4 ARCCOS.
2FA5 2FD8 ARCTAN.
2FD9 2FE0 ARCTAN subroutine
2FE0 300F Data for ARCTAN
3010 3051 FACT.
3052 3093 SQR.
3094 3099 Part of power routine
309A 30CD '↑' Power routine.
30CE 30E3 Part of power routine
30E4 3169 LOG.
316A 3183 Data for LOG routine.
3184 318F EXP.
3190 31AA TAN.
31AB 31F5 ANTILOG.
31F6 3219 Data for ANTILOG
321A      COS.
3223 328A SIN.
328B 32A4 Data for SIN and COS
32A5 32AE NOT.
32AF 32BC = as in IF A=B
32BD 32C1 <> Not equal routine.
32C2 32CF > Greater than routine.
32D0 32D4 <= Less than or equal to routine.
32D5 32D9 < Greater than routine.
32DA 32DE >= Greater than or equal to routine.
32DF 32F5 OR.
32F6 3303 AND.
3304 3306 . Decimal point
3307 3310 . Decimal point
3311 331C - Minus sign.
331D 3323 HIMEM.
3324 3328 HL.
3329 3335 INP.
3336 333E DPEEK.
333F 3344 PEEK.
3345 334D DEG.
334E 3356 RAD.
3357 3376 Utility called by RAD and DEG
3377 3433 Take data from program line.
3434 3439 ABS.
343A 3455 SGN.
3456 3474 INT.
3475 3496 FRAC.
3497 34C3 Convert floating point in WRA1 to binary in HL.
34C4 34EE Binary in HL to floating point in WRA1.
34EF 34F4 Test for '0' to '9'
34F5 3503 Test for 'A' to 'Z' and 'a' to 'z'.
3504 350C Test for uppercase/lowercase.
350D 3517 Zero WRA1
3518 351A Data. Used to to 'Clear to end of line'
351B 3529 RST 08 Clear to end of line and CR.
352A 352E Output a space
352F 3538 Output character in A to the screen.
3539 3541 Output text; pointed to by HL, ended with 0.
3542 3553 Swap WRA1 and WRA2.
3554 3560 Called by the subtraction routine (366A).
3561 3578 Compare WRA2 & WRA2.
3579 3588 Subroutine called by compare routine.
3589 35AD Subroutine called by the addition routine.
35AE 35B0 FALSE ie load WRA1 with zero.
35B1 35B9 Load number pointed to by HL into WRA1
35BA 35BE Load number pointed to by HL into WRA2
35BF 35C3 TRUE ie load WRA1 with one.
35C4 3609 Data. Five byte numbers.
360A 360B I don't think that these bytes are used.
360C 3610 PI.
3611 3615 Data for PI.
3616 3661 Numeric manipulation routines.
3662 3669 ? Fancy rotate with (HL) !! ??
366A 366C '-' WRA1=WRA1-WRA2.
366D 36B5 '+' WRA1=WRA1+WRA2
36B6 36C7 Called from Plus sign routine.
36C8 374B '*' WRA1=WRA1*WRA2.
374C 3769 Subroutine called by multiply.
376A 379E Subroutine called by multiply.
379F 37AF Subroutine called by multiply.
37B0 3888 '/' WRA1=WRA1/WRA2.
3889 3895 Jumped to from 349C
3896 38AA Subroutine called by multiply and divide.
38AB 38E7 Multiply subroutine.
38EB 38EE 'J' command.
38EF 3AF9 Data. Error messages.
3AFA 3B61 Error message selection code.
      3B4A is the DEFPROC execution address.
3B64 3BC7 MONitor.
3BC8 3BCE Data '????' . Monitor error message.
3BCF 3C02 Two byte addresses for the monitor commands A-Z.
3C03 3C07 Monitor S command.
3C08 3C1E Monitor B command.
3C1F 3C65 Monitor P command.
3C66 3C73 Monitor M command.
3C74 3C88 Monitor N command.
3C89 3CE7 Monitor I command.
3CE8 3D08 Data for I command - the register names.
3D0C 3D16 Display HL.
3D17 3D23 Pick up argument into HL.
3D24 3D43 Subroutine called by routine above.
3D44 3D5F Pick up three arguments.
3D60 3D6F Monitor T command.
3D70 3D77 Monitor C command.
3D78 3D8A Monitor J command.
3D8B 3DC7 Monitor A command.
3DC9 3DCB Data '??' used in A command.
3DCC 3DDD ???? Doesn't seem to be used!!!!
3DDE 3DEA Monitor Q command.
3DEB 3DF5 Monitor O command.
3DF6 3DF9 Monitor G command.
3DFA 3DFF Test for a monitor command letter.
3E00 3E20 Monitor W command.
3E21 3E29 L command subroutine.
3E2A 3E43 Monitor L command.
3E44 3E5E Monitor V command.
3E5F 3E6D Part of U command.
3E6E 3E8D Monitor U command.
3EBE 3EB5 Monitor X command.
3EB6 3EA9 Display memory starting from where DE points.
3EAA 3ECA Display the flags.
3ECB 3ED3 Data. Flags.
3ED4 3EFB Monitor E command.
3EFC 3F45 Monitor D command.
3F46 3F4D Write HL to tape.
3F4E 3F61 Check for valid file name.
3F62 3F69 MLOAD.
3F6A 3FC2 Monitor R command.
3FC3 3FD2 Part of R routine.
3FD3 3FD9 Multiply subroutine.
3FDA 3FE4 Part of DIM routine.
3FE5 3FFF Set to FF and not used.
```

Machine Code for Beginners
Part 5.

I have had some letters which indicate that I have 'lost' a few of you so it would seem to be a good idea to recap what has been covered so far. The heart of a micro-computer is the Central Processor Unit or CPU for short. In our case this is the Z80 chip. The other components of a computer are Memory and Input/Output devices (I/O for short).

CPU

The Z80 takes instructions (or codes) from memory and processes them. These instructions are called a PROGRAM. The program is what gives the computer its power because it can be set up by the programmer to perform many different tasks. Machine Code for Beginners is all about learning what the codes do and how to put them together to form a program. Internally the Z80 has several REGISTERS which are really internal memory (RAM). The difference between reg- isters and RAM is that the Z80 can perform special manipulations (eg addition) with registers which it can't do with memory.

MEMORY

There are basically two types of memory Random Access Memory (RAM) and Read Only Memory (ROM). There are several variants of ROM (PROM, EPROM EEPROM) but for our purposes they may all be considered as ROM. There are also two types of RAM. These are static and dynamic. Once again for our purposes they may both be simply considered as RAM.

I/O DEVICES

These devices allow the computer to communicate with the external world. All LYNXes have keyboards (input) and screens (output). Additionally they may have cassette re- corders (I/O), disks (I/O), Printer (output), etc.

The CPU is connected to its memory and I/O devices by a BUS (wires). In actual fact this bus is three buses. They are the DATA bus for the communication of data, the ADDRESS bus which supplies the addresses for memory and I/O devices and a CONTROL bus which controls the operations of the computer.

When you switch-on the Z80 the first instruction is picked up and processed. Once it has processed the instruct- ion the next instruction is picked up and processed. This process is continuous. All the Z80 does is fetch, process, fetch, process, fetch, process etc. etc.

The Z80 works in binary - humans don't (well not normally). The Z80 will be quite happy to process codes like 01110110. Unfortunately not only does it means little to humans but they have great difficulty in distinguishing between binary numbers. It is fairly trivial to convert from binary to hexadecimal and so it is normal to use hex rather than binary.

Having looked at some of these codes it was noted that there were several problems associated with putting programs together in hex. The answer to these problems was the use an ASSEMBLER. An assembler takes 'English' style words and translates (or assembles) them into a machine code program. To help the programmer they have other features which make the task even easier such as symbolic labels, jump calculation etc. They also have pseudo-opcodes. As their name implies these are not Z80 opcodes but facilities which aid the programmer to generate his program.

The Z80's instruction set is split into several groups. So far the following have been covered:-

LOAD

These instructions are used to load data from memory to the registers and from the registers to the to memory.

PUSH & POP for loads to and from the stack.

MATHEMATICAL

8-bit and 16-bit addition.

8-bit and 16-bit subtraction.

LOGICAL

XOR exclusive or.

OR logical or.

AND logical and.

CONTROL

CALL for calling subroutines.

RETURN for returning from subroutines.

RST for one byte calls.

JUMPS for jumping around in the program.

BIT MANIPULATION

SET for setting bits in a byte.

RES for reseting bits in a byte.

BIT for testing whether a bit is set.

INPUT and OUTPUT

IN input data to the A register.

OUT output data.

There are still a few more opcodes to be covered but we have covered enough to write reasonable programs. So how do you write program?

Not so long ago I was thinking of a program to put on a display stand. It would save me the trouble of having to tell people about NILUG. Although you can use double height characters to aid clarity I found that I really needed something even bigger. I felt that double width characters was what I needed since they could be used in conjunction with the double height facility to produce chunky characters.

Once you have decided what you want to program next comes deciding how you are going to solve the problem you have set yourself. Note that this is not the same as deciding how you are going to PROGRAM the solution. What comes first is deciding upon the METHOD.

The details on how characters are stored in the EPROMs are given in Issue 1 so I will just give a brief explanation here. The ASCII chracters from 20H to 7FH are formed by bytes in a table. There are 10 bytes per character and since characters are six pixels wide only bits 5-0 are used. Bits 7 and 6 are ignored. So a capital E could be stored like this:-

```
00000000
00000000
00111111
00100000
00111110
00100000
00100000
00111111
00000000
00000000
```

One way of creating double width characters would be to split up each letter into two characters of the normal size like this:-

```
00000000   and   00000000
00000000         00000000
00111111         00111111
00110000         00000000
00111111         00111100
00110000         00000000
00110000         00000000
00111111         00111111
00000000         00000000
00000000         00000000
```

When they were displayed on the screen they would look like:-

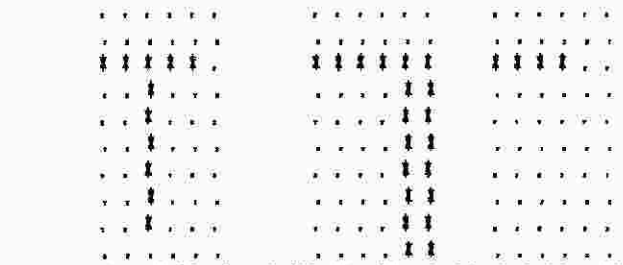
```
000000000000
000000000000
111111111111
110000000000
111111111100
110000000000
110000000000
111111111111
000000000000
000000000000
```

With the double height option in effect the characters would be well proportioned.

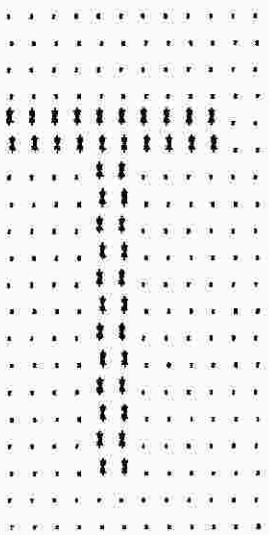
I would only need about 40 characters (A-Z. 0-9, plus a few more ,.?! etc) and this would involve defining 40x2x10 ie 800 bytes. You can program them if you want but its not for me.

A second method would be to take the bytes that are already defined and expand them. Then you wouldn't have to enter the pixel definitions for all the new characters. Each dot or pixel in the original character def- inition will have to become two dots on the expanded character.

Take the following character 'T' as an example. What we want is code to change



to form the following (with the the double height option in effect):-



The Z80 has bit manipulation opcodes and as you will see it is not difficult to expand the bytes. The next problem is to decide where the bytes should be expaned to. Its not enough to just expand the bytes because they have to be put out to the screen. Fortunately the Lynx has user defined graphics characters. So the obvious thing to do is to expand the bytes into two user defined characters. Once defined they can then simply be printed.

The following program will produce double width characters and print them.

```
7000            0010        ORG  7000H
7000 0000       0020 NULL   EQU  0
7000 000D       0030 CR     EQU  0DH
7000 0001       0040 INK    EQU  1
7000 0002       0050 PAPER  EQU  2
7000 0004       0060 CLS    EQU  4
7000 0007       0070 BEEP   EQU  7
7000 0012       0080 INVERS EQU  18
7000 0017       0090 HOME   EQU  23
7000 0018       0100 DON    EQU  24
7000 0019       0110 DOFF   EQU  25
7000 0001       0120 BLUE   EQU  1
7000 0002       0130 RED    EQU  2
7000 0004       0140 GREEN  EQU  4
7000 0006       0150 YELLOW EQU  6
7000 0007       0160 WHITE  EQU  7
7000 0008       0170 BLACK  EQU  8

                0190 ;SET THE GRAPHICS POINTER TO
                0200 :WHERE THE TWO CHARACTERS WILL
                0210 ;BE DEFINED
7000 218B70     0220        LD   HL,CHRS
7003 227162     0230        LD   (6271H),HL

7006 219E70     0250 RSTART LD   HL,TEXT-1
7009 23         0260 LOOP   INC  HL
700A 7E         0270        LD   A,(HL) ;GET THE CHARACTER
700B B7         0280        OR   A  ;TEST FOR THE LAST BYTE
700C 28F8       0290        JR   Z,RSTART

700E FE20       0310        CP   20H ;IN RANGE 1-1FH ?
7010 3003       0320        JR   NC,DOUBLE
                0330 ;SPECIAL BYTE
7012 CF         0340        RST  8  ;OUTPUT AS NORMAL
7013 18F4       0350        JR   LOOP

7015 CD7270     0370 DOUBLE CALL DELAY
                0380 ;SAVE THE POINTER TO THE CHARACTER
                0390 ;IN THE MESSAGE
7018 E5         0400        PUSH HL
7019 CDCE00     0410        CALL 00CEH :FIND BYTES
701C 060A       0420        LD   B,10
                0430 ;POINT DE TO WHERE GRAPHICS
                0440 ;CHARACTERS 80H & 81H WILL
                0450 ;BE DEFINED
701E 118B70     0460        LD   DE,CHRS
                0470 ;LOOP THROUGH THE 10 BYTES
7021 4E         0480 CLOOP  LD   C,(HL)
7022 AF         0490        XOR  A :ZERO ALL BITS OF A
7023 CB69       0500        BIT  5,C  ;TEST BIT 5
7025 2804       0510        JR   Z,NS1
7027 CBEF       0520        SET  5,A :SET BITS 5 AND 4
7029 CBE7       0530        SET  4,A
702B CB61       0540 NS1    BIT  4,C ;IS BIT 4 SET?
702D 2804       0550        JR   Z,NS2 ;JUMP IF NOT SET
702F CBDF       0560        SET  3,A
7031 CBD7       0570        SET  2,A
7033 CB59       0580 NS2    BIT  3,C
7035 2804       0590        JR   Z,NS3
7037 CBCF       0600        SET  1,A
7039 CBC7       0610        SET  0,A
703B 12         0620 NS3    LD   (DE),A ;SAVE THE BYTE
                0630 ;NOW ADD 10 TO DE TO MAKE IT
                0640 ;POINT TO THE SECOND CHARACTER
703C E5         0650        PUSH HL
703D 210A00     0660        LD   HL,10
7040 19         0670        ADD  HL,DE
7041 EB         0680        EX   DE,HL
7042 E1         0690        POP  HL
                0700 ;NOW TEST BITS 2,1 AND 0
7043 AF         0710        XOR  A
7044 CB51       0720        BIT  2,C
7046 2804       0730        JR   Z,NS4
7048 CBEF       0740        SET  5,A
704A CBE7       0750        SET  4,A
704C CB49       0760 NS4    BIT  1,C
704E 2804       0770        JR   Z,NS5
7050 CBDF       0780        SET  3,A
7052 CBD7       0790        SET  2,A
7054 CB41       0800 NS5    BIT  0,C
7056 2804       0810        JR   Z,NS6
```

```
7058 CBCF      0820        SET  1,A
705A CBC7      0830        SET  0,A
705C 12        0840 NS6    LD   (DE),A ;SAVE THE BYTE
               0850 ;POINT DE AND HL TO THE NEXT
               0860 ;BYTES TO BE PROCESSED
705D 13        0870        INC  DE
705E 23        0880        INC  HL
               0890 ;RESET HL
705F E5        0900        PUSH HL
7060 21F6FF    0910        LD   HL,-10
7063 19        0920        ADD  HL,DE
7064 EB        0930        EX   DE,HL
7065 E1        0940        POP  HL
               0950 ;KEEP GOING UNTIL FINISHED
7066 10B9      0960        DJNZ CLOOP
               0970 ;DISPLAY CHARACTERS &80 AND &81
               0980 ;WHICH HAVE JUST BEEN DEFINED
7068 3E80      0990        LD   A,80H
706A CF        1000        RST  8
706B 3E81      1010        LD   A,81H
706D CF        1020        RST  8
               1030 ;RESTORE HL WHICH POINTS TO THE
               1040 ;BYTES OF THE MESSAGE
706E E1        1050        POP  HL
706F C30970    1060        JP   LOOP

               1080 ;A DELAY ROUTINE WHICH ALSO TEST
               1090 ;FOR THE ESCAPE KEY
7072 E5        1100 DELAY  PUSH HL
7073 C5        1110        PUSH BC
7074 F5        1120        PUSH AF
7075 210600    1130        LD   HL,0006 ;A DELAY COUNTER
7078 2B        1140 DEL1   DEC  HL
               1150 ;TEST FOR ESCAPE
7079 018000    1160        LD   BC,0080H
707C ED78      1170        IN   A,(C)
707E CB77      1180        BIT  6,A
7080 CA0000    1190        JP   Z,0
7083 7C        1200        LD   A,H
7084 B5        1210        OR   L
7085 20F1      1220        JR   NZ,DEL1
7087 F1        1230        POP  AF
7088 C1        1240        POP  BC
7089 E1        1250        POP  HL
708A C9        1260        RET

               1280 ;SPACE FOR CHARACTERS 80H & 81H
708B 00000000  1290 CHRS   DEFB 0,0,0,0,0
     00
7090 00000000  1300        DEFB 0,0,0,0,0
     00
7095 00000000  1310        DEFB 0,0,0,0,0
     00
709A 00000000  1320        DEFB 0,0,0,0,0
     00
               1330 ;THE MESSAGE
709F 02080418  1340 TEXT   DEFB PAPER,BLACK,CLS,DON,INK,GREEN
     0104
70A5 2A202A20  1350        DEFM /* *  N I L U G  * * /
     204E2049
     204C2055
     20472020
     2A202A20
70B9 0102      1360        DEFB INK,RED
70BB 202A202A  1370        DEFM / * * N I L U G * *  /
     204E2049
     204C2055
     2047202A
     202A2020
70CF 0106      1380        DEFB INK,YELLOW
70D1 2A202A20  1390        DEFM /* *  N I L U G  * * /
     204E2049
     204C2055
     20472020
     2A202A20
70E5 00        1400        DEFB NULL
```

How did I go about writing the program? Firstly I decided that the message would be held in ASCII within the program. I would need some means of deciding when the message was finished. This can be done by putting a byte on the end of the message which wouldn't be printed. Zero is a good candidate for this since it is easy to test for.

Having decided upon the message we need to consider how to print characters. This is already written for us. We can use RST 8. To use this we need to load the A register with the ASCII for the character. To work through the message we will need some form of pointer to indicate where the next character will come from. You could use HL, DE or BC register pairs for the pointer. Why did I choose HL? There is no real reason so it must just be habit. There are some operations which can be done with HL which can't be done with BC or DE so I usually use HL where possible.

The first part of the program is a loop which increments HL, loads the A register with what HL points to and then tests to see if zero has been picked up. See lines 250-290. Now there are two types of byte that may be in the message - character bytes and special bytes in the range 0-1FH. The next part of the program test for which type of byte has been loaded and either we jump to the routine to double the width of the character or we fall through into a part of the program to handle the special bytes. The special bytes are simply output and this allows us to put codes to change the colour of the INK, PAPER, clear the screen, etc, within the message.

The double width code starts at line 370. In the program a delay routine is called which not only creates a delay but also tests for the ESCAPE key. This is not strictly necessary but I planned to extend the program to take codes to control the speed of the display and hence a delay routine was necessary. After this we save the HL register pair by pushing it onto the Z80 stack. It could have been by loading HL to some memory location but pushing is quite appropriate here because it can be popped off at the end of this section of code. The A register has the character required in it and there is a routine at 00CEH which will return HL pointing to the first of the ten bytes which make up the character. This byte is loaded into the C register and the A register is zeroed with XOR A. Now we examine the C register to see which bits are set and which are not. Bits 0,1,2,3,4 & 5 of the C register are used to define the pixels of the character. The first stage in doubling the width of the character is to examine bits 5,4 and 3 and to set bits 5 & 4, 3 & 2 and 1 & 0 of register A as required. See lines 490 to 610. Once the double width byte has been created it is saved by loading it to where the DE register pair point. DE is then adjusted to point to the first byte of the second graphics character. The C register is then examined again. This time bits 2,1 and 0 are examined to create the graphics byte which is then stored. All ten bytes of the normal character are analysed in this way to form the two graphics characters. Once complete both graphics char- acters are printed and with the double height option in effect the characters will appear well proportioned.

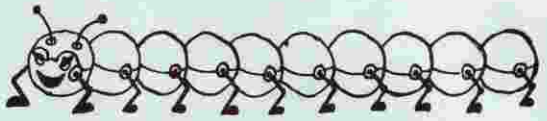
As I have already indicated there is more to machine code programming than I have been able to cover in this series. However if you have grasped the concept that the Z80 takes a code from memory, processes it and then picks up the next you are well on the way to understanding machine code. You should be able to pick up the rest of the codes from a good book. I can suggest Programming the Z80 by Rodnay Zaks and Z80 Language programming by Lance Leventhal. Unfortunately neither are cheap.

LYNX 48/96K SOFTWARE

QUAZAR COMPUTING.	
Siege Attack	£ 5.95
Puzzle Pack	£ 5.95
The Worm	£ 5.95
Labyrinth	£ 4.75
Chancellor	£ 4.75
Reversals	£ 4.75
Space Trek	£ 4.75
Disassembler	£ 4.75
Midnight Blitz	£ 4.75
Quacman	£ 5.95
GEM SOFTWARE	
Monster Mine	£ 7.95
Golf	£ 7.95
Sultan's Maze	£ 7.95
Gempack IV	£ 7.95
Oh Mummy!	£ 7.95
Gamespack I	£ 7.95
Spanner Man	£ 7.95
Twinkle	£ 7.95
Digger Man	£ 7.95
LEVEL 9 COMPUTING.	
Colossal Adventure	£ 9.90
Adventure Quest	£ 9.90
Dungeon Adventure	£ 9.90
Snowball	£ 9.90
Compass (Compression Assembler)	£15.00
ABERSOFT.	
Mazeman	£ 4.95
LYNXMAN.	
Vorlon Invaders	£ 5.00
ROMIK SOFTWARE	
Power Blaster	£ 9.99
Floyd's Bank	£ 9.99
Atom Smasher	£ 9.99
3D Monster Chase	£ 9.99
FL SOFTWARE	
Roader	£ 5.95
Coder	£ 7.50

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

CENTIPEDE



CENTIPEDE is now available. An excellent machine code game, very fast, noisy, breathtaking full colour graphics and exhilarating sound. High score table and joystick compatible. Read the review in this issue of Nilug. A bargain at only £7.95.

ASTRO ARENA will soon be available. 100% machine code with smooth high resolution full colour graphics. Fantastic sound, high score table and joystick compatible. Keep an eye open for this eye opener.

Our software is of the highest quality and nothing else, so,

DONT JUST IMAGINE THE ULTIMATE GAME, **PLAY IT!**

If abroad, please add 50p for P+P, and make cheques/PO's payable to PLAY IT, 79 Sleaford Road, BOSTON, Lincs. PE21 8EY

Z80 MACHINE CODE PROGRAMMERS

Wanted for very lucrative translation work on both games and business software. The work is on a free-lance basis and is paid by royalty payments. Projects are on-going and may run on for several years forming a stable basis for a career in the micro-computer industry. Lynx machine coders are particularly welcome but the ability to write Z80 assembler (on a Spectrum for example) does not exclude your application. Apply in the first instance by phoning Barry Hartley-Jones on 01-567 4341 or Prestel Mailbox 015671460.

LYNX SUPER CHESS

-7 levels of play
-recommended move
-set up position
-self play mode
-excellent graphic board display

Comes on commercially recorded cassette with full instructions.

Price : £8.95

From:-
CP Software
2 GLEBE RD,
UXBRIDGE,
MIDDLESEX
UB8 2RD.

LYNX HARDWARE

Three in one PCB
1. EPROM Programmer
2. Centronics Interface
3. 24 I/O Lines

Complete with software driver routines for the programmer and Centronics Interface and Circuit diagrams.
Send SAE for comprehensive details to:-

J.P.Kupps
8 Sherbourne Avenue,
Nuneaton,
Warks CV10 9JE.

VORLON INVADERS

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

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