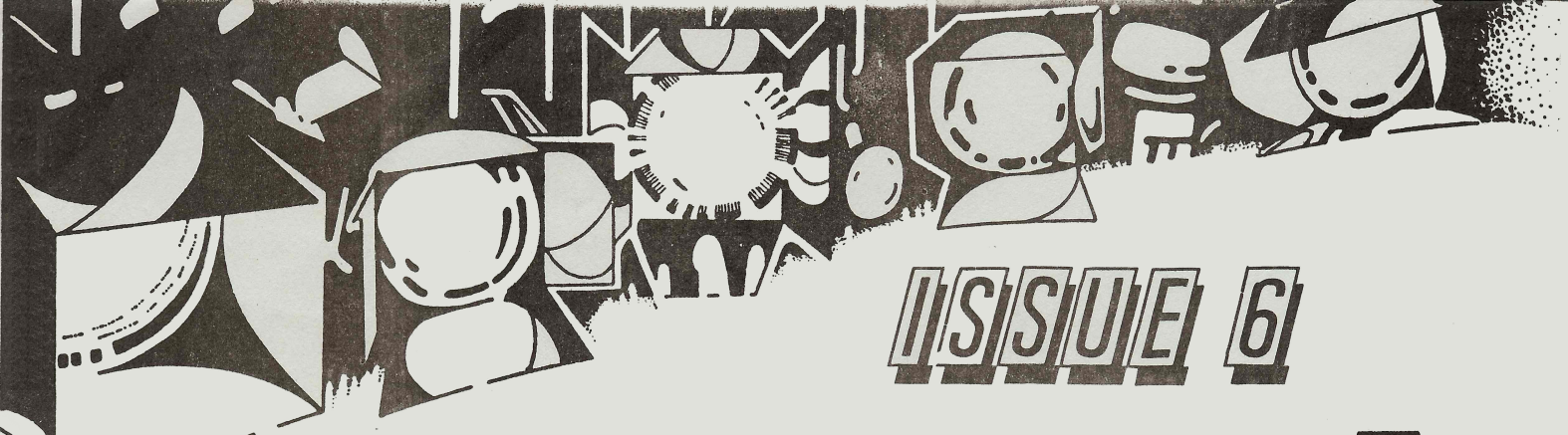
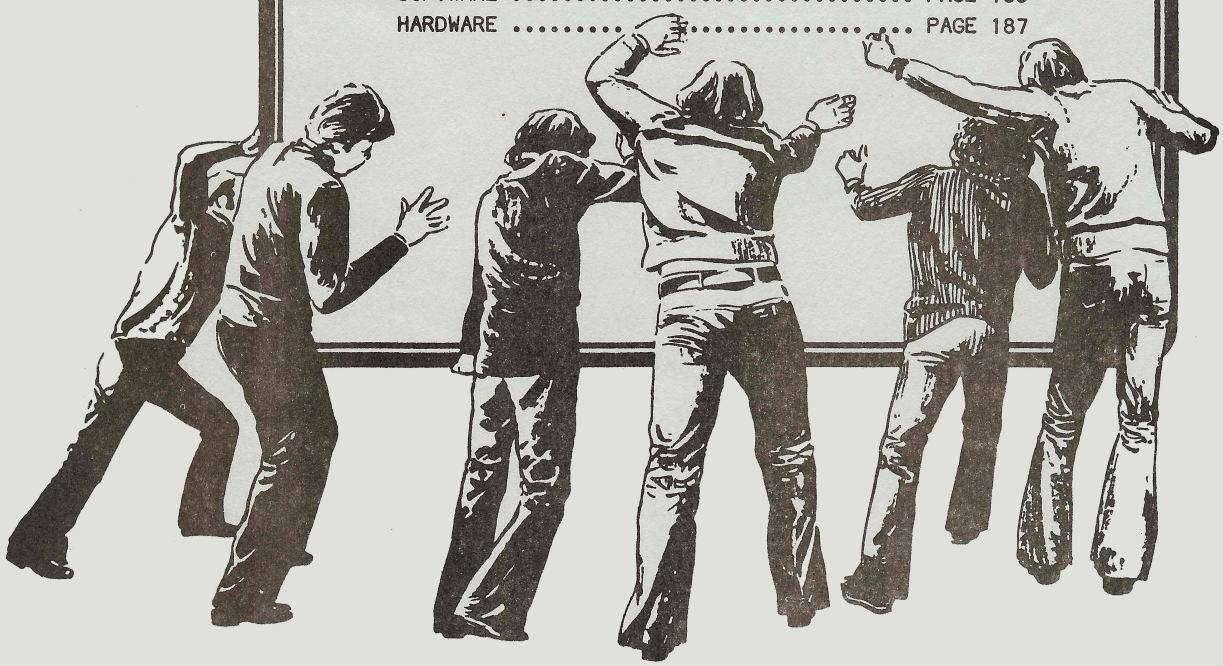


ISSUE 6

memopad

Memotech Computer User Club Magazine

Contents



Volume 0010

£1.35

memopad

Memotech Computer User Club Magazine

Edited by Tim Marstian

Artwork Anthony "Joe90"

Executive Editor Keith Hook

MEMOPAD IS PUBLISHED BY SYNTAXsoft FOR THE MEMOTECH USER GROUP
UNIT 109,GLENFIELD PARK, GLENFIELD ROAD, NELSON BB9 8AR
TELEPHONE: 0282 698849

AVAILABLE BY SUBSCRIPTION ONLY.

Editorial

By the time you recieve this magazine you may have read all sorts of conflicting stories regarding the demise of Memotech. Yes, Memotech is going out of business on the 31st of March 1985.

However, this should cause you no inconvenience. The plain truth is that Memotech has been taken over by Geoff Boyd. Geoff Boyd is the brains behind the original design of the machine. It is also obvious that bankers etc have great faith in his ability to carry on the business by making funds available for him to take over the company .

As far as you, the end user, is concerned you should not notice the difference - except that there will be a geater effort to get the Memotech machine recognised and located in its rightful place within the British market. Geoff visted Syntaxsoft two weeks ago to put his cards on the table and has offered, as a sign of good faith, to extend the warranty on all machines by six months.

This is really an exciting time - it is almost as exciting as the original launch. The new management have a serious desire to make the computer succeed. At this point the Company needs the support of you, the end user, and it can be safely stated that there has never been a more loyal set of users, whose needs have never been satisfied by the existing company. But after saying this, 90% of you have remained loyal to the computer. The new management recognise this fact and are anxious to try and reach a solution that will provide a service that you deserve.

All of us who are involved with Genpat have pledge our 100% support to the new company, and if necessary, will carry out certain tasks free of charge in order that the new directors, who have as much faith in the machine as we have, and have also put money where their mouth is, have a fair chance of promoting the machine and developing new products.

So, instead of this being a requiem consider it a christening ! And we all wish Geoff Boyd the very best of luck. It would also be nice if you would drop him a line and proclaim your continued support.

Don't forget, anyone who wants to pay us a personal call is most welcome to visit on Saturdays between 11-00am and 4-00pm. If you want to visit on other days, please telephone us first.

Keep on tapping.

P.s. we are still recieving new memberships at a rate of 5 per week which means we are NOT in a declining market.

TM

Subscriptions

IF YOUR MEMBERSHIP NUMBER IS BETWEEN 0 & 1339 (ALL LETTERS) YOU ARE NOW DUE TO RENEW YOUR SUBSCRIPTIONS. TO MAKE SURE OF YOUR NEXT ISSUE PLEASE SEND YOUR SUBSCRIPTION TO REACH US NO LATER THAN THE 19th March 86

GENPAT HIT LIST....

arcade

1.	SEPULCRI SCELERATI	MEGASTAR
2.	SUPA CODER	SYNTAXsoft
3.	USER BASIC	SYNTAXsoft
4.	MANIC MINER	SOFTWARE PROJECTS
5.	ROLLA BEARING	MEGASTAR
6.	JET SET WILLY	SOFTWARE PROJECTS
7.	SMG	MEGASTAR
8.	ESCAPE FROM ZARKOS	MEGASTAR
9.	MEMOSKETCH	SYNTAXsoft
10.	DOWNSTREAM DANGER	MEGASTAR

adventure

1.	SNOWBALL	LEVEL 9
2.	LORDS OF TIME	LEVEL 9
3.	EMERALD ISLE	LEVEL 9
4.	THE KEYS TO TIME	SENTIENT
5.	MURDER AT THE MANOR	SENTIENT

educational

1.	HELLI-MATHS	SENTIENT
2.	MATHS 1	CONTINENTAL
3.	WORDS & PICTURES	CONTINENTAL
4.	SPELLI-COPTER	SENTIENT
5.	FIRST WORDS	CONTINENTAL

High Scores

Can you do better ?

ASTRO-PAC	213,430	Michael Hunt
KNUCKLES	1,147,160	T. Eriksson
CHAMBERBIDS	Completed 4 mins	T. Eriksson
MISSION ALPHATRON	74,240	Gordon Hurd
TAPEWORM	175,900	Richard Franks
TOADD	178,292	Gavin Gaunt and Nicholas Locke
POT HOLE PETE	105,630	Richard Franks
MAXIMA	1,479,710	S. Olander
STAR COMMAND	140,430	Ian Nichols
PHAID	26,000	Sally Street
ORLOIDS	62,400+	M. Hurley
KILOPEDE	82,253	Richard Nash
3D TACHYON FIGHTER	10,700	Lesa Woodger
CONTINENTAL RAIDERS	106,240	Sean Haverty
BLORRO	148,283	Elizabeth Mahon
QUANTUM	14	Ian Cartwright
QOGO 2	205,000	R. Siddall
MINEFIELD	1,500	David Nash
FLUMMOX	239,550	Andrew Miller
TURBO	18,610	Michael Hunt
FATHOMS DEEP	2,390	T. Eriksson
AGROVATOR	875,000+	P. Howard
FIREHOUSE FREDDIE	29,620	T.Eriksson
QOGO	43,960	T. Eriksson
ARCADIANS	25,900	Adrian Joynson
MISSILE COMMAND	27,500	Adrian Joynson
LITTLE DEVILS	34,120	Leslie Banks
FELIX IN THE FACTORY	14,740	Peter Crighton
MUNCHY	7,900	R.Harmer
SON OF PETE	17,233	T. Eriksson
HAWKWARS	16,600	T. Eriksson
ESCAPE FROM ZARCOS	46 Items	PaG Howard
SALTY SAM	40,642	Andrew Johnson
MISSION OMEGA	9,350	R.Harmer
ICEBURG	17,431	Alan Dobson
SNOWBALL	450	P. Crighton
EMERALD ISLE	300/1000	Richard Franks
SUPERMIKE	23.9km	R. Clark
DOODLEBUG	3,820	Andrew Miller
DR. FRANKIE	65,435	J Graham
TARGET ZONE	14,560	P. Howard
MINER DICK	22,520	R. Siddall
JUMPING JACK	20,120	Andrew Miller
SURFACE SCANNER	72,080	T. Eriksson
CAVES OF ORB	421/500	P. Korsten
SEPULCRI SCELERATI	7,925	Andrew Miller GAME COMPLETED
S.M.G.	6,140	Andrew Miller
COMBAT	42,310	Andrew Miller
QUAZZIA	26,660	Andrew Miller
OBLITERATION ZONE	32,670	Alan Dobson
ASTROMILON	3,680	Alan Dobson
CRYSTAL	1,013	Alan Dobson
DRIVE THE CEE-5	7,255	Alan Dobson
THE WALL	39,720	T. Eriksson

Mike Mash has completed Qogo 2 and has quoted the final message - "At last, you have found the Qogan diamonds"

Can you beat these high scores ? Do you have a high score for a game not mentioned above ?

BASIC GRAPHICS Part 7

Michael Gant

In this month's article, I will explain as promised the more advanced sprite commands available on the MTX. Let's begin by defining a 16 by 16 pixel sprite. As this is double the width and height of the sprites we used last time (8 by 8), we must define four times as many patterns. To do this, we use the GENPAT statement under modes 4, 5, 6 and 7. Each function defines a quadrant (or quarter) of the whole shape. Functions 4, 5, 6 and 7 define the top left, bottom left, top right, and bottom right quadrants respectively. Anyway, here is the program to do this:-

```
10 VS 4:CLS
20 GENPAT 4,0,255,255,255,255,255,255,255,255
30 GENPAT 5,0,255,255,255,255,255,255,255,255
40 GENPAT 6,0,255,255,255,255,255,255,255,255
50 GENPAT 7,0,255,255,255,255,255,255,255,255
```

Obviously, in order to see our creation on the screen, we require more program lines than those above. First of all, we must set the number of sprites etc. using the CTLSPR command.

```
60 CTLSPR 2,1:REM One sprite on screen.
70 CTLSPR 6,2:REM Set 16*16 mag 1.
80 CTLSPR 5,1:REM One moving sprite.
```

The only remaining command required is SPRITE which we use to set the position and speed of our sprite.

```
90 SPRITE 1,0,128,96,10,0,15
100 GOTO 100
```

If you RUN the program, you will see a spaceship travelling across the screen from left to right. If an error appears, check your program against this listing and make absolutely certain that you haven't made any spelling mistakes.

Moving on, let's take a look at the MVSPR command. Try adding the following lines to the program above.

```
65 CTLSPR 4,1
95 MVSPR 8,1,0
```

If you RUN the program again, you will see that the spaceship leaves a trail behind it. This is all done automatically by the computer. You will notice that we have only used one sprite. This seems quite a waste when we consider that the MTX can display 32 sprites at once. The following program sets up a vertical row of 12 sprites. After pausing for a few seconds, the sprites move off at random speeds and directions. To save space, I will only explain the new commands and routines.

```
10 VS 4:CLS
20 GENPAT 4,0,255,255,255,255,255,255,255,255
30 GENPAT 5,0,255,255,255,255,255,255,255,255
40 GENPAT 6,0,255,255,255,255,255,255,255,255
50 GENPAT 7,0,255,255,255,255,255,255,255,255
60 CTLSPR 2,12
70 CTLSPR 5,12
80 CTLSPR 3,12
90 CTLSPR 6,2
100 FOR F=1 TO 12
110 SPRITE F,0,128,(F-1)*16,0,0,INT(RND*15)+1
120 NEXT F
130 PAUSE 2000
140 FOR F=1 TO 12
150 ADJSPR 4,F,INT(RND*100)-50:ADJSPR 5,F,INT(RND*100)-50
160 NEXT F
```

```
170 GOTO 170
```

Lines 60-90 use the CTLSPR command to set up 12 sprites, 12 moving sprites, 12 circling sprites and 16 by 16 size mag 1 respectively.

Line 100 sets up a FOR-NEXT loop to count from 1 to 12 (once for each sprite). Line 110 displays sprite number F, with pattern 0, X coordinate 128 (half way across the screen), Y coordinate equal to F-1 multiplied by 16 (why not try working out why we need this?), X and Y speeds set to 0, and a random colour between 1 and 15. The NEXT statement in Line 120 finishes off this loop.

Line 130 causes the program to pause for 2 seconds before continuing.

The second FOR-NEXT loop begins at line 140. It is used to set the sprites in motion using the ADJSPR command. Each sprite requires two ADJSPR commands. One sets the X speed while the other sets the Y speed. The formula used in both cases generates a random number between 50 and -50. This ensures that the sprites move in all directions.

Line 170 causes the computer to loop around until the break key is pressed. This shows clearly that all sprites are in fact moving automatically.

Well, that covers all the sprite functions available on the Memotech. So, to round off with, here is a simple drawing program that allows you to draw on the screen using a pen. The pen is of course a sprite that has been created as a plot sprite i.e. a pixel is plotted at the centre as it moves.

```
1 LET X=128: LET Y=96
10 VS 4: CLS
12 LET P=1
15 CTLSPR 0,1
20 CTLSPR 2,1
30 CTLSPR 3,1
35 CTLSPR 4,1
40 CTLSPR 5,1
50 CTLSPR 6,2
60 GENPAT 4,0,0,32,80,40,20,10,7,3
70 GENPAT 5,0,0,0,0,0,0,0,0,0
80 GENPAT 6,0,0,0,0,0,0,0,0,0
90 GENPAT 7,0,0,0,0,0,0,0,0,0
100 SPRITE 1,0,128,96,0,0,15
110 LET A=ASC(INKEY$)
120 IF A=26 THEN  LET P=1-P: PAUSE 100: GOTO 200
130 IF A=11 THEN  LET Y=Y+1: GOTO 200
140 IF A=8 THEN  LET X=X-1: GOTO 200
150 IF A=25 THEN  LET X=X+1: GOTO 200
160 IF A=10 THEN  LET Y=Y-1: GOTO 200
170 IF A=9 THEN  LET X=X-1: LET Y=Y+1: GOTO 200
180 IF A=127 THEN  LET X=X+1: LET Y=Y+1: GOTO 200
190 IF A=21 THEN  LET X=X-1: LET Y=Y-1: GOTO 200
195 IF A=12 THEN  LET X=X+1: LET Y=Y-1: GOTO 200
200 ADJSPR 2,1,X: ADJSPR 3,1,Y
205 IF P=1 THEN  ATTR 2,0 ELSE  ATTR 2,1
206 MVSPR 8,1,0
210 GOTO 110
999 GOTO 105
1000 PRINT ASC(INKEY$): GOTO 1000
```

As an exercise, work through the program step by step to find out exactly how it works. See you again next month!! ★

The History of Computing (I think!) Part 2

P Knaggs

The next bit of interest to us happened around 1880 when the Americans took a census of the entire population. (They do one every 10 years). The data from this census was not collated and presented into some kind of report for a further eight years after the census had been taken. Someone worked out that at that rate they would not have all the data from the 1890 census until 1902. The Government decided that this was not a good idea, and so they offered a competition for a method of speeding up the calculations. There were three entrants in the competition. Two of whom used sticks for counting, and then there was Professor Herman Hollerith, who used their idea of Jacquard's cards. It was Herman who won the competition and received an order for his device, known as a tabulator. He instantly left the Government Statistics Department and formed his own company, The Tabu 0 or a 1). The reason for this was quite simple. It is extremely difficult to design machinery to work in DENERY (with 10 different states or digits), it is much simpler to design something which can be said to be On or Off (the answers to a di-continous within 10 minutes rather than several days. Also of course being that it took less time then it was overall cheaper as not so many people have to be employed for so long to do the same amount of information.

In fact using this tabulator machine the 1890 census was completed by 1894, taking four years less than the one before it, and it contained much more information.

After this Hollerith went and improved his design. This new tabulator could handle up to 80 multiple-chosen questions of up to 10 answers each. Much better than the old 8 di-continue questions. Because of this the card had to be big enough to handle all the information. It was decided that a card the same size as a one dollar bill was adequate for all this and a little more, and was a simple usable size. This kind of machine was used for the 1900 census and was a great improvement over the tabulator, although it was much more expensive.

As Herman was developing this new system, he was costing his company quite a lot of money. In fact they had a serious cash flow problem. It was at this point that an Office Furnishing Company became interested and through a series of mergers and takeovers bought out Hollerith's company. The new parent company then financed Hollerith to finish the product; in the meantime they changed their name (so as to incorporate the new department to the company) to the International Business Corporation (better known as IBM).

Just about every one was using these tabulators in some form or another by 1920.

The valve was invented. At first this meant nothing except when you look into it. A valve has two state (like Binary) and can hold its state for a while (thus providing memory). This provided a rather good method of storing Binary digits.

Now we come to a rather well documented part of our history, but is distintly lacking in the importance that calculation machines had to play in it. This is the Second World War. After eight years of development in Harard College, the Automatic Sequence Controlled Calculator was completed (this was 1944). This was a development of Hollerith's ideas and input into the machine was by punched cards (known as Hollerith Cards). The answers came out on a typewriter by electro-mechanical means.

One of the project leaders (one Howard Alken, who was teamed up with IBM thought out this development) happened to discover Lady Lovelace(s notes of the old Babbage days. Up to this point everybody had forgotten about poor old Babbage. This early computing machine was used in Harvad for a further ten years. Several years later (1946) over in the University of Pennsylvania they were building the Electronic Numerical Integrator and Calculator, known more simply as ENIAC. This was essentially a calculator which could be programmed by means of switches and cables. ENIAC was a massive creation consisting of 18,000 valves and weighing some eral days. Also of course being that it took less time then it was overall cheaper as not so many people have to be em effort in performing calculations in the design of the Atom Bomb.

Back in to England and 1943. The Researchers at Manchester University developed a calculating machine that could break the German coded signals, and produce an English translation. This was known as COLLOSUS (as it was a Colloe machine and it provided an eye into the German affairs).

Well, with the war over and everyone pleased with themselves because that won, a Dr John von Neumann (a German who had gone to the U.S. before the war) came up with a rather startling revelation. He thought if we can store data in memory, then why can we not store a programme in a certain area of memory and then have the data in another area. For this to happen you would need a great deal more memory. However, you would not need to feed in the programme to interpret the data every time you enter a new number, all you would have to do is enter the data. This would mean that the entry of calculations (and indeed the calculations) would be speeded up somewhat. These ideas have proved to be so popular that ever since all computing aids have worked in this manner.

We now enter the field of commercially available computing machines. This is where we start given generation numbers. So on to the first commercially available machine (known as the first generation). Two of the designers of the ENIAC went off and formed their own company, the UNIVAC Corporation. Messrs. Eckert and Mauchly were the first to make commercially available computers (The UNIVAC 1 in 1951). In the same year the University of Manchester in collaboration with Ferranti produced a device known as the Mark 1 . This was then produced into a small series of computing machines. IBM were not left out of the act either; they had two big hits with the IBM-701 and IBM-650. Unfortunately all of these machines had several things in common:

a. They were all large
b. They worked off Valves - Large and unreliable things
c. Each had a special way of giving instruction, and different sets of instruction. Hence to train a person to control one would cost a great deal of money, and he would then be dedicated to that machine. If you were to replace it with a new, more capable machine (from the same company) you would have to re-train all of your staff at great expense.

At the end of the 1940's a William Shockley, working for Bell Labs invented this thing he called a transistor. This was small, easily produced, and reliable. By the late 1950's it managed to revolutionise digital computing by replacing the valves, giving rise to smaller, reliable computing machines. These transistor computers are known as the second generation.

IBM had the advantage over all the others in this field. This was due to their marketing policy. In fact one IBM the IBM-1401 out-sold all of the other makes put together. As part of their sales drive IBM decided that this idea of having to re-train staff all of the time was not that hot. So they developed a high level language. That is to say, they developed a way of telling it things that would work on any of their machines. As the instructions were relatively English like (as compared to all that had gone before) it was called a language. The first such language was FORTRAN. However, soon after came some more of the likes of COBOL and ALGOL. These were problem-orientated, for example COBOL is very useful for handling business data and giving reports, whilst FORTRAN is very good for mathematicians.

COBOL is short for COmmon Business Orientated Language.
FORTRAN for FORmular TRANslater.

The advantages of such languages is that you could spend less time in gearing your system to a particular application as there was a language available to do that for you. Indeed for most of the common uses for computers all you had to do was go out and buy some programmes that could work it in the way you want. It also meant that you could buy in a trained programmer, rather than having to pay for his training yourself. And so you found people who were specialising in making these machines do certain types of things. Thus the computer specialist was born.

The American space race gave rise to a method of storing many transistors (and logic gates) onto one small piece of silicon. These pieces of silicon have become known as Integrated Circuits or IC Chips (IC for integrated circuits, Chips from the fact that silicon was derived from a Chip of rock).

The introduction of IC's in the 1960's herealded the third generation of computing machines. IC's dramatically reduced the size of computer circuitry, thereby making the whole thing smaller, and thus cheaper. It has since been found that an IC is very reliable and so the reliability of these things increased. Further characteristics of the genre included time-sharing operating system in which several users could access a single computer system simultaneously. So one company could buy a big

computer system, but all of the departments could use it. This type of system became known as a MAIN-FRAME System (due to the fact that it was stored in one main framework). These computer systems also proved to be very cost effective.

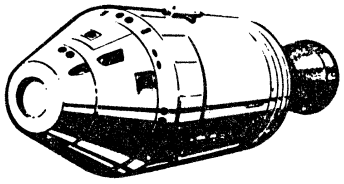
The transistor also allowed a thing called a MINI-COMPUTER to be introduced . This was a computing machine but was not as capable as the big Main-frame's ones that the company were using up to now. The Digital Equipment Corporation's computer released in 1964 was one such device (the DEC PDP-8). It was also a move away from the now IBM dominated market of computing machines. These new MINI's were now in the reach of a single departmental budget. They were also capable of up to 71000 times faster and more accurate than the old ENIAC system of eighteen year previous. They also have the advantage that people already know how to control them due to these standard languages (like FORTRAN and COBOL).

As the amount of Transistors that could be mounted on to a IC then the Micro-Computer became possible (by mid 1970's). These were indeed a step down from their big brothers but were a great deal cheaper and were in the price range of the more moderate businesses (selling at some #2000 for a standard business system).

Until chips with a large scale integration (as that is what they call packing transistors on to this bit of rock), LSI or even very large scale integration (ULSI) are now considered to be fourth generation (we are now up to ULSI, Ultra-Large scale integration). It is this technology which has brought about the growth of affordable and powerful personal computers such as the Micro-VAX and WICAT, as well as cheaper home machines. The recent introduction of full computers on a single chip (such as the Inmos Transputer) may be the start of a new fifth generation, but most commentators feel that this next generation of computers will be typified by parallel processing and greater advances in machine intelligence. Much of the current research in computing concerns artificial intelligence and expert systems (such as the newly installed Verchical Monitering System, at Godstone on the M25). The US Government's planned Strategic Defence Initiative (commonly known as the Star Wars project, oops, sorry we can't call it that any more, can we, I meant the SDI) involves the development and production of very sophisticated computers with a large degree of intelligence, recognition, and decision-making capabilities. These may be machines of the fifth generation.

However, the best is yet to come. (We will probably still be waiting when I'm dead and gone).

P. Knaggs ★



ANY ONE TITLE - 2.50
BUY THREE AND GET ON FREE!
TAKE THE WHOLE LOT FOR JUST 19.95!

1. REVERSI 2. TOADO 3. DRAUGHTS 4. BACKGAMMON

5. MUSIC PAD 6. GRAPHICS 7. LITTLE DEVILS 8. MAN FROM GRANNY

9. QOGO 10. SPELLI-COPTER 11. HELI-MATHS 12. MISSION ALPHATRON

CONTACT AND CHEQUE TO MR. D. S. LAM, 105 GEARY ROAD, LONDON NW10 1HS

The ED Text Editor

ED is a text editor supplied on your CP/m system disc. With ED you can create or edit text files. However, be warned, ED is only an elementary editor, and work that involves a significant amount of editing should be undertaken with NewWord. In the early days customers were not treated to special package deals, and we all know how much Word Processors such as Wordstar cost. ED is a leftover from the days when DR (Digital Research) took pity on their customers and supplied the programme on system disc.

To initiate ED you must type:-
ED FILE NAME

Don't make the common mistake of trying to initiate the editor by typing ED as this will result in an error message being displayed.

ED FILE NAME. EXT
ED TEST. DAT
ED NODDY. TXT

If the file does not exist or is not on your current disc then ED will create a new file named from the initiation sequence.

When you create a new file ED stores the data in the TPA (Transient Programme Area). You should be aware that ED does not operate in the same manner as NewWord. It doesn't automatically store the file to disc. You must use the relevant commands to save the file.

If you are editing a file, ED moves the file from disc into the TPA. At this point you can then use the editing commands to insert, delete, or change data within the file. On terminating the session you must instruct ED to save the edited file to disc. The altered file now becomes the main file and the original is saved as a back-up file with the BAK extension. This means that you can always return to the status quo by renaming the back-up file.

FILES ON DISC BEFORE EDIT SESSION:

ED . COM
TEST . DAT.

FILES ON DISC AFTER EDIT SESSION

ED . COM
TEST . COM .... EDITED FILE
TEST . BAK. ... ORIGINAL FILE

Because ED is what is commonly termed a character editor you can only work on one character at a time. As you enter text into the buffer, ED keeps a note of your position in memory with an internal CHARACTER POINTER. The CHARACTER POINTER is important and can be imagined as an arrow which always comes to rest between two characters. For example:

NOW IS THE TIME FOR
^ ....CHARACTER POINTER HERE

If we give the command: D ED would delete the character to the RIGHT of the character pointer (in this case H).

If we gave the command: I ED would insert characters to the RIGHT of the character (in this case H).

The command minus -D would delete the character to the left of the character pointer (the T).

The editor has a set of commands to move the character pointer anywhere within the file. You can also display the current line and find exactly where the character pointer is positioned.

To aid with your editing ED can be made to display line numbers. When you first enter ED the line numbering mode is off to initiate line numbering you must type :V.

It can be turned off by typing minus V.

Remember, that line numbers, even though they have been used within your file, are not saved with the data at the end of the session. If you carry out further editing on the file, line numbers will need to be re-initialised by typing :V.

The following is a list of characters that can be used within ED.

^ = CTRL KEY.

THIS MUST BE HELD DOWN WHILE PRESSING THE RELEVANT KEY.

^ C = PRESS AND HOLD DOWN CTRL KEY. NOW PRESS C.

THIS SEQUENCE IS THE SAME WHENEVER YOU SEE ^

^C = Reboot to system (warm re-boot)
"BREAK" also causes a re-boot

^E = Move cursor to next line without generating a "carriage return"

^H = Backspace one character (this is a hard backspace it erases the character).

^I = Tab cursor to next 7 columns

^J = Return

^M = Same

^L = Return used with search and substitute commands

^P = Echo everything to printer

^R = Retype current line

^S = Halt when display long files. Any key to continue

^U = Erase current line

^X = Backspace to beginning of current line and erase to end of line

^Z = Terminate the I command

COMMANDS AVAILABLE UNDER ED

Note X = Integer number between 0 to 65535

XA = Append X number of lines from source file to edit buffer
Omitting X will append one line
X = 0 will fill half of buffer
X = # will append 65535 lines

XB = Move character pointer to beginning of buffer
-XB Move character pointer to end of buffer
XC Move character pointer forward x characters
-XC Move character pointer backward x characters
NOTE: Carriage return counts as two characters

XD Delete x number of characters from character pointer (includes CP).

-XD Delete x number of characters to the left of the character pointer (not including the CP).

Omitting X = delete/Character to left or right of character pointer E.
Save all buffered text and source file.
Rename old file to backup file and terminate session.

XFSTRING = Find the string x number of times . Omitting x = find string once.
The character pointer is moved to the end of the found string

H = Perform an E command but don't terminate session. Save file updates and return to edit mode.

I = Insert a new line after the character pointer. The CP then moves to end of last line inserted.

Capital I = All text must be inserted into buffer in upper case. Small i = insert all text in lower case.

^I String Z = insert this string to the right of character pointer. CP moves to end of last character inserted.

XJ String A^Z String B ^ZString C
Find String A and insert String B at the end of String A. Then delete all characters up to String C (but not including C). Character pointer is positioned at the start of String C.

XK = Delete x number of lines from, and including, character following character pointer on current line.

-XK Delete x number of previous lines including current characters to left of CP on current line.

XL Move character pointer to beginning of current line and down x lines.

-XL Move CP backward x number lines
If x = 0 Move character pointer to beginning of current line.

XM String Repeat String of ED commands x number of times.

X = 0 Repeat until error.

XN String Search until the xth occurence of the String is found in the buffer or source file.
Note: N appends lines from source file until String is found. Character pointer moves to end of the located String.

O Forget this session and return to original file

XP Move character pointer to beginning of page and print x number pages following the CP (1 page = 24 lines)

-XP As above but print x number of pages before the CP.

X = 0 = Print current line and next 23 lines

Q = Quit. Don't make any file alterations.
ED returns to system without altering temporary $$$ file or current source file.
*** However, if there is a BAK file with the same name ED deletes it.

R = Read from $$.LIB file and insert the lines after the character pointer. CP moves to end of inserted lines.

R <File Name> Read from file name. LIB and insert as above.

XS old string ^Z new String = Find the old string to the right of the character pointer and replace with new string. Do it x number of times.

XT. Display the following x number of lines including the current line.

-XT Display the previous x number of lines not including the current line

If x = 1 or is omitted = Display to the right of the CP to end of line.

X = 0 = Display characters on current line up to character pointer.

U = All input chracters are translated into upper case.

-U = Turn off above command.

V = Turn line numbering on.
-V = Turn line numbering off.

OV = Display number of free bytes left in buffer and total size of buffer
Display shows: Free bytes/total byte size.

XW = Write out the following x number of lines including current line to a temporary output file with $$$ attribute.

If X is omitted write out the current line only.

xX Copy the following x number of lines of text to the file $$$. LIB.
X = 0 will delete the $$$.LIB file.

X: Move character pointer to beginning of line number x.

X1::x2 Specify a range of line numbers X1 to X2.

Now you have memorised all of ED's commands let's try creating a test file.

Before we start copy ED on to your empty disc and then you are sure that you don't do any damage to your original copy.

Now type: ED SAMPLE.TXT and then in `RET".

ED will now display the message NEW FILE, and after a short pause you will be faced with a display as follows

:*

The asterisk is ED's way of telling you that the editor is in the command mode and is awaiting your instructions.

Before we can insert any text into the buffer we must tell ED that we want to use the INSERT MODE.

I`RET"

ED now displays a line number

1:

You can now start typing in the text. Remember, if you make any mistakes during your typing session, you can use the CP/m standard control characters to delete your mistakes. You will notice that ED uses a HARD BACK SPACE and deletes the character as it moves left. To alter single characters in the middle of a line, for example, without deleting the other characters you need to enter the line into the buffer, complete with mistakes, and then use the commands to delete and insert single characters.

If your mistake was made in line 3 you could do the following from the command mode:

*B	Move to top of buffer
*4T	Display the next 4 lines
*6C	Move CP to the 6th character
*T	Display from character pointer to confirm correct position
*D	Delete character to right of CP (delete mistake)
*I	Insert character required
^Z	return to command mode

You must always return to command mode to issue commands when in insert mode.

Use the T command to find out exactly where the CP is situated.

On the Memotech the BS key works as expected. The left arrow key also performs correctly. However, all the other cursor keys have unpredictable results and the Home Key performs a ^Z.★

Light Riders

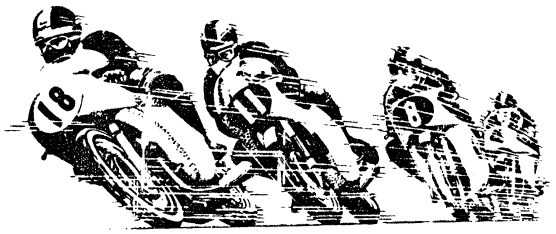
Barry Young

LIGHT RIDERS is a games program for the MTX 512/500 and is very similar in concept to "TRON" which is to be found in the arcades. You have to avoid hitting the light trails left by your opponent, and, obviously, you must not collide with the obstacles in the games area.

Full instructions are included within the game and you will be pleased to note that this is a two player game.

Type in the program as listed but OMIT LINE 40 until you are sure the game operates correctly. Once you have a finished version, include line 40 and then type RUN. This will save you an auto-run version of the program.

MTX 500 owners type in the program exactly as listed BUT your MEMORY addresses will be +4000H so that the machine code should start at #83D8.★



```
10 REM ##################################
20 REM ######### LIGHT RIDERS #########
30 REM ##################################
40 SAVE "Light-Riders"
100 VS 4: PAPER 4: COLOUR 4,4: INK 15: CLS
110 CSR 5,0: PRINT "L I G H T  R I D E R S"
130 CSR 10,5: INK 7: PRINT "INSTRUCTIONS"
140 CSR 1,7: PRINT " Race your Light-Bikes around  the Games Arena, avoiding your  opponent, light trails and  walls and obstacles."
150 PRINT : PRINT "  LEFT PLAYER     RIGHT PLAYER"
160 PRINT "     use             use"
170 PRINT " Left joystick   Right joystick"
180 PRINT "     or              or "
190 PRINT "   Z,C,B,M        CURSOR KEYS"
200 PAPER 9: INK 4: CSR 11,18: PRINT "GOOD LUCK"
210 CSR 4,20: PRINT "Author Barry Young. '85"
220 GENPAT 3,1,0,16,40,255,40,16,0,0
230 GENPAT 3,2,0,8,20,255,20,8,0,0
240 GENPAT 3,3,16,16,56,84,56,16,16,16
250 GENPAT 3,4,16,16,16,56,84,56,16,16
260 GENPAT 1,129,129,66,153,60,165,90,36,60
270 GENPAT 1,130,127,191,223,239,247,251,253,254
280 GENPAT 1,131,254,253,251,247,239,223,191,127
290 GENPAT 1,132,255,255,255,255,255,255,255,255
300 INK 15: CSR 4,22: PRINT "PRESS ANY KEY TO START"
310 IF INKEY$="" THEN  GOTO 310
320 PAPER 4: INK 15: COLOUR 4,1: CLS
1000 CODE

43D8 INIT:   RST 10
43D9         DB £4C
43DA         LD HL,MESS1
43DD         LD DE,£0106
43E0         CALL MESS
43E3         LD HL,MESS2
43E6         LD DE,£0403
43E9         CALL MESS
43EC         LD HL,MESS3
43EF         LD DE,£0704
43F2         CALL MESS
43F5         LD HL,MESS4
43F8         LD DE,£0804
43FB         CALL MESS
43FE         LD HL,MESS5
4401         LD DE,£0904
4404         CALL MESS
4407         LD HL,MESS6
440A         LD DE,£0A04
440D         CALL MESS
4410         LD HL,MESS7
4413         LD DE,£0B04
4416         CALL MESS
4419         LD HL,MESS8
441C         LD DE,£0F04
441F         CALL MESS
4422 SPRCL:  LD A,£C0
4424         OUT (2),A
4426         LD A,£81
4428         OUT (2),A
442A         LD DE,£3F00
442D         LD A,E
442E         OUT (2),A
4430         LD A,D
4431         OR 64
4433         AND 127
4435         OUT (2),A
4437         LD B,128
4439 CLATT:  XOR A
443A         CALL OUT
443D         DEC B
443E         LD A,B
```

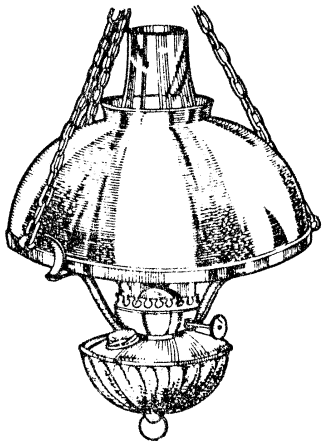
```
443F        JR NZ,CLATT
4441        XOR A
4442 SETSC: LD (LSCORE),A
4445        LD (RSCORE),A
4448 INP:   LD A,£FE
444A        OUT (5),A
444C        IN A,(5)
444E INP1:  CP 254
4450        JR NZ,INP3
4452        LD BC,2000
4455        LD (SPEED),BC
4459        JR GAME
445B INP3:  CP 253
445D        JR NZ,INP5
445F        LD BC,500
4462        LD (SPEED),BC
4466        JR GAME
4468 INP5:  CP 251
446A        JR NZ,INP2
446C        LD BC,2
446F        LD (SPEED),BC
4473        JR GAME
4475 INP2:  LD A,253
4477        OUT (5),A
4479        IN A,(5)
447B        CP 253
447D        JR NZ,INP4
447F        LD BC,1000
4482        LD (SPEED),BC
4486        JR GAME
4488 INP4:  CP 251
448A        JR NZ,INP
448C        LD BC,250
448F        LD (SPEED),BC
4493 GAME:  RST 10
4494        DB £4C
4495 BORDER: RST 10
4496        DB £A5,2,0,16,255,16
449C        DB £A5,2,255,16,255,183
44A2        DB £A5,2,255,183,0,183
44A8        DB £85,2,0,183,0,16
44AE CROWDT: LD HL,CROWDDB
44B1        LD DE,£0000
44B4        CALL MESS
44B7 CROWDB: LD HL,CROWDDB
44BA        LD DE,£1600
44BD        CALL MESS
44C0        LD HL,PLAYERS
44C3        LD DE,£1700
44C6        CALL MESS
44C9        CALL PRSCR
44CC BLOCKS: LD HL,BLOCK1
44CF        LD DE,£0508
44D2        CALL MESS
44D5 B2:    LD HL,BLOCK2
44D8        LD DE,£0608
44DB        CALL MESS
44DE B3:    LD HL,BLOCK1
44E1        LD DE,£0516
44E4        CALL MESS
44E7 B4:    LD HL,BLOCK2
44EA        LD DE,£0616
44ED        CALL MESS
44F0 B5:    LD HL,BLOCK1
44F3        LD DE,£0A0F
44F6        CALL MESS
44F9 B6:    LD HL,BLOCK2
44FC        LD DE,£0B0F
44FF        CALL MESS
4502 B7:    LD HL,BLOCK1
4505        LD DE,£0F08
4508        CALL MESS
450B B8:    LD HL,BLOCK2
450E        LD DE,£1008
4511        CALL MESS
4514 B9:    LD HL,BLOCK1
4517        LD DE,£0F16
451A        CALL MESS
451D B10:   LD HL,BLOCK2
4520        LD DE,£1016
4523        CALL MESS
4526 INITPOS: LD A,32
4528        LD (LX),A
452B        LD A,R
452D        ADD A,50
452F        LD (LY),A
4532        LD A,1
4534        LD (LDIR),A
4537        LD A,223
4539        LD (RX),A
453C        HALT
453D        LD A,R
453F        ADD A,50
4541        LD (RY),A
4544        LD A,2
4546        LD (RDIR),A
4549        CALL LSPROUT
454C        CALL RSPROUT
454F        CALL CHEER
4552        CALL SOUNDON
4555 GETRTJ: LD A,£F7
4557 LEFTR: CALL STROBE
455A        JR NZ,RIGHTR
455C        LD A,2
455E        LD (RDIR),A
4561 RIGHTR: LD A,£EF
4563        CALL STPOBE
4566        JR NZ,UPR
4568        LD A,1
456A        LD (RDIR),A
456D UPR:   LD A,£FB
456F        CALL STROBE
4572        JR NZ,DOWNR
4574        LD A,3
4576        LD (RDIR),A
4579 DOWNR: LD A,£BF
457B        CALL STROBE
457E        JR NZ,TESTR
4580        LD A,4
4582        LD (RDIR),A
4585 TESTR: LD A,(RDIR)
4588        CP 1
458A        JR NZ,LR
458C        LD A,(RX)
458F        INC A
4590        LD (RX),A
4593        JP POINTR
4596 LR:    CP 2
4598        JR NZ,UR
459A        LD A,(RX)
459D        DEC A
459E        LD (RX),A
45A1        JP POINTR
45A4 UR:    CP 3
45A6        JR NZ,DR
45A8        LD A,(RY)
45AB        INC A
45AC        LD (RY),A
45AF        JP POINTR
45B2 DR:    CP 4
45B4        JR NZ,POINTR
45B6        LD A,(RY)
45B9        DEC A
45BA        LD (RY),A
45BD POINTR: LD A,(RX)
45C0        LD (POINTX),A
45C3        LD A,(RY)
45C6        LD (POINTY),A
45C9        CALL POINT
45CC        CP 01
45CE        JP Z,CRASHR
45D1        LD A,(RX)
45D4        LD (PLOTX),A
45D7        LD A,(RY)
45DA        LD (PLOTY),A
45DD        CALL PLOT
45E0        CALL RSPROUT
45E3 GETLTJ: LD A,127
45E5        OUT (5),A
45E7        IN A,(5)
45E9 LEFTL: CP 254
45EB        JR NZ,RIGHTL
45ED        LD A,2
45EF        LD (LDIR),A
45F2 RIGHTL: CP 253
45F4        JR NZ,UPL
45F6        LD A,1
45F8        LD (LDIR),A
45FB UPL:   CP 251
45FD        JR NZ,DOWNL
45FF        LD A,3
4601        LD (LDIR),A
4604 DOWNL: CP 247
4606        JR NZ,TESTL
4608        LD A,4
460A        LD (LDIR),A
460D TESTL: LD A,(LDIR)
4610        CP 1
4612        JR NZ,LL
4614        LD A,(LX)
4617        INC A
4618        LD (LX),A
461B        JP POINTL
461E LL:    CP 2
4620        JR NZ,UL
4622        LD A,(LX)
4625        DEC A
4626        LD (LX),A
```

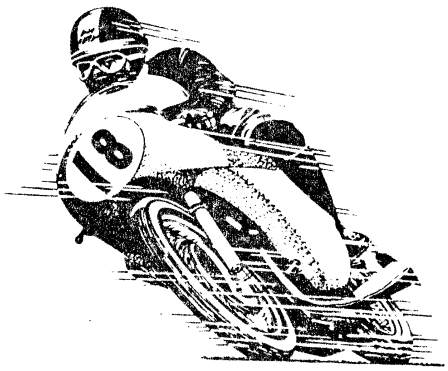
```
4629          JP POINTL
462C UL:      CP 3
462E           JR NZ,DL
4630          LD A,(LY)
4633          INC A
4634          LD (LY),A
4637          JP POINTL
463A DL:      CP 4
463C          JR NZ,POINTL
463E          LD A,(LY)
4641          DEC A
4642          LD (LY),A
4645 POINTL:  LD A,(LX)
4648          LD (POINTX),A
464B          LD A,(LY)
464E          LD (POINTY),A
4651          CALL POINT
4654          CP 1
4656          JP Z,CRASHL
4659          LD A,(LX)
465C          LD (PLOTX),A
465F          LD A,(LY)
4662          LD (PLOTY),A
4665          CALL PLOT
4668          CALL LSPROUT
466B PAUSIN:  CALL £0079
466E          CP 13
4670          JR NZ,DELAY
4672 PAUSE:   HALT
4673          HALT
4674          HALT
4675          HALT
4676          CALL £0079
4679          CP 13
467B          JR NZ,PAUSE
467D DELAY:   LD BC,(SPEED)
4681 DELOOP:  DEC BC
4682          XOR (IX+0)
4685          LD A,B
4686          OR C
4687          JR NZ,DELOOP
4689 BACK:    JP GETRTJ
468C MESS:    LD (COORD),DE
4690          RST 10
4691          DB £03,3
4693 COORD:   DW 0000
4695 MLOOP:   LD A,(HL)
4696          CP 0
4698          RET Z
4699          LD (CHAR),A
469C          RST 10
469D          DB £01
469E CHAR:    DB 0
469F          INC HL
46A0          JR MLOOP
46A2          RET
46A3 OUT:     OUT (1),A
46A5          RET
46A6 PRSCR:   LD A,(LSCORE)
46A9          CP 10
46AB          JP Z,LEFTWIN
46AE          ADD A,48
46B0          LD (LDISP),A
46B3          LD DE,£170C
46B6          LD HL,LDISP
46B9          CALL MESS
46BC          LD A,(RSCORE)
46BF          CP 10
46C1          JP Z,RIGHTWIN
46C4          ADD A,48
46C6          LD (RDISP),A
46C9          LD DE,£1713
46CC          LD HL,RDISP
46CF          CALL MESS
46D2          RET
46D3 LSPROUT: LD DE,£3F00
46D6          LD A,E
46D7          OUT (2),A
46D9          LD A,D
46DA          OR 64
46DC          AND 127
46DE          OUT (2),A
46E0          LD A,(LY)
46E3          CPL
46E4          SUB 68
46E6          CALL OUT
46E9          LD A,(LX)
46EC          SUB 3
46EE          CALL OUT
46F1          LD A,(LDIR)
46F4          CALL OUT
46F7          LD A,8
46F9          CALL OUT
46FC          RET
46FD RSPROUT: LD DE,£3F04
4700          LD A,E
4701          OUT (2),A
4703          LD A,D
4704          OR 64
4706          AND 127
4708          OUT (2),A
470A          LD A,(RY)
470D          CPL
470E          SUB 68
4710          CALL OUT
4713          LD A,(RX)
4716          SUB 3
4718          CALL OUT
471B          LD A,(RDIR)
471E          CALL OUT
4721          LD A,11
4723          CALL OUT
4726          RET
4727 CHEER:   LD D,150
4729 UNFIN:   LD HL,£2000
472C          CALL ROW
472F          LD HL,£3500
4732          CALL ROW
4735          DEC D
4736          LD A,D
4737          JR NZ,UNFIN
4739 PLAIN:   LD HL,£2000
473C          CALL NORMAL
473F          LD HL,£3600
4742          CALL NORMAL
4745          CALL SOUNDOFF
4748          RET
4749 ROW:     LD A,L
474A          OUT (2),A
474C          LD A,H
474D          OR 64
474F          AND 127
4751          OUT (2),A
4753          LD B,255
4755 VOL:     LD A,R
4757          SLA A
4759          SLA A
475B          SLA A
475D          SLA A
475F          ADD A,4
4761          CALL OUT
4764          DEC B
4765          LD A,B
4766          JR NZ,VOL
4768          RET
4769 NORMAL:  LD A,L
476A          OUT (2),A
476C          LD A,H
476D          OR 64
476F          AND 127
4771          OUT (2),A
4773          LD B,8
4775 NORM1:   LD C,32
4777          LD HL,FANCOL
477A NORM2:   LD A,(HL)
477B          SLA A
477D          SLA A
477F          SLA A
4781          SLA A
4783          ADD A,4
4785          CALL OUT
4788          INC HL
4789          DEC C
478A          LD A,C
478B          JR NZ,NORM2
478D          DEC B
478E          LD A,B
478F          JR NZ,NORM1
4791          RET
4792 STROBE:  OUT (5),A
4794          IN A,(5)
4796          CP 127
4798          RET
4799 SOUNDON: XOR A
479A          LD (£FE14),A
479D          LD HL,500
47A0          LD (£FE16),HL
47A3          LD A,200
47A5          LD (£FE18),A
47A8          CALL £8F6
47AB          LD A,1
47AD          LD (£FE14),A
47B0          LD HL,750
47B3          LD (£FE16),HL
47B6          LD A,200
47B8          LD (£FE18),A
47BB          CALL £8F6
47BE          RET
47BF SOUNDOFF:LD B,2
47C1 QUIET:   LD A,B
47C2          LD (£FE14),A
47C5          LD HL,0000
47C8          LD (£FE16),HL
47CB          LD (£FE18),HL
47CE          PUSH BC
47CF          CALL £8F6
47D2          POP BC
47D3          DEC B
47D4          LD A,B
47D5          CP 255
47D7          JR NZ,QUIET
47D9          RET
47DA POINT:   RST 10
47DB          DB £85,27,67
47DE POINTX:  DB 0
47DF POINTY:  DB 0
47E0          DB 1
47E1          LD A,(£FE1A)
47E4          RET
47E5 PLOT:    RST 10
47E6          DB £83,1
47E8 PLOTX:   DB 0
47E9 PLOTY:   DB 0
47EA          RET
47EB CRASHR:  CALL SOUNDOFF
47EE          CALL BANG
47F1          CALL SOUNDOFF
47F4          CALL CHEER
47F7          LD A,(LSCORE)
47FA          INC A
47FB          CP 10
47FD          JP Z,LEFTWIN
4800          LD (LSCORE),A
4803          JP GAME
4806 CRASHL:  CALL SOUNDOFF
4809          CALL BANG
480C          CALL SOUNDOFF
480F          CALL CHEER
4812          LD A,(RSCORE)
4815          INC A
4816          CP 10
4818          JP Z,RIGHTWIN
481B          LD (RSCORE),A
481E          JP GAME
4821 LEFTWIN: RST 10
4822          DB £4C
4823          LD HL,LWINMESS
4826          LD DE,£0A09
4829          CALL MESS
482C          LD HL,STARS
482F          LD DE,£0B09
4832          CALL MESS
4835          LD HL,CROWDDB
4838          LD DE,0000
483B          CALL MESS
483E          LD HL,CROWDDB
4841          LD DE,£1600
4844          CALL MESS
```

```
4847            CALL CHEER
484A            CALL SOUNDOFF
484D            JP INIT
4850            RET
4851 RIGHTWIN:RST 10
4852            DB £4C
4853            LD HL,RWINMESS
4856            LD DE,£0A09
4859            CALL MESS
485C            LD HL,STARS
485F            LD DE,£0B09
4862            CALL MESS
4865            LD HL,CROWDDB
4868            LD DE,0000
486B            CALL MESS
486E            LD HL,CROWDDB
4871            LD DE,£1600
4874            CALL MESS
4877            CALL CHEER
487A            CALL SOUNDOFF
487D            JP INIT
4880 BANG:      LD BC,1500
4883 REPT:      LD A,0
4885            LD (£FE14),A
4888            LD A,R
488A            LD H,A
488B            LD L,A
488C            LD (£FE16),HL
488F            LD A,R
4891            LD (£FE18),A
4894            PUSH BC
4895            CALL £8F6
4898            POP BC
4899            DEC BC
489A            LD A,B
489B            OR C
489C            JR NZ,REPT
489E            RET
489F            NOP
48A0            NOP
48A1            NOP
48A2            NOP
48A3            NOP
48A4            NOP
48A5 MESS1:     DB "L I G H T  R I D E R S",0
48BC MESS2:     DB "Please choose the Race Speed.",0
48DA MESS3:     DB "1. Easy.",0
48E3 MESS4:     DB "2. More Difficult.",0
48F6 MESS5:     DB "3. Getting interesting.",0
490E MESS6:     DB "4. Masochists only.",0
4922 MESS7:     DB "5. Forget it!!!",0
4932 MESS8:     DB "Use RETURN to pause game.",0
494C LSCORE:    DB 0
494D RSCORE:    DB 0
494E SPEED:     DW 0000
4950 CROWDDB:DB 129,129,129,129,129,129,129,129
4958            DB 129,129,129,129,129,129,129,129
4960            DB 129,129,129,129,129,129,129,129
4968            DB 129,129,129,129,129,129,129,129
4970            DB 0
4971 PLAYERS:DB "** PLAYER 1>0 **** 0<PLAYER 2 **",0
4992 BLOCK1:    DB 130,131,0
4995 BLOCK2:    DB 131,130,0
4998 LX:        DB 0
4999 RX:        DB 0
499A LY:        DB 0
499B RY:        DB 0
499C LDIR:      DB 0
499D RDIR:      DB 0
499E LDISP:     DW 0000
49A0 RDISP:     DW 0000
49A2 FANCOL:    DB 9,9,9,9,9,2,2,2
49AA            DB 9,9,9,9,9,8,8,8
49B2            DB 9,9,9,9,9,7,7,7
49BA            DB 9,9,9,9,9,13,13,13
49C2 LWINMESS:DB "PLAYER 1 WINS!",0
49D1 RWINMESS:DB "PLAYER 2 WINS!",0
49E0 STARS:     DB "***************",0
49EF            RET
49F0            RET
```

Symbols:

INIT	43D8	MESS1	48A5
MESS	48BC	MESS2	48BC
MESS3	48DA	MESS4	48E3
MESS5	48F6	MESS6	490E
MESS7	4922	MESS8	4932
SPRCL	4422	CLATT	4439
OUT	46A3	LSCORE	494C
SETSC	4442	RSCORE	494D
INP	4448	INP1	444E
INP3	445B	SPEED	494E
INP5	4468	INP2	4475
INP4	4480	GAME	4493
BORDER	4495	CROWDDB	4950
CROWDT	44AE	CROWDB	44B7
PLAYERS	4971	PRSCR	46A6
BLOCK1	4992	BLOCKS	44CC
BLOCK2	4995	B2	44D5
B3	44DE	B4	44E7
B5	44F0	B6	44F9
B7	4502	B8	450B
B9	4514	B10	451D
INITPOS	4526	LX	4998
LDIR	499C	RX	4999
RDIR	499D	LSPROUT	46D3
RSPROUT	46FD	CHEER	4727
SOUNDON	4799	GETRTJ	4555
STROBE	4792	LEFTR	4557
RIGHTR	4561	UPR	456D
DOWNR	4579	TESTR	4585
LR	4596	POINTR	45BD
UR	45A4	DR	45B2
RY	499B	POINTX	47DE
POINTY	47DF	POINT	47DA
CRASHR	47EB	PLOTX	47E8
PLOTY	47E9	PLOT	47E5
GETLTJ	45E3	LEFTL	45E9
RIGHTL	45F2	UPL	45FB
DOWNL	4604	TESTL	460D
LL	461E	POINTL	4645
UL	462C	DL	463A
LY	499A	CRASHL	4806
PAUSIN	466B	DELAY	467D
PAUSE	4672	DELOOP	4681
BACK	4699	COORD	4693
MLOOP	4695	CHAR	469E
LEFTWIN	4821	LDISP	499E
RIGHTWIN	4851	RDISP	49A0
UNFIN	4729	ROW	4749
PLAIN	4739	NORMAL	4769
SOUNDOFF	47BF	VOL	4755
NORM1	4775	FANCOL	49A2
NORM2	477A	QUIET	47C1
BANG	4880	LWINMESS	49C2
STARS	49E0	RWINMESS	49D1
REPT	4883		





Z80 Late Xtra!

Lurking within the heart of your Z80 chip are a set of undocumented commands. The reason they are not mention in Zilog's manuals is a mystery. I have tried the commands on the Einstein,MSX,MTX,and Amstrad .... they work on all machines which leads me to believe that they will work with all Z80 chips.

The OP CODE for references to the 16-bit IX REGISTER is 0FDHex and for the IY REGISTER 0DDHex. LD A,(IX) is assembled as:-DD7E00. ALL the new commands deal with the IX or IY registers, and to get the Z80 to execute the commands the first OP CODE must be FD or DD depending which register you are going to use.

All normal operations can be performed on the Index registers and they can be used as single registers for any sort of operation. For the sake of this article let's say that the X register has a LOW byte and a HIGH byte so that: LD IX,3B0AH

```
LD A,Low Byte IX
LD B,High Byte IX
```

This would result in register A containing 0A & register B holding 3B.

Obviously, the assembler will not recognise the new commands because the op codes are not built into it so we have to devise some way of duping it into accepting the new commands. The following is the easiest way to do just this, and still retain logical mnemonics.

Although we will be using the H & L registers these now stand for H(igh byte) L(ow byte) and the normal HL register pair is used in the normal manner - we are just borrowing their op codes !

Mnemonic	Op Code	Mnemonic	Op Code
LD H,A	67	LD H,B	60
LD H,C	61	LD H,D	62
LD H,E	63	LD L,A	6F
LD L,B	68	LD L,C	69
LD L,D	6A	LD L,E	6B
LD A,H	7C	LD B,H	44
LD C,H	4C	LD D,H	54
LD E,H	5C	LD A,L	7D
LD B,L	45	LD C,L	4D
LD D,L	55	LD E,L	5D
INC H	24	INC L	2C
LD H,L	65	LD L,H	6C

Suppose that we want to carry out the following instructions .....

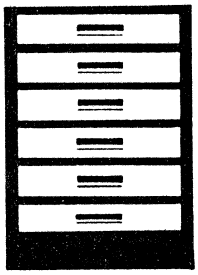
LD LOW BYTE OF IX INTO A AND THE LOW BYTE OF IY INTO B AND THEN LOAD THE A REGISTER INTO THE HIGH BYTE OF IY YOU WOULD DO THE FOLLOWING.

```
DB £DD          ;IX REGISTER PREFIX
LD A,L          ;LOW BYTE NOW IN A
DB £FD          ;IY REGISTER PREFIX
LD B,L          ;LOW BYTE IY NOW IN B
DB £FD          ;IY REFERENCE
LD H,A          ;HIGH BYTE OF IY NOW = TO A WHICH = LOW BYTE OF IX
```

THAT'S ALL THERE IS TO IT! YOU HAVE NOW ANOTHER 4 SINGLE REGISTERS THAT CAN BE USED FOR INSTRUCTIONS.

WARNING ££££ YOU CANNOT SINGLE STEP USING THE FRONT PANEL BECAUSE THE PANEL ROUTINES WILL NOT PICK UP ON THE Defined Byte.

A Simple Disc Database (Play around with it)



```
5 LET A=0
10 LET P=10: LET I=1
20 LET R=0: LET G=0: LET TS=0: LET NR=0
30 CLS : PAPER P: INK I: CSR 15,0: PRINT "MAIN MENU"
35 CSR 15,1: PRINT "_________"
40 CSR 8,3: PRINT "1 .... Open a file"
50 CSR 8,5: PRINT "2 .... Enter a record"
60 CSR 8,7: PRINT "3 .... View records"
70 CSR 8,9: PRINT "4 .... Search records"
80 CSR 8,11: PRINT "5 .... Save data to disc"
90 CSR 8,13: PRINT "6 .... Load data from disc"
100 CSR 8,15: PRINT "7 .... Quit program"
105 CSR 8,17: PRINT "8 .... Reorganize fields"
110 CSR 8,19: PRINT "9 .... Change colours"
115 CSR 9,22: PRINT "SELECT OPTION REQUIRED"
120 LET IN$=INKEY$: IF IN$<"1" OR IN$>"9" THEN  GOTO 120
130 IF IN$<>"1" AND IN$<>"6" AND IN$<>"9" AND R=0 AND IN$<>"7" THEN  GOTO 120
140 CLS : LET IN=VAL(IN$)
150 ON IN-1 GOSUB 1000,2000,6000,5000,7000,8000,9000,11000,10000
160 GOTO 30
1000 CSR 11,2: PRINT "CREATE A NEW FILE": IF A<1 THEN  GOTO 1030
1004 CSR 10,6: PRINT "Are you sure (y/n)?"
1005 CSR 16,16: PRINT "WARNING": CSR 4,18: PRINT "Creating a new file will erase"
1006 CSR 6,19: PRINT "the file already in memory"
1010 LET IN$=INKEY$: IF IN$<>"Y" AND IN$<>"y" AND IN$<>"N"  AND IN$<>"n  THEN  GOTO 1010
1020 IF IN$<>"Y" AND IN$<>"y" THEN  RETURN
1025 CLEAR : GOTO 5
1030 REM
1040 CLS : CSR 11,2: PRINT "CREATE A NEW FILE": CSR 11,3: PRINT "_________________": PAUSE 200
1050 CSR 10,20: PRINT "Number of fields (1-9)?";: LET IN$=INKEY$: IF IN$="" THEN  GOTO 1050
1055 LET A=VAL(IN$)
1060 IF A>9 OR A<1 THEN  GOTO 1050
1070 DIM A(A),N$(A,10)
1080 CSR 0,20: PRINT CHR$(5): FOR N=1 TO A
1090 PRINT CHR$(12): CSR 5,22: PRINT "( Up to 10 characters only )": CSR 0,0: PRINT "Name of fie
ld";N;" ";: INPUT X$: IF LEN (X$)>10 THEN  GOTO 1090
1100 LET N$(N)=X$
1120 LET TS=TS+A(N)
1130 NEXT : LET R=INT(2000/A): CLS : CSR 0,12: PRINT "MAX NUMBER OF RECORDS = ";R
1140 DIM A$(R,A,19): RETURN
2000 LET G=0
2010 IF NR=R THEN  GOTO 2180
2020 LET NR=NR+1
2030 CLS : CSR 8,0: PRINT NR-1;" out of";R;" records in use"
2040 FOR N=1 TO A: CSR 0,2*N+1: PRINT N$(N)
2050 CSR 11,22: PRINT "(Up to 19 characters)": CSR 0,20: PRINT N$(N);" ";CHR$(5);: INPUT X$: IF
LEN (X$)>19 THEN  GOTO 2050
2055 LET A$(NR,N)=X$
2060 LET Y=ASC(A$(NR,N)): IF Y=0 AND N=1 THEN  LET N=A: LET G=1: GOTO 2080
2070 LET A$(NR,N)=LEFT$(A$(NR,N),A(N)): CSR 12,2*N+1: PRINT CHR$(5);: PRINT A$(NR,N)
2080 NEXT
2090 IF G=1 THEN  GOTO 2160
2100 LET C=NR: PAUSE 500: IF NR=1 THEN  GOTO 2150
2110 CSR 16,1: PRINT "SORTING": IF A$(C,1)>=A$(C-1,1) THEN  GOTO 2150
2120 FOR N=1 TO A: LET X$=A$(C,N): LET A$(C,N)=A$(C-1,N): LET A$(C-1,N)=X$: NEXT : LET C=C-1
2130 IF C=1 THEN  GOTO 2150
2140 GOTO 2110
2150 GOTO 2010
2160 LET NR=NR-1
2170 RETURN
2180 CLS : CSR 10,12: PRINT "FILE FULL"
2190 PAUSE 5000: RETURN
3000 GOSUB 8500
3010 CSR 1,21: PRINT "Modify which field (1 TO";A;") ?";
3020 LET IN$=INKEY$: IF INKEY$="" THEN  GOTO 3020
3030 LET J=VAL(IN$)
3040 IF J<1 OR J>A THEN  GOTO 3020
3050 LET A$(D,J)="                    "
3060 CSR 13,2*J+1: PRINT CHR$(5): CSR 1,21: PRINT CHR$(5);: PRINT "Enter modified field now-";:
INPUT A$(D,J)
3070 LET A$(D,J)=LEFT$(A$(D,J),A(J))
3080 IF D=NR THEN  LET J=-1: GOTO 3130
```

```
3090 IF D=1 THEN  LET J=1: GOTO 3120
3100 IF A$(D,1)>A$(D+1,1) THEN  LET J=1
3110 IF A$(D,1)<A$(D-1,1) THEN  LET J=-1
3120 IF A$(D+1,1)="" AND J=1 THEN  GOTO 3180
3130 IF J=1 THEN  GOTO 3160
3135 IF D<=1 THEN  GOTO 3180
3140 IF A$(D,1)>=A$(D-1,1) THEN  GOTO 3180
3150 FOR N=1 TO A: LET X$=A$(D,N): LET A$(D,N)=A$(D-1,N): LET A$(D-1,N)=X$: NEXT : LET D=D-1: GO
TO 3080
3160 IF A$(D,1)<=A$(D+1,1) THEN  GOTO 3180
3170 FOR N=1 TO A: LET X$=A$(D,N): LET A$(D,N)=A$(D+1,N): LET A$(D+1,N)=X$: NEXT : LET D=D+1: GO
TO 3080
3180 PAUSE 500: RETURN
4000 LET G=D
4010 GOSUB 8500
4020 CSR 1,22: PRINT "ARE YOU SURE YOU WISH TO DELETE ?";
4030 LET IN$=INKEY$: IF IN$="" THEN  GOTO 4030
4040 IF IN$<>"Y" AND IN$<>"y" THEN  RETURN
4050 IF G=NR THEN  LET G=G-1
4060 CLS : CSR 16,12: PRINT "DELETING"
4070 IF D=NR THEN  GOTO 4090
4080 FOR N=1 TO A: LET A$(D,N)=A$(D+1,N): NEXT : LET D=D+1: GOTO 4070
4090 FOR N=1 TO A: LET A$(D,N)="                    ": NEXT : LET NR=NR-1: LET D=G
4100 PAUSE 1000: RETURN
5000 PRINT CHR$(12);"FIELD NO.",,"NAME OF FIELD": FOR N=1 TO A: CSR 1,2*N+1: PRINT N,,,N$(N): NE
XT
5010 CSR 1,21: PRINT "Search which field (1 TO";A;") ?";
5020 LET IN$=INKEY$: IF IN$="" THEN  GOTO 5020
5030 IF VAL(IN$)<1 OR VAL(IN$)>A THEN  GOTO 5020
5040 LET Z=VAL(IN$): CSR 1,21: PRINT CHR$(5);"Search field";Z;" for what ?";
5050 INPUT " ";Z$
5060 LET D=1: LET G=0: LET CH=1
5070 CSR 30,0: PRINT "SEARCHING": IF D>NR AND G=1 THEN  LET G=0: LET CH=-1 ELSE  IF D>NR THEN  G
OTO 5230
5080 IF D<1 AND G=1 THEN  LET G=0: LET CH=1 ELSE  IF D<1 THEN  GOTO 5230
5085 IF D<1 THEN  LET D=1
5090 IF LEFT$(A$(D,Z),LEN (Z$))<>Z$ THEN  LET D=D+CH: GOTO 5070
5100 LET LD=D: LET G=1: GOSUB 8500
5105 CSR 0,20: PRINT "----------------------------------------"
5110 CSR 0,21: PRINT "1-forwards   2-backwards    3-menu": PRINT "4-amend    5-delete   6-print"
5120 LET IN$=INKEY$: IF IN$="" THEN  GOTO 5120
5130 IF IN$<"1" OR IN$>"6" THEN  GOTO 5120
5135 LET IN=VAL(IN$)
5140 ON IN-1 GOTO 5160,5170,5180,5190,5200,5210
5150 GOTO 5120
5160 LET CH=1: LET D=D+1: GOTO 5070
5170 LET CH=-1: LET D=D-1: GOTO 5070
5180 RETURN
5190 GOSUB 3000: GOTO 5090
5200 GOSUB 4000: GOTO 5090
5210 GOSUB 12100: GOTO 5070
5220 GOTO 5110
5230 CLS : CSR 12,9: PRINT "NO RECORD WITH ";: CSR 20-LEN (Z$)/2,12: PRINT Z$
5240 CSR 14,15: PRINT "IN FIELD ";Z
5250 PAUSE 2000: RETURN
6000 LET D=1
6010 IF NR<1 THEN  GOTO 6170
6020 GOSUB 8500
6025 CSR 0,20: PRINT "----------------------------------------"
6030 CSR 0,21: PRINT "1-forwards   2-backwards    3-menu": PRINT "4-amend    5-delete   6-print"
6040 LET IN$=INKEY$: IF IN$="" THEN  GOTO 6040
6050 IF IN$<"1" OR IN$>"6" THEN  GOTO 6040
6055 LET IN=VAL(IN$)
6060 ON IN-1 GOTO 6080,6090,6100,6110,6120,6130
6070 GOTO 6030
6080 LET D=D+1: GOTO 6140
6090 LET D=D-1: GOTO 6140
6100 RETURN
6110 GOSUB 3000: GOTO 6020
6120 GOSUB 4000: GOTO 6010
6130 GOSUB 12100: GOTO 6030
6140 IF D>NR THEN  LET D=1
6150 IF D<1 THEN  LET D=NR
6160 GOTO 6010
6170 CSR 14,12: PRINT "FILE EMPTY"
6180 PAUSE 2000: RETURN
```

```
7000 CLS : CSR 5,8: PRINT "";: INPUT "Name of file to be saved? ";FI$
7010 CSR 5,11: PRINT "Insert disc and press <RET>"
7020 PAUSE 200
7030 IF INKEY$<>CHR$(13) THEN  GOTO 7030
7035 CLS : CSR 5,12: PRINT "SAVING ";FI$
7040 USER OPEN£1,FI$,"O"
7050 USER  PRINT £1,R,A,NR
7055 FOR N=1 TO A
7060 USER  PRINT £1,N$(N),A(N)
7065 NEXT
7070 LET C=1
7080 FOR N=1 TO A
7090 USER  PRINT £1,A$(C,N)
7095 NEXT
7100 LET C=C+1: IF C<=NR THEN  GOTO 7080
7110 USER CLOSE£1
7120 RETURN
8000 CLS : CSR 10,8: PRINT "Are you sure? (Y/N)"
8010 LET IN$=INKEY$: IF IN$="" OR IN$="6" THEN  GOTO 8010
8020 IF IN$="N" OR IN$="n" THEN  RETURN
8030 CSR 4,11: PRINT "";: INPUT "Name of file to be loaded? ";FI$
8035 PAUSE 200
8040 CSR 5,14: PRINT "Insert disc and press <RET>"
8050 IF INKEY$<>CHR$(13) THEN  GOTO 8050
8055 CLS : CSR 8,12: PRINT "LOADING ";FI$
8060 USER OPEN£1,FI$,"I"
8070 USER  INPUT £1,R,A,NR
8080 DIM A(A),N$(A,10),A$(R,A,19)
8085 FOR N=1 TO A
8090 USER  INPUT £1,N$(N),A(N)
8095 NEXT
8100 LET C=1
8105 FOR N=1 TO A
8110 USER EOF£1,8140
8120 USER  INPUT £1,A$(C,N)
8125 NEXT
8130 LET C=C+1: GOTO 8105
8140 USER CLOSE£1
8150 RETURN
8500 CLS : CSR 11,0: PRINT "RECORD NUMBER";D
8510 FOR N=1 TO A: CSR 0,2*N+1: PRINT N$(N)
8520 CSR 11,2*N+1: PRINT ">  ";A$(D,N)
8530 NEXT
8540 RETURN
9000 CLS : CSR 11,12: PRINT "ARE YOU SURE(Y/N)?"
9010 LET IN$=INKEY$: IF IN$<>"Y" AND IN$<>"y" AND IN$<>"N" AND IN$<>"n" THEN  GOTO 9010
9020 IF IN$="N" OR IN$="n" THEN  RETURN
9030 CLS : STOP
10000 CLS : CSR 13,0: PRINT "COLOUR CHANGE"
10005 CSR 13,1: PRINT "_____________"
10010 CSR 5,5: PRINT CHR$(5);: INPUT "Paper Colour ";P: IF P<0 OR P>15 THEN  GOTO 10010
10020 CSR 7,7: PRINT CHR$(5);: INPUT "Ink Colour ";I: IF I<0 OR I>15 THEN  GOTO 10020
10030 PAUSE 1000
10040 RETURN
11000 IF A=1 THEN  CSR 6,12: PRINT " CANNOT REORGANIZE 1 FIELD!": PAUSE 3000: RETURN
11010 PRINT "FIELD NO.",,"NAME"
11015 PRINT "________",,"____"
11020 FOR N=1 TO A: CSR 0,2*N+1: PRINT N,,,N$(N): NEXT
11030 PRINT : PRINT : PRINT "  Enter number of new": PRINT : PRINT "control field (2 TO";A;")";:
 INPUT NC
11040 LET NC=INT(NC): IF NC<2 OR NC>A THEN  CLS : GOTO 11010
11050 CLS : PRINT "CHANGING AND SORTING FIELDS"
11060 LET T$=N$(1): LET N$(1)=N$(NC): LET N$(NC)=T$: LET T=A(1): LET A(1)=A(NC): LET A(NC)=T
11070 FOR N=1 TO NR: LET T$=A$(N,1): LET A$(N,1)=A$(N,NC): LET A$(N,NC)=T$: NEXT
11080 FOR N=1 TO NR-1: LET K=N
11090 FOR J=N+1 TO NR
11100 IF A$(J,1)<A$(K,1) THEN  LET K=J
11110 NEXT : IF N<>K THEN  FOR C=1 TO A: LET T$=A$(K,C): LET A$(K,C)=A$(N,C): LET A$(N,C)=T$: NE
XT
11120 NEXT : RETURN
12000 USER  SAVE "DATABASE.BAS"
12010 RUN
12100 RETURN ★
```

View Letters Point

TO THE EDITOR

Dear sir,

I have had a number of letters asking how to merge the USER BASIC utility to existing basic programs, as the merge program published in issue 8 does not work correctly. The merge program is the cause of the problems as the subroutine being merged is moved to #C400 and may overwrite the SDX BDOS code at #D700.

To merge SDX versions of USER BASIC use the following procedure instead of the merge utility.

1: Load USER BASIC sent by Syntaxsoft
2: Save USER BASIC code to disc as a hex data file

```
(MTX 512) USER WRITE"SDX_512.HEX",16391,3100
(MTX 500) USER WRITE"SDX_500.HEX",32775,3100
```

3: Load basic program you want USER BASIC merged into
4: Type in the following CODE line 0

```
ASSEM 0         <RET>
Assemble>       <RET>

DS 250          ; USER BASIC <RET>
DS 250          <RET>
DS 250            "
DS 250            "
DS 250            "
DS 250            "
DS 250            "
DS 250            "
DS 250            "
DS 250            "
DS 250            "
DS 250            "
DS 200            "
RET

Return to basic with <CLS> <RET>
Assemble> <CLS> <RET>
```

5: Read in USER BASIC hex file from disc with

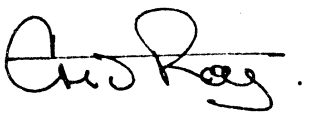
```
   USER READ"SDX_512.HEX",16391 <RET> (MTX 512)
or USER READ"SDX_500.HEX",32775 <RET> (MTX 500)
```

I have also included a listing which changes the SDX disc EOF# command so that it can jump to a line name. The instructions after ; are for MTX 500 machines.

The new EOF# command should be the only command in the basic line, and has the following syntax....

USER EOF#<channel>,<linename> which will jump to the <linename> specified if end of file found on <channel>.

Yours sincerly,



Eric Roy.

```
4D92          CP (HL)
4D93          RET NZ
4D94          DJNZ SYN1
4D96          RET
4D97 EOFNAM:  DB 03,"OF#"
4D9B          RET

Symbols:
SETUP   4D40     UBEOF   4D4C
EOFCOM  4D5F     EXIT    4D55
EOFNAM  4D97     SYNTAX  4D8E
RANGE   4D8C     FCBLP   4D7C
SYN1    4D8F

100 REM ******************************
110 REM **** USER BASIC EOF# DEMO ****
120 REM ****  SDX VERSIONS ONLY   ****
130 REM ******************************
140 REM Syntax for new EOF# command...
150 REM USER EOF#<channel>,<linename>
160 REM-------------------------------
170 REM  Create demo text file.
180 REM-------------------------------
190 USER BASIC
200 USER OPEN#1,"EOFDMO","O"
210 USER @GETTEXT
220 PRINT "Enter text or <RET> to close file "
230 INPUT T$
240 IF T$="" THEN  USER JUMP_GETEND
250 USER  PRINT #1,T$
260 USER JUMP_GETTEXT
270 USER @GETEND
280 USER CLOSE#1
290 REM-------------------------------
300 REM Read and display demo. file
310 REM-------------------------------
320 USER OPEN#2,"EOFDMO","I"
330 USER @INTEXT
340 USER EOF#2,INEND
350 USER  INPUT #2,T$
360 PRINT T$
370 USER JUMP_INTEXT
380 USER @INEND
390 USER CLOSE#2
400 PRINT
410 PRINT "End of demo file."

0 CODE

4007          DS 250      ;--USER BASIC--
4101          DS 250      ;    V 1.2
41FB          DS 250      ;(C) E.Roy 1985
42F5          DS 250
43EF          DS 250      ;--------------
44E9          DS 250      ;DO NOT DELETE
45E3          DS 250      ;  THIS LINE.
46DD          DS 250      ;--------------
47D7          DS 250      ;   MTX  512
48D1          DS 250      ; SDX DISC-TAPE
49CB          DS 250
4AC5          DS 250      ;--SYNTAXSOFT--
4BBF          DS 100
4C23          RET

Symbols:

1 CODE

4D40 SETUP:   LD A,02     ; IF MTX 500 enter
4D42          LD (#4314),A          LD (#8314),A
4D45          LD HL,UBEOF
4D48          LD (#FA8D),HL
4D4B          RET
4D4C UBEOF:   LD (#4BBD),DE       ; LD (#8BBD),DE
4D50          LD A,(DE)
4D51          CP "E"
4D53          JR Z,EOFCOM
4D55 EXIT:    LD A,(#FAD2)
4D58          LD DE,(#4BBD)       ; LD DE,(#8BBD)
4D5C          JP #F5B6
4D5F EOFCOM:  LD HL,EOFNAM
4D62          CALL SYNTAX
4D65          JR NZ,EXIT
4D67          INC DE
4D68          RST 30
4D69          LD A,B
4D6A          AND A
4D6B          JR NZ,RANGE
4D6D          OR C
4D6E          JR Z,RANGE
4D70          CP 05
4D72          JR NC,RANGE
4D74          PUSH DE
4D75          LD DE,#0028
4D78          LD IX,#D818
4D7C FCBLP:   ADD IX,DE
4D7E          DEC A
4D7F          JR NZ,FCBLP
4D81          POP DE
4D82          BIT 7,(IX+#27)
4D86          JP Z,#27AA
4D89          JP #4189    ; JP #8189
4D8C RANGE:   RST 28
4D8D          DB #22
4D8E SYNTAX:  LD B,(HL)
4D8F SYN1:    INC HL
4D90          INC DE
4D91          LD A,(DE)
```

We have received this letter from Len Clark.

A BASIC Problem

I have a problem and I wonder whether some intrepid reader of MEMOPAD is able to solve it for me. To state the problem I must first summarise some facts about simple string handling in MTX BASIC.

Consider the very simple program:

```
10 LET A$="MTX"
20 LET K=LEN (A$)
30 PRINT K
```

This will display the result, 3, on the screen. I did not need to declare the variable A$ in any way and, so long as it never exceeds 64 characters in length, I will not need to declare it. Even if I want it to exceed 64 characters, all I have to do is add the line:

```
5 DIM A$(150)
```

and I will have up to 150 characters.

Consider the following:

```
10 DIM A$(120)
20 LET A$="MEMOPAD"
30 LET K=LEN (A$)
40 PRINT A$;" IS ";K;" characters long."
```

This will display the message:

```
MEMOPAD is 7 characters long.
```

At this point caution is necessary. If I subsequently reuse the variable I may have problems. If, for example, I add on:

```
60 LET A$="MTX"
70 PRINT A$
```

to my surprise I get

```
MTXOPAD
```

However, this is easily solved. I slip in the line

```
50 LET A$=""
```

(for which I cunningly left space) and the variable is cleared before reuse.

All this is fairly obvious. Now we approach the problem. If I want to feed in a large number of strings (say, read in the football clubs in a league), I don't want to have to use a different variable for each string. Naturally enough I would want to write something like:

```
*100 FOR I=1 TO 32
*110 INPUT "Team";TEAM$(I)
*120 NEXT I
```

(I have to put stars beside the lines to show that, in fact, it won't work.) It won't work because TEAM$(1) leads the computer to expect a single character in first position in the variable. If I precede my wrong program by

```
DIM TEAM$(32)
```

I will get a single string variable of 32 characters in length.

Still no great problem. I can use a double array:

```
10 DIM (32,64)
```

The only limitation is that I must now specify the length of each string as 64. Thus I have, in effect, 32 variables each of 64 characters length.

Now comes the problem. Suppose I take just one of those variables and write the following:

```
10 DIM A$(32,64)
20 LET A$(1)="MEMOPAD"
30 LET K=LEN (A$(1))
40 PRINT A$(1);" is ";K;" characters long."
```

This is basically the same as an earlier program except that I am using arrayed string variables. The resulting message is:

```
MEMOPAD is 64 characters long.
```

Worse is to come. If I add the lines

```
50 LET A$(1)=""
60 LET A$(1)="MTX"
70 PRINT A$
```

I still get the result:

```
MTXOPAD
```

There seems to be no simple way of clearing the old string. The only way I have found of doing it is by a subroutine such as the following:

```
200 LET DUM=0
210 LET DUM=DUM+1
220 LET K=ASC(A$(1,DUM))
230 IF K=0 THEN RETURN
240 LET A$(1,DUM)=CHR$(0)
250 GOTO 210
```

This does the trick but is extremely slow. If the program is handling any number of changes in value, the program can be extremely slow. The obvious case in point is a dictionary SORT which has to swap strings about within an array to allow insertion or deletion of an item.

So how can I do the sorting in a quicker way?

We would like to know if any other member has experienced the same problem as our machines give the current response.★

00106	26X26 SPREAD SHEET	UTIL	SYNT	7.95	I	ANY
00057	3D TACHYON FIGHTER	ARC	CONT	6.95	I	ANY
00135	9 ELECTRIC. PROGS	EDUC	SSFT	13.95	I	ANY
00062	ADVENTURE QUEST	ADV	LVL9	8.75	I	ANY
00033	AGROVATOR	ARC	SYNT	5.95	I	512
00125	AIRCRAFT MAGNETISM	FLGT	AVTN	?	I	ANY
00120	AIRCRAFT PAYLOADS	FLGT	AVTN	?	I	ANY
00122	AIRLAW 2	FLGT	AVTN	?	I	ANY
00123	AIRSPEED INDICATOR	FLGT	AVTN	?	I	ANY
00071	ALICE IN WONDER.	ADV	CONT	6.02	I	ANY
00121	ALTIMETER	FLGT	AVTN	?	I	ANY
00008	ASTROMILON	ARC	CONT	6.02	I	ANY
00047	ASTROPAC	ARC	CONT	6.02	I	ANY
00058	BACKGAMMON	BRD	CONT	7.95	I	ANY
00041	BASIC BUSINESS	BS	CONT	5.95	I	ANY
00043	BLORBO	ARC	CONT	6.02	I	ANY
00073	BOUNCING BILL	ARC	SYNT	4.95	I	ANY
00074	BRIDGE	CARD	CONT	6.95	I	512
00130	BUSINESS GAME	BRD	SSFT	15.95	I	ANY
00077	CANVAS	UTIL	CONT	7.95	I	ANY
00085	CAVES OF ORB	ADV	SYNT	5.95	I	512
00137	CESIL INTERPRETER	LANG	SSFT	5.95	I	ANY
00094	CHAMBEROIDS	ARC	MEGA	5.95	I	ANY
00059	CHESS	BRD	CONT	8.75	I	ANY
00053	COBRA	ARC	CONT	6.02	I	ANY
00025	COLOSSAL ADVENTURE	ADV	LVL9	8.75	I	ANY
00098	COMBAT	ARC	PANS	2.95	I	512
00028	COMPOSER	UTIL	XAV	13.00	I	ANY
00046	CONT RAIDERS	ARC	CONT	6.02	I	ANY
00099	CRIBBAGE	CARD	SCRP	2.95	E	ANY
00110	CRYSTAL	ARC	MEGA	5.95	I	ANY
00050	DEN.GOES BANANAS	ARC	SCRP	2.95	I	ANY
00011	DENNIS & CHICKEN	ARC	SCRP	2.95	I	ANY
00103	DENNIS AND CIRCUS	ARC	SCRP	2.95	I	ANY
00068	DOODLEBUG	ARC	SYNT	4.95	I	ANY
00108	DOWNSTREAM DANGER	ARC	MEGA	5.95	I	ANY
00096	DR. FRANKIE	ARC	SYNT	5.95	I	512
00056	DRAUGHTS	BRD	CONT	6.95	I	ANY
00111	DRIVE THE CEE 5	ARC	MEGA	5.95	I	ANY
00063	DUNGEON ADVENTURE	ADV	LVL9	8.75	I	ANY
00067	EDASM	UTIL	SYNT	7.95	I	512
00066	EMERALD ISLE	ADV	LVL9	5.95	I	ANY
00038	ESCAPE FROM ZARKOS	ARC	MEGA	5.95	I	ANY
00081	EXTENDED BASIC	6.95	SENT	6.95	I	ANY
00082	FATHOMS DEEP	ARC	MEGA	5.95	I	ANY
00090	FIG FORTH	LANG	SYNT	15.75	I	512
00055	FIREHOUSE FREDDIE	ARC	CONT	6.02	I	ANY
00021	FIRST LETTERS 1	EDUC	CONT	8.75	I	ANY
00092	FKEY DEFINER	UTIL	MEMB	6.95	I	ANY
00137	FLUMMOX	ARC	SYNT	5.95	I	512
00132	FRACT, PERCENTAGES	EDUC	SSFT	5.95	I	ANY
00052	GAUNTLET	ARC	CONT	6.02	U	ANY
00102	GHOSTLY CASTLE	ADV	PANS	2.95	I	ANY
00031	GOLDMINE	ARC	CONT	6.02	I	ANY
00069	GRAPHICS	UTIL	CONT	5.95	I	ANY
00087	H & L DUMP	UTIL	MEM	4.95	I	ANY
00072	HAWKWARS	ARC	SYNT	4.95	I	ANY
00065	HELI-MATHS	EDUC	CONT	5.95	I	ANY
00034	HUNCHY	ARC	SYNT	4.95	I	ANY
00083	ICEBERG	ARC	SYNT	4.95	I	ANY
00105	JET SET WILLY	ARC	SPRJ	6.95	I	ANY
00015	JOHNNY REB	WAR	LOTH	6.02	I	ANY
00097	JUMPING JACK FLASH	ARC	SYNT	5.95	I	512
00115	KARATE KING	ARC	MEGA	5.95	I	ANY
00016	KEY TO TIME	ADV	LUMP	6.02	I	ANY
00042	KILOPEDE	ARC	CONT	6.02	I	ANY
00019	KNUCKLES	ARC	CONT	7.95	I	ANY
00078	LES FLICS	ARC	PSS	6.95	E	ANY
00032	LITTLE DEVILS	ARC	SYNT	4.95	I	ANY
00024	LORDS OF TIME	ADV	LVL9	8.75	I	ANY
00035	M COMMAND & ARCAD.	ARC	SYNT	4.95	I	ANY
00070	MAN FROM GRANNY	ADV	SYNT	4.95	I	512
00104	MANIC MINER	ARC	SPRJ	6.95	I	ANY
00119	MAPS AND CHARTS	FLGT	AVTN	?	I	ANY
00126	MAPS AND CHARTS 1	FLGT	AVTN	?	I	ANY
00022	MATHS 1	EDUC	CONT	8.75	I	ANY
00013	MAXIMA	ARC	CONT	6.02	E	ANY
00006	MEMOCHEQUE	UTIL	SYNT	6.95	I	ANY
00075	MEMOSKETCH	UTIL	SYNT	7.95	I	ANY
00089	MINER DICK	ARC	XAV	6.95	I	ANY
00044	MISSION ALPHATRON	ARC	CONT	6.02	I	ANY
00030	MISSION OMEGA	ARC	SYNT	4.95	I	ANY
00054	MURDER AT MANOR	ADV	LUMP	6.02	I	ANY
00010	MUSIC PAD	UTIL	CONT	6.02	I	ANY
00003	NEMO	ARC	CONT	6.00	I	ANY
00131	NETWORK LOADER	UTIL	SSFT	8.95	I	ANY
00112	OBLITERATION ZONE	ARC	MEGA	5.95	I	ANY
00045	OBLOIDS	ARC	CONT	6.02	I	ANY
00129	PAINTBOX	UTIL	SYNT	5.95	I	ANY
00001	PAYROLL	UTIL	CONT	21.25	I	512
00005	PHAID	ARC	CONT	6.02	I	ANY
00061	PHYSICS 1	EDUC	CONT	8.75	I	ANY
00124	PILOT NAVIGATION	FLGT	AVTN	?	I	ANY
00012	PONT & BLACKJACK	CARD	CONT	6.02	I	ANY
00009	POT HOLE PETE	ARC	CONT	6.02	I	ANY
00040	PURCHASE LEDGER	BN	CONT	12.75	I	512
00048	QOGO	ARC	CONT	6.02	I	ANY
00076	QOGO 2	ARC	MEGA	5.95	I	ANY
00095	QUANTUM	ARC	SYNT	5.95	I	ANY
00109	QUAZZIA	ARC	MEGA	5.95	I	ANY
00107	RED MOON	ADV	LVL9	?	U	ANY
00127	RELATIVE VELOCITY	FLGT	AVTN	?	I	ANY
00064	RETURN TO EDEN	ADV	LVL9	8.75	I	ANY
00020	REVERSI	BRD	CONT	7.95	I	ANY
00114	ROLLA BEARING	ARC	MEGA	5.95	I	512
00100	RUTHLESS BASTARD	ARC	LSFT	2.50	I	512
00002	SALES LEDGER	UTIL	SYNT	15.75	I	512
00029	SALTY SAM	ARC	SYNT	4.95	I	ANY
00113	SEPULCRI SCELERATI	ARC	MEGA	5.95	I	512
00101	SLOOPY'S CHRISTMAS	ARC	PANS	2.95	I	ANY
00116	SMG	ARC	MEGA	5.95	I	ANY
00049	SNAPPO	ARC	CONT	6.02	I	ANY
00023	SNOWBALL	ADV	LVL9	8.75	I	ANY
00036	SON OF PETE	ARC	MEGA	5.95	I	ANY
00136	SOUND & RESISTORS	EDUC	SSFT	5.95	I	ANY
00026	SPELLI-COPTER	EDV	CONT	5.95	I	ANY
00080	SPOOLER	UTIL	MEM	4.95	I	ANY
00017	STAR COMMAND	ARC	CONT	6.95	I	ANY
00014	SUPA CODER	UTIL	SYNT	7.95	I	ANY
00084	SUPER BIKE	ARC	SYNT	4.95	I	ANY
00004	SUPER MINEFIELD	ARC	CONT	6.02	I	ANY
00093	SURFACE SCANNER	ARC	MEGA	5.95	I	ANY
00133	SYMMETRY & GLASS	EDUC	SSFT	5.95	I	ANY
00039	TAPE TO DISC	UTIL	MEM	6.95	I	ANY
00007	TAPEWORM	ARC	CONT	6.02	I	ANY
00088	TARGET ZONE	ARC	SYNT	6.95	I	ANY
00118	THE DESIGNER	UTIL	HALT	8.95	I	ANY
00128	THE WALL	ARC	SYNT	4.95	I	512
00051	THE ZOO GAME	ADV	CONT	6.02	I	512
00134	TITRATION,CHROMATO	EDUC	SSFT	5.95	I	ANY
00006	TOADD	ARC	CONT	6.02	I	ANY
00018	TURBO	ARC	CONT	6.95	I	ANY
00117	USER BASIC	UTIL	SYNT	8.95	I	ANY
00079	USER EXTEND	UTIL	MEM	7.95	I	ANY
00027	UTILITIES 1	UTIL	CONT	4.95	I	ANY
00091	VERNON & VAMPIRES	ARC	SYNT	5.95	I	ANY
00138	WOOD SIMULATION	EDUC	SSFT	5.95	I	ANY
00060	WORD & PICTURE	EDUC	CONT	8.75	I	ANY

PLEASE ONLY ORDER THOSE MARKED " I "

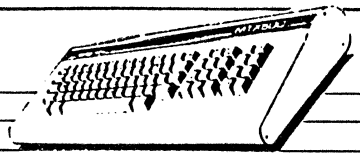
KEY:- STOCK NUMBER TITLE TYPE HOUSE MEMBER PRICE STOCK MACHINE

E=EXPECTED SOON I = IN STOCK U = UNAVAILABLE AT PRESENT

Syntaxsoft

UNIT 109, GLENFIELD PARK, GLENFIELD ROAD, NELSON BB9 8AR.
TELEPHONE [0282] 698849
TELEX:635480 CENTIM.G. ANSWERBACK SYNTAX

HARDWARE



PRICES EFFECTIVE FROM 1ST DECEMBER 1985

MTX 512 ..... #119.00 MEMBERSHIP EXTRA
MTX 500 ..... # 69.95 MEMBERSHIP EXTRA

32K MEMORY EXPANSION .....#36.74
64K MEMORY EXPANSION ..... #45.43
128K MEMORY EXPANSION ..... #69.52
**** PLEASE STATE FOR WHICH MODEL ******

NEWWORD 32K ON ROM .....#36.74
PASCAL 16K ON ROM .....#36.74
SPECULATOR .....#36.95

DMX80 PRINTER .....#179.95
PRINTER CABLE 8.95

SDX 500K DRIVE + INTERFACE .....#222.50
SDX 1MEG DRIVE + INTERFACE .....#265.83

2ND 250K SDX DRIVE .....# 89.00
2ND 500K SDX DRIVE .....#169.00
2ND 1MEG SDX DRIVE .....#203.00

FDX 2ND DRIVE 500K .....#141.00
FDX 2ND 1MEG DRIVE .....#163.00

DUST COVER .....#3.50
DMX80 PRINTER RIBBON .....#8.98

80 COL CARD + CPM +NW/SC .....#180.95
80 COL UPGRADE KIT (RS232) .....# 27.00

PACKAGE ONE
SDX 500K DRIVE + INTERFACE
+80 COL BOARD +CPM + NW + SC .....#355.00

PACKAGE TWO
AS ABOVE BUT WITH 1MEG DRIVE .....#395.00

PACKAGE THREE
SDX 500K DRIVE : INTERFACE:
COMPUTER MTX512: 80 COL PCB CPM .....#449.00

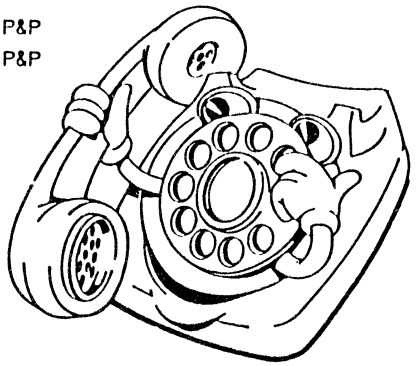
PACKAGE FOUR
AS ABOVE BUT WITH 1MEG DRIVE .....#489.00

FDX SINGLE 500K CPM SYSTEM .....#539.00
FDX SINGLE 1MEG CPM SYSTEM .....#675.00

FDX TWIN 500K CPM SYSTEM .....#569.00
FDX TWIN 1MEG CPM SYSTEM .....#740.00
(REquires RS232 comms board)

SILICON DISCS
500K ..... #145.90
1MEG ..... #441.00

FLOPPY DISCS (BOX 10) GUARANTEED ..... #18.95
PACE NIGHTINGALE MODEM .....#119.00 + #5.00 P&P
250K DISC DRIVE + INTERFACE .....#199.00 + #5.00 P&P



DISC CASES HOLD 10 DISC ..... # 2.55
FLOPPICLENE DISC CLEANING KIT .....#17.20
ANTISTATIC SCREEN WIPES ...(10)....# 1.50
DISC CABINETS (LOCKABLE) 110 DISCS #36.95

CRIB CARD ..... # 2.16
ROM LISTINGS ..... #45.00
SDX CONTROLLER LISTINGS .#20.00
ROM CALLS INFO SHEET ... 50p
RST 10 CALLS INFO SHEET 50p
INTERRUPTS INFO SHEET 80p
DDT INFO BOOK .....#2.00
VDP TECHNICAL MANUAL ....#5.95
MTX SERVICE MANUAL .....#6.95

THE ABOVE PRICES ONLY APPLY TO U.K SALES & BFPO SALES
BFPO SHOULD ADD AN EXTRA £30.00 TO COVER ADMINISTRATION BY MEMOTECH

UPGRADE PACKAGE 1 ..... £198.00
UPGRADE PACKAGE 2 ..... £223.39
The above require factory fitting so add an extra £25 to cover cost of this service.

DON'T FORGET IF YOU HAVE A DISC DRIVE YOU SHOULD OWN A HIGH QUALITY HEAD CLEANER see FLOPPICLENE
.... over half the problems handled by us are due to dirty disc heads.

Abridged Terms & Conditions (Downloads)

Disclaimer

www.primrosebank.net, (***the website***) is provided by Dave Stevenson as a service to the public, is provided "as is" and carries no warranties, expressed or implied, of any kind.

Dave Stevenson is not responsible for, and expressly disclaims all liability for, damages of any kind arising out of use, reference to, or reliance on any information contained within the website or made available for download. Whilst the information contained within the website site is periodically updated, no guarantee is given that the information provided on the website is correct, complete, and up-to-date.

A number of articles on the website contain technical data and practical guidance which may be of use in testing and maintaining various items of vintage computer and electronics hardware. Such articles are not intended to cover all aspects of the tasks involved and may omit essential information, including necessary safety precautions. Performance of the tasks described may risk damage to equipment and/or people. The reader is responsible for ensuring that he/she is capable of performing the tasks described and well as assessing the inherent risks involved and taking appropriate measures to mitigate such risks.

Dave Stevenson expressly disclaims all liability for, damages to equipment or injury of any kind arising out of use of such technical data and guidance.

As far as possible, information included on the website from other sources has been credited to the respective author and/or publisher. The majority of content on the website is derived from material first published in the 1980s. ***This material is likely still under copyright of the original author and/or publishers.*** The authors and/or publishers may not have given express permission to copy, transmit or make this information available for download, but I believe that they would have no objection to this archive information being placed into the public domain.

However, should the author and/or publisher of the original material find any content on the website for which they wish to assert their rights, they should notify Dave Stevenson (by e-mail to: webmaster@primrosebank.net) who would be pleased to enter into a dialogue to agree a satisfactory resolution of their concerns.

If you obtained this file as part of a paid-for package, you have been scammed! I suggest that you request a refund from the seller, please also advise Dave stevenson at the e-mail address above.