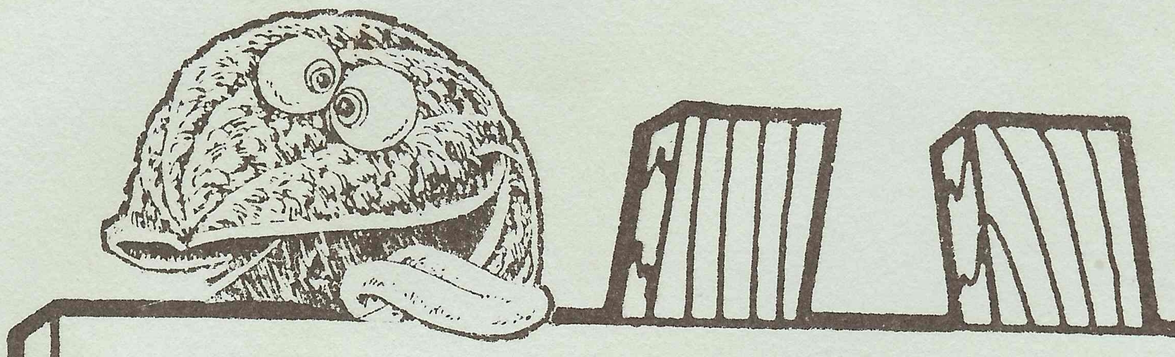
MEMOPAD

VOL 010

ISSUE 9





In This Issue

EDITORIAL ..... PAGE 249

SOFTWARE REVIEW ..... PAGE 251

DISC MANIA - PART 2 ..... PAGE 253

PROMZAP CORRECTIONS ..... PAGE 260

MASTERMIND ..... PAGE 261

SHELL SHOT ..... PAGE 264

MTX TOKENS ..... PAGE 265

VIEWPOINT ..... PAGE 267

PASCAL ..... PAGE 269

ATTENTION GETTER ..... PAGE 273

SOFTWARE ..... PAGE 274

HARDWARE ..... PAGE 275

YOU CAN TALK TO US ON
PRESTEL 919998675

£1.35

memopad

Memotech Computer User Club Magazine

Edited by Tim Marstian

Artwork Anthony "Joe90"



Exective 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

COVER PRICE £1.25. MEMOPAD IS COPYRIGHT SYNTAXsoft 1985

AVAILABLE BY SUBSCRIPTION ONLY.

Editorial

Having begged Tim to allow me to write this month's editorial, I find it hard to know where to start. I have just returned from a trip to the Far East which took in countries such as Hong Kong, Taiwan, and Japan. Since my return I am finding it hard to settle down and get back to some real work !

It is a pity that the country with the lowest inflation and the highest valued currency is finding it hard to compete in world markets. Japan is slowly finding it more difficult to export its products. The value of the Yen is slowly having a drastic effect on the country. How its population survives is an ever increasing wonder - water melons at #20.00 each, whiskey at #54.00 a bottle, and add bread to your meal and you're into next month's salary !

MSX-2 is dead, already, in Japan. I found that the latest models were on sale at 75% discount in all major stores. It is not that the system is useless- Japan have a games machine "The Nintendo" which has saturated the games market and fulfilled the need for the enthusiast who requires nothing more than a joystick and a few shoot-em up games. However, the machines that are selling are CPM versions of MSX2 with the ability to do something wonderful with graphics and a video camera. All MTX owners should keep this in mind when the next creation from Memotech Computers is publicised.

It is apparent from the news media that most people are concerned with the so called 'Russian Threat'. After visiting Taiwan I am more frightened of them than the Russians. The escalation in quality and technical know-how is frightening. Everywhere there are small manufacturers springing up from where once stood hovels - some are still abysmal buildings but the quality of the end product is fast gaining world recognition. I predict that from this country will come the next boom in computers. From what I witnessed, it will not be long before the IBM Compatible will be on sale for the price of an Amstrad. In fact, if the truth is known, many of the low priced machines now available, started their origins in Taiwan. However, for me, it was the warmth, and friendliness of the people that captured my heart - I really didn't want to come home !

Anyway, back to business. We shall shortly be in a position to bring you full details of the new machines from Geoff. Production has gained ground on the back-log of orders and by the time this magazine falls on the mat most of you will have received your hardware. I would, once again, like to thank you for being patient.

In this edition, you will see a re-print from another magazine on HIGHWAY ENCOUNTER. I hope this goes to show you how good a program this game really is. I must say, in all honesty, that I am a litt disappointed in the sales of Tournament Snooker - we had to put money where our mouth was, and at present we are selling nothing like the quantities I would have thought this quality piece of software would have be capable of generating. Never the less, we shall continue to look for those games to satisfy the games player. CP/m users can now obtain Microsoft's MULTIPLAN on the FDX & SDX systems. We hope to be in a position to add a further 200 titles to the CP/m range within the near future. We are also hoping to have a supply of Public Domain software within the next few weeks. The first two titles from Mastertronics will be available by the time your next magazine is due. So don't despair, there is plenty for you to go at, and you can now order your add-on with confidence and a future exciting,development will be available as an upgrade - but more about this next time!

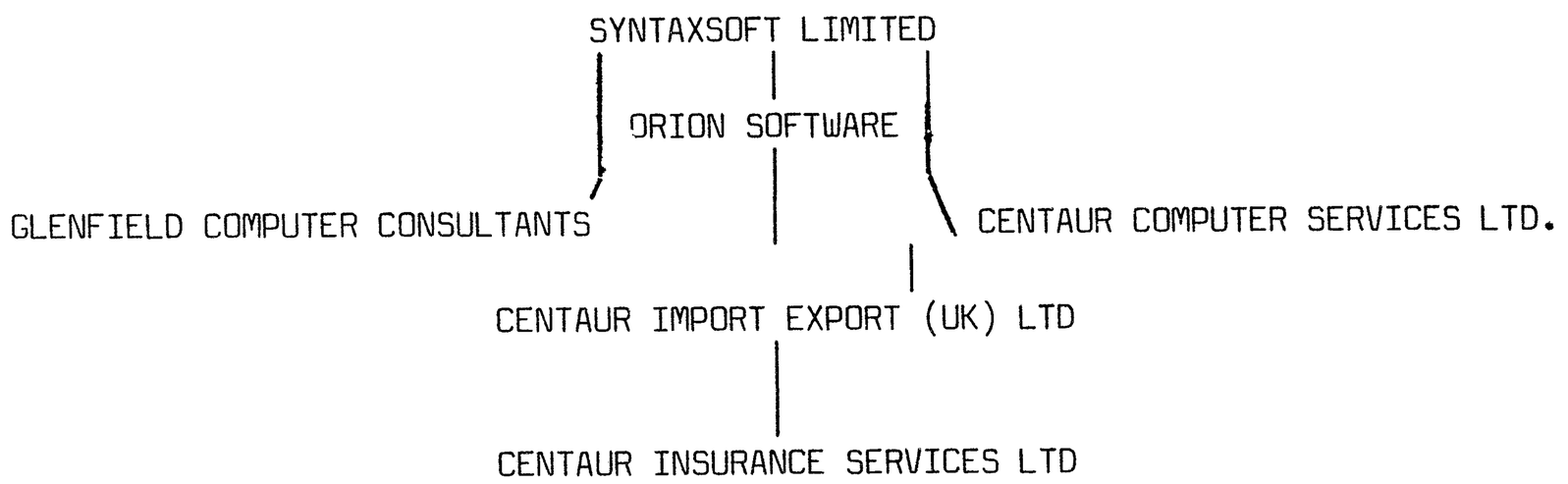
Because so much has happened within Syntaxsoft Limited during the course of these past months we thought we would let you know about it.

Syntaxsoft now owns an Import/Export company with offices in Taiwan, Hong Kong & Japan. Obviously, with this kind of operation we can acquire items at a much cheaper rate than buying them in this country. However, the Company can also, and does, sell on to wholesalers and retailers in this country. We have some very big accounts but it would not be prudent to mention them. From time to time we will pass on some of the bargains to you and these will be the subject of a separate mailing sheet.

Another company Syntax has set up is Centaur Insurance Limited. This company is an insurance brokerage with excellent rates on car insurance, office insurance, personal investment policies etc. Subscribers to the magazine are welcome to write in for a quotation and are entitled to special rates. We are, at present, investigating a certain policy which may be of interest to all members and more details of this will be covered in a future edition.

We have now appointed a new director and Company Secretary, who is well known in this area for his expertise in business management, and for the help he has given to hundreds of small businesses. He is ROY PEARSON, and from the time you receive this magazine he will have taken over all Keith's responsibilities in the running of the companies. No, Keith has not left. He will now be left with more time to carry out the projects he knows and loves. ★

THE COMPANY STRUCTURE:



KH

MEMOTECH COMPUTERS LIMITED

New Products

PLEASE NOTE THE FOLLOWING WILL BE AVAILABLE IN 8 WEEKS

SYSTEM ONE

1 MEG 3½ DRIVE
+ INTERFACE £129.00

SILICON DISCS

1 MEG £189.00
2 MEG £298.00
4 MEG £556.00

SILICON DISCS SHOULD BE FACTORY FITTED AND CAN BE COLLECTED, FITTED, AND RETURNED FOR £25.00

SYSTEM TWO

1 MEG 3½ DRIVE
INTERFACE
512K SILICON DISC
80 COLUMN + CPM + NEWWORD £209.00

80 COLUMN UPGRADE FOR SYSTEM ONE
CPM + SILICON DISC £109.00

ALL SUBJECT TO £5.00 POSTAGE AND PACKING

SOFTWARE REVIEW

Highway Encounter

It's been a long time since we have received anything from this well known and widely acclaimed author; his 3D graphics are both distinctive and cleverly applied.

Highway Encounter has a simple plot. You control a droid or Vorton, which has the simple task of taking an explosive device from one end of a straight road to the enemy base at the other. If the device is successfully delivered then the enemy advance will be halted and you will have won. You view the action from an oblique aerial angle, seeing one complete section of road at a time. In all you have a total of five droids under your command, but you can only directly control one at any given time. The remaining droids are automatically programmed to push the device down the centre of the road until they meet an obstruction whereupon they come to a dead stop.

The control droid is steered in a similar way to the robot in *Alien 8*. To move in a given direction you must first rotate until the droid is facing the appropriate way and then move forward. This is a difficult system to come to terms with: the droid reacts as if subject to inertia, and it takes some time to accelerate to full speed and seemingly even longer to come to a halt.

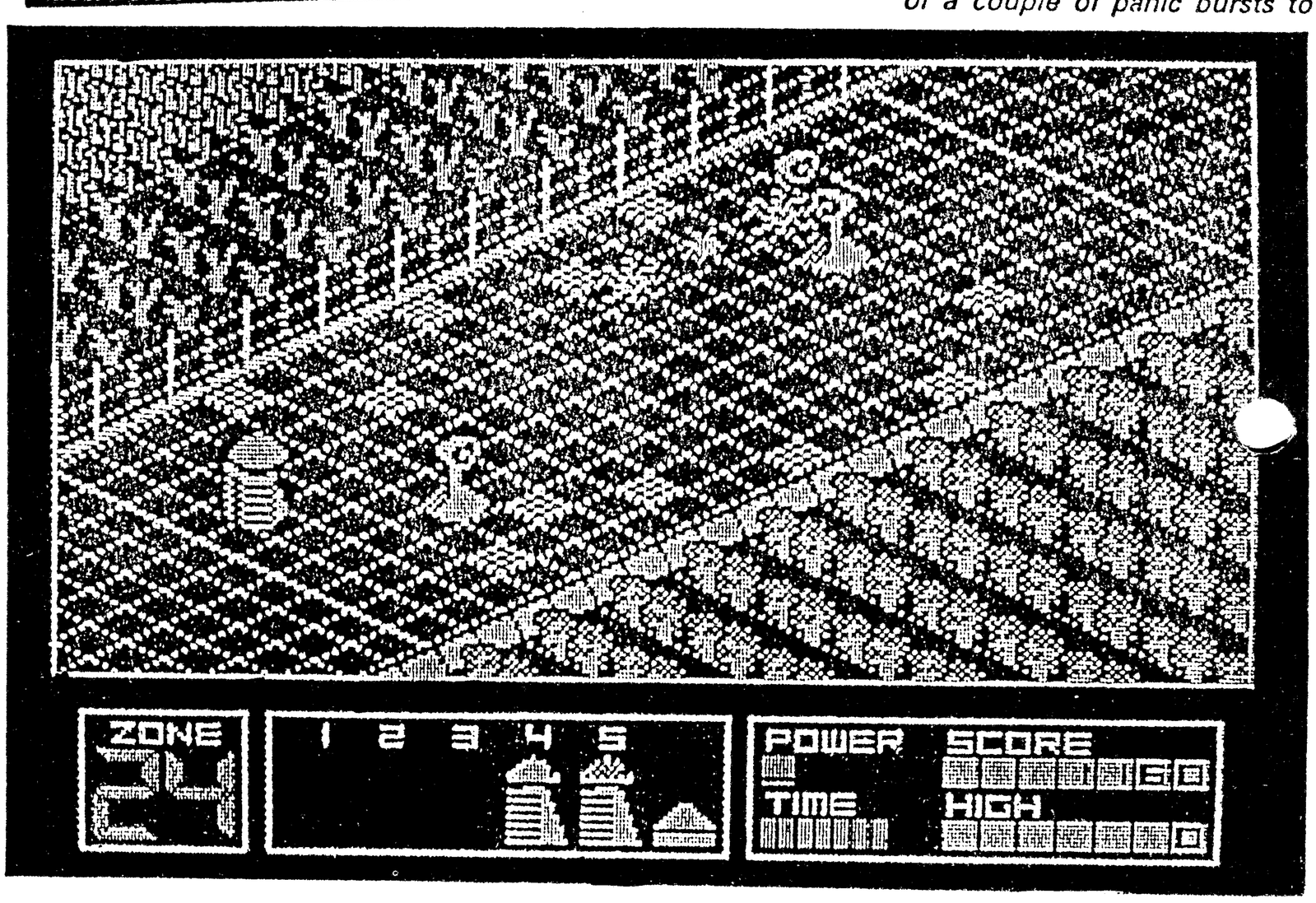
At first sight the game looks very simple — so long as the road ahead is clear the explosive device and automatic droids will make their way down the middle of the road. They will only stop if their path is blocked, as it frequently will be. Most of the blockages are caused by large barrels and boxes left lying around but very soon you realise that although these objects are a pain they can in fact be put to very good use. You see, the enemy aren't just sitting back waiting for you and your bomb to turn up on their doorstep . . . they have some pretty smart defence systems waiting for you. The first type take the form of stationary mines, if you collide with one then you lose a droid and will have to take over one of the automatic ones. The mobile mines are a bit more of a problem, they slide back and forth across the width of the road and while you can dodge them with the controlled droid, the only way of ensuring a safe passage for the rest of your party is to block the mines in against the wall with boxes or barrels.

Generally the controlled droid can nudge objects along the road, but it is often quicker to manoeuver them into place by firing laser bolts at them. Sometimes more than one object can be moved at a time, but the whole operation becomes a lot slower. Beware of the objects that look like boxes but in fact melt when fired at. The laser fires in bolts of three blasts, each blast sending out a sort of hairy bullet thingy which hits home with devastating effect. However, after firing a burst of three bolts the laser pauses briefly to re-charge and this can be a bit of a problem when it comes to dealing with the third type of defence system — the enemy's droids.

The enemy machines come in all shapes and sizes, and some just seem to wander about while others actively hunt for you. They vary in the way they move, while some droids are prepared to try a frontal assault and are easy to take care of, others sneak up on you while their allies draw your fire. Clearing a path along the highway becomes progressively more difficult. To start with, you find it fairly easy to move items around to block the mines off, later on, not only do the mines appear in larger groups, but they move more erratically and your attempts to trap them will be hampered by ever increasing numbers of nasties attacking you.

To make matters worse the game has a time limit for completion. The amount of time remaining is shown by a bar graph on the lower part of the screen. Within the same area you are told how many droid Vortons you have left and which of the 30 zones you are in. Points are awarded according to how many droids have reached which zone. The more droids you have surviving and the nearer the end of the road they are, the more points you will score.

How dare they say he's a cross between a Dalek and and dustbin? The little droidy fellow's real cute — dig those hairy laser bolts.



CRITICISM

● 'I knew it, for all these years the Daleks have been given bad press. *Highway Encounter* gives them a chance to save a World (actually they are Vortons, but they look something like a cross between a Dalek and a dustbin, and yes, I have heard the joke). The game boasts excellent 3D graphics — Ultimate-type both in design and quality. This is a very playable and highly addictive game. I loved the introduction as well.'

● *'I am very impressed by this highly original and excellently designed game. The idea of having to chunter a bunch of droids down a road may sound a little boring, but wait until you try. Clearing the way through calls for a bit of thought but most of all you had better have a steady nerve when you meet the enemy droids. The lasers are devastating but their repeat rate is low, and it's all to easy to fire of a couple of panic bursts to find yourself with an 'empty' gun when it comes to killing the nasty bearing down on you. Time and time again I found myself looking for strategic spots from which to fire, some sort of cover so I could only be attacked from one direction and thus forcing the enemy into a 'killing zone'. This is a superb game, the gameplay is fast and thrilling, the graphics quick smooth and detailed; absorbing? Blotting paper doesn't even come near.'*

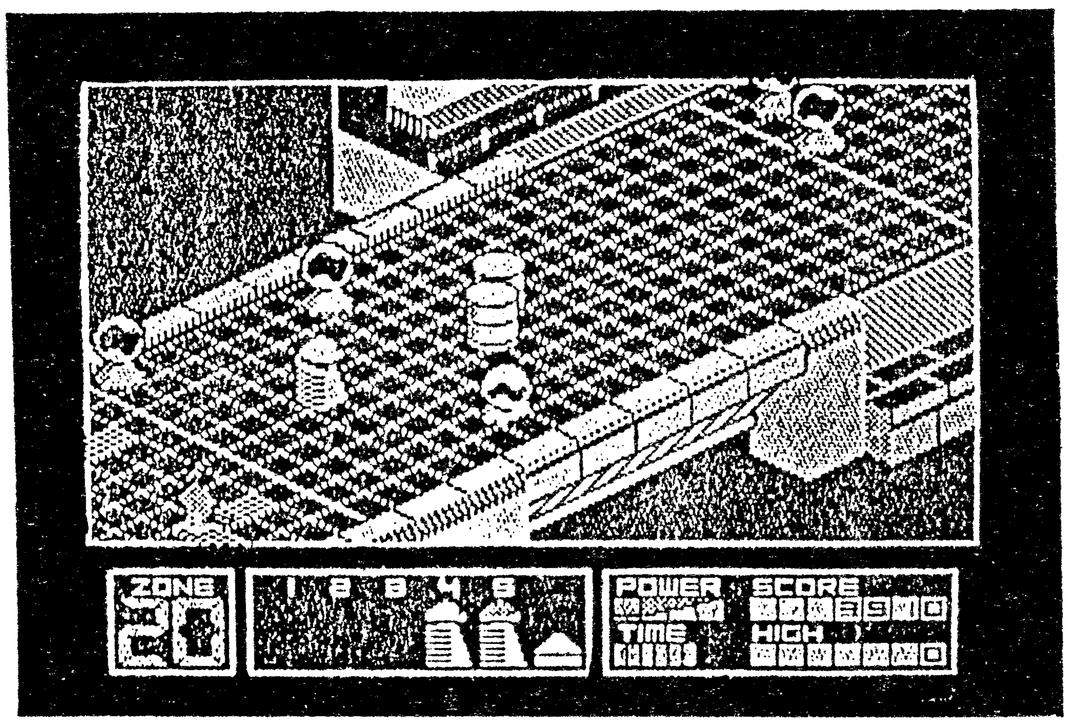
● 'This has to be one of the best games of the year. It has 3D graphics which would make Ultimate's hair stand on end. The clarity of the graphics was surprising — with most other 3D graphics I tend to get a bit 'lost' in them, and my eyes start playing tricks on me after a while. None of these things happened to me while I was playing this one. All the characters are very well drawn and nicely animated. There are many different types of nasty, some of which move in a set pattern, and on those screens you have to devise a routine which will get you and your 'Lazertron' across safely. When you're up against aliens which move around erratically it is best to blast hell out of them. I really enjoyed playing *Highway Encounter* but unfortunately I think it will be easy to complete in a short amount of time and I will probably get bored with it then.'

COMMENTS

Control keys: 1/Q accelerate/decelerate, o/p left/right, space to fire
Joystick: Interface Two, Kempston and Protek
Keyboard play: very fast, probably better than using a joystick
Use of colour: road and objects in black and white but nice use of colour for the roadside graphics
Graphics: brilliant
Sound: some very nice effects
Skill levels: one
Lives: five
Screens: 30 and a surprise!
General rating: magic

Use of computer	85%		
Graphics	96%	Addictive qualities	96%
Playability	95%	Value for money	84%
Getting started	93%	Overall	95%

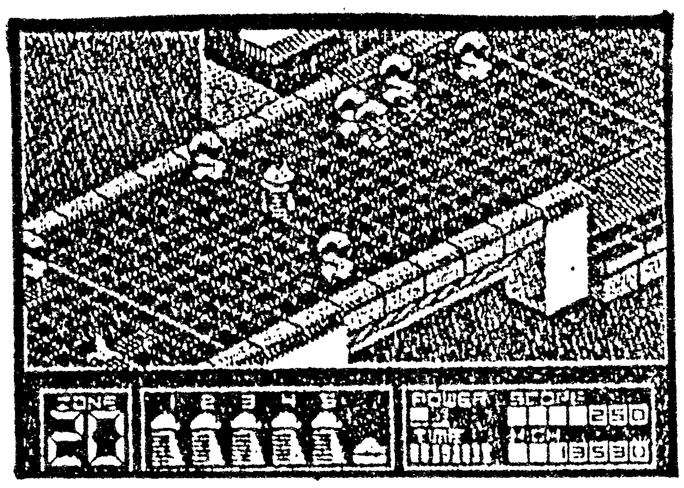
Tote that box, heave that barrel, pen up the nasties and let the good guys get down the road. Droid fun in HIGHWAY ENCOUNTER, beautifully projected in 3D.



Highway Encounter

THE ORDERLY advance of the aliens to destroy civilisation as Slugger would like it to be, is given a new twist in **Highway Encounter** from Vortex. The aliens stick to the main road, and your job is to ferry the lasertron to their base where it will obliterate the nasties.

The twist is the combination of state-of-the-art **Knight Lore** graphics with straight shoot-'em-up button bashing. The lasertron is propelled by a chain of five Vortons, dalek-like robots with guns in their heads. You control one Vorton at a time, but the other four will continue to move the lasertron forward if they can, and are thus vulnerable to attack. It seems particularly devious to design a game where you can lose your other lives before you even get to play them.



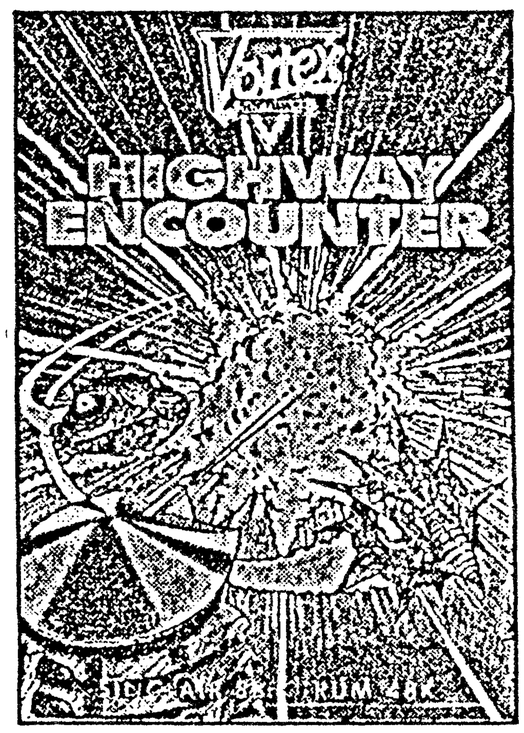
There are 32 screens of 3D highway to negotiate, each one containing a problem of its own. Some involve moving oil-drums around to prepare the way for the lasertron, others are free-for-all scraps with the aliens. The most difficult involve both.

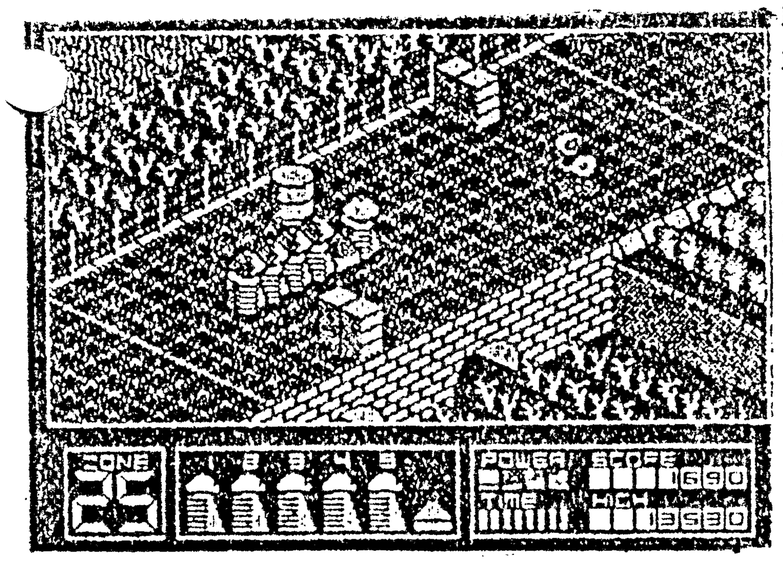
The aliens have an innocent fury about them. Some look like rejects from a remake of *War of the Worlds*; others resemble psychotic eggs. Vortex promises staggering effects when the lasertron is finally brought to its destructive goal.

In many respects the game is superior to **Knight Lore** and **Alien 8**, in that everything moves faster and in a true eight directions. There is colour in the screens, though not much, and most importantly, the action continues off-screen so you can't hang about for long trying to work out the best tactics. Weaknessess include less variety in the problems, and the lack of a maze — the highway is a simple straight road, easy to wander along to the end, but much more difficult to negotiate with the lasertron in tow.

It is a long time since we last saw a Vortex game — **TLL** and **Cyclone** were the last products. **Highway Encounter** is the best yet, innovative and addictive, and should go down a treat this summer.

Chris Bourne





Vortex have been making addictive 3D games for a long time while now – worth pointing out because a lot of people are going to say that *Highway Encounter* looks an awful lot like an Ultimate game. So it does, and there's no way the author of *Highway Encounter* hasn't seen *Knight Lore*. Nevertheless it's a technically superb, thoroughly addictive game with plenty of original touches.

To visualise *Highway Encounter* imagine a lateral *Knight Lore* – your view of the map stretching over several screens from bottom left to top right, rather like *Zaxxon* but with the screen scrolling on only when you reach the top right corner rather than continuously.

Highway Encounter takes some of the Ultimate ideas of 'solid' 'usable' objects which can be shunted around to make a sophisticated strategic zap game rather than the judgement/timing challenge of *Knight Lore*.

The idea is this, a team of five little robots called Vortons must push a litle 3D diamond shaped object over more than 30 screens. One robot is under your control, the others follow each other along the centre line. Each screen has different horrors

and different possibilities for action. Nasties include eyeballs, vicious venus flytraps, indestructible sparks and little space ships. Some follow set patterns and can be blasted relatively easily – some of them come for you.

There are also a number of objects scattered around which may be used as a barricade or to block in the indestructible baddies. Pushing or blasting them into the right position is a matter of some thought and planning – a nice balance to the 'blast the baddies' element.

After some disastrous first tries I realised that blasted baddies stay blasted and trapped aliens stay trapped – for this reason the best technique is to block up your diamond shaped object and spare robots and then go ahead alone killing as much as possible and using objects to block in the baddies you can't kill. Then, theoretically, you can unblock your diamond and other robots and make one dash for it to the end.

My single worry about the game is that it may prove too easy. After grasping the above idea, I've got fairly close to clearing the way for a run through to the final zone – that's after around six plays. Still, it may still be a hell of a lot more difficult than I, smugly, think. ★

Disc Mania by *Eric Roy* Part 2

This program seems to be the character generator program of books written about CP/M, i.e. they all have it in one form or another, so why not here. It's purpose is to count and display the number of words in a text file, (useful if you write articles and get paid by the word, Ed!!!!).

Okay it's not much, but it does show you how BDOS uses the functions (see last issue) to find the file in the directory, then read the data from the disc.

The USER command WORDS, "filename" will envoke the new function and display the number of words in a text file. The filename can contain the wildcard characters? and * if you wish to find the words in a number of files, i.e. USER WORDS, "*.TXT" would count the words in all the files on disc with the extension letters TXT.

After checking the syntax of the new command and getting the filename into the file control block a loop is entered which uses functions #11, search for first, and #12, search for next, to scan the directory looking for names that match the one in the file control block. These functions load four directory entries at a time into a 128 byte area of memory known as the DMA. The address of the DMA having been set with function #1A, set DMA. (Use the panel to display #E680 after USER DIR). If a match was found then the filename is copied from the DMA into a buffer used by the program to store the filenames.

The screen then clears and the title is printed. A loop is then entered which gets a filename from the buffer and puts it in the file control block. A call to the COUNT subroutine then counts the words in the file.

Count first displays the filename it is working on, then uses function #0F to open the file. The word count is set to 1 and a loop is entered to read one record at a time into the DMA (128 bytes) using function #14 read sequential. The DMA is then scanned looking for spaces, carriage returns, line feeds and end of file characters.

When an end of text file character is found (#1A) the loop exits and the file is closed using function #10, close file. This process is continued until all the filenames in the buffer have been read.

The source code written on the Edasm macro assembler (SDX version now available from Syntaxsoft) is well documented and the various subroutines used should not need explaining. If you don't have Edasm then the best way of entering the code is to use the panels Display>F618 command and then enter the hex data bytes directly into memory. When finished enter WRITE "WORDS.HEX",63000,458 to save object code to disc.

If you used Edasm to enter the source, use the LOAD directive to load object code to #B000 then WRITE "WORDS.HEX",45056,458. ★

Enter READ "WORDS.HEX",63000 : let a = USR(63000) to reload code and initialise. The USER Q command can be used to set the SDX back to normal operation.

```
D8E8                ;**************************************
                    ;       ---WORD COUNT TEXT FILES---
                    ;
                    ;       SDX 250k single disc system
                    ;
                    ;         by Eric Roy  March 1986
                    ;**************************************
                    ;
                            ORG     0F618H          ; = 63000
                            LOAD    0B000H          ; = 45056
                    ;
                    ; Equates
                    ;----------
                    ;
FAD2                PAGE    EQU     0FAD2H
0CAB                PRINTX  EQU     00CABH
0250                BASIC   EQU     00250H
0DD0                BCASC   EQU     00DD0H
02E8                BCDEC   EQU     002E8H
0009                TAB     EQU     09
000D                CR      EQU     00DH
000A                LF      EQU     00AH
001A                EOF     EQU     01AH
0000                WFLAG   EQU     00
0001                SPFLAG  EQU     01
                    ;
000F                OPENF   EQU     00FH
0010                CLOSEF  EQU     010H
0011                SEARCHF EQU     011H
0012                SEARCHN EQU     012H
0014                READF   EQU     014H
001A                SETDMA  EQU     01AH
2F2F                UFILEN  EQU     02F2FH
F5B0                BDOS    EQU     0F5B0H
E680                DMA     EQU     0E680H
D8E8                FCB5    EQU     0D8E8H
                    ;
                    ; Vector SDX commands to NEWCOM routine
                    ;---------------------------------------
                    ; USER commands jump to #F5B3, the first 3 byte
                    ; instruction LD A,(PAGE) is changed to JP NEWCOM
                    ;
                    ; LET x = USR(63000) from basic to init.
                    ;
                    VECSDX
F618 2124F6                 LD      HL,NEWCOM
F61B 22B4F5                 LD      (0F5B4H),HL
F61E 3EC3                   LD      A,0C3H
F620 32B3F5                 LD      (0F5B3H),A
F623 C9                     RET
                    ;
                    ; Entry point for SDX commands
                    ;------------------------------
                    ;
                    NEWCOM
F624 ED53DCF7               LD      (SNAME),DE      ; Save pos. in basic
F628 1A                     LD      A,(DE)          ; Get first letter
F629 FE57                   CP      "W"
```



```
F62B 281A               JR      Z,WORDCOM
F62D FE51               CP      "Q"
F62F 280A               JR      Z,QUITCOM       ; Q (quit)
                ;
                ; Exit to SDX if not W or Q
                ;---------------------------
                ; Register pair DE -> start of USER command
                ;                   A  loaded with (PAGE)
                ;
                NAMERR
F631 ED5BDCF7           LD      DE,(SNAME)      ; Position in basic
F635 3AD2FA             LD      A,(PAGE)        ; A = current page/rom
F638 C3B6F5             JP      0F5B6H          ; Jump to new SDX entry
                ;
                ; Here if Q command
                ;-------------------
                ; Reset #F5B3   JP NEWCOM to LD A,(PAGE)
                ;
                QUITCOM
F63B 21D2FA             LD      HL,PAGE
F63E 22B4F5             LD      (0F5B4H),HL
F641 3E3A               LD      A,03AH
F643 32B3F5             LD      (0F5B3H),A
F646 C9                 RET
                ;
                ; Here if WORDS,"filename" command
                ;---------------------------------
                ; Test name is legal, then check for filename.
                ; Filename can contain wildcards ? and *
                ;
                WORDCOM
F647 21D6F7             LD      HL,WORDN        ; HL -> Valid name data
F64A CDCBF7             CALL    UNAME           ; Test syntax
F64D 20E2               JR      NZ,NAMERR       ; Failed
                ;
F64F 13                 INC     DE
F650 1A                 LD      A,(DE)
F651 FEFF               CP      0FFH            ; Test for filename
F653 CAD4F7             JP      Z,UNDEF         ; after WORDS,
                WORD
F656 21E2F7             LD      HL,DIRBUF
F659 22DEF7             LD      (FNBUF),HL
F65C AF                 XOR     A
F65D 32E0F7             LD      (FNCNT),A       ; Init. buffer variables
                ;
F660 DD21E8D8           LD      IX,FCB5         ; File control block
F664 CDB6F7             CALL    GETUF           ; Get filename from basic
F667 1B                 DEC     DE
F668 D5                 PUSH    DE              ; Position in basic
F669 1180E6             LD      DE,DMA
F66C 0E1A               LD      C,SETDMA
F66E CDB0F5             CALL    BDOS                    ; DMA at #E680
                ;
F671 11E8D8             LD      DE,FCB5
F674 0E11               LD      C,SEARCHF       ; Search directory for filenames
F676 CDB0F5             CALL    BDOS            ; On return A = #FF if no names
F679 FEFF               CP      0FFH            ; found, else A = 0,1,2 or 3
F67B CA7BF7             JP      Z,NOFILE        ; = position in DMA.
                ;
                ; Get filename(s) that match and store in buffer
                ;
                WORD2
F67E CD8AF7             CALL    FADR            ; HL -> filename
```

```
F681 ED5BDEF7           LD      DE,(FNBUF)      ; Store filename in buffer
F685 010B00             LD      BC,0000BH
F688 EDB0               LDIR                    ; Move it into buffer
F68A ED53DEF7           LD      (FNBUF),DE      ; Save next free space
F68E 3AE0F7             LD      A,(FNCNT)
F691 3C                 INC     A
F692 32E0F7             LD      (FNCNT),A       ; Inc. name count
                        ;
F695 0E12               LD      C,SEARCHN       ; Search directory for
F697 CDB0F5             CALL    BDOS            ; next filename.
F69A FEFF               CP      0FFH            ; On return A = #FF if end
F69C 20E0               JR      NZ,WORD2        ; of directory reached.
                        ;
                        ; To get here there must be at least one
                        ; filename in the buffer that matches.
                        ;
F69E D7                 RST     010H            ; Print heading
F69F 6D                 DB      06DH
F6A0 B7                 DB      0B7H
F6A1 0946696C
F6A5 656E616D
F6A9 65202020
F6AD 20202020
F6B1 576F7264
F6B5 730D0A             DB      TAB,"Filename       Words",CR,LF
F6B8 97                 DB      097H
F6B9 092D2D2D
F6BD 2D2D2D2D
F6C1 2D2D2D2D
F6C5 2D2D2D2D
F6C9 2D2D2D2D
F6CD 2D0D0A             DB      TAB,"------------------------",CR,LF
                        ;
F6D0 21E2F7             LD      HL,DIRBUF       ; Reset to start of
F6D3 22DEF7             LD      (FNBUF),HL      ; filename buffer
                    WORD3
F6D6 21E8D8             LD      HL,FCB5
F6D9 E5                 PUSH    HL
F6DA CD6CF7             CALL    CLRFCB          ; Clear FCB of rubbish
F6DD D1                 POP     DE
F6DE 13                 INC     DE              ; DE -> FCB+1
F6DF 2ADEF7             LD      HL,(FNBUF)      ; HL -> name in buffer
F6E2 010B00             LD      BC,000BH        ; BC =  length
F6E5 EDB0               LDIR                    ; Move name into FCB5
F6E7 22DEF7             LD      (FNBUF),HL      ; Start of next name
                        ;
F6EA CDFCF6             CALL    COUNT           ; Count the words in file
F6ED D7                 RST     010H
F6EE 82                 DB      082H
F6EF 0D0A               DB      CR,LF
                        ;
F6F1 3AE0F7             LD      A,(FNCNT)       ; Decrease name count
F6F4 3D                 DEC     A
F6F5 32E0F7             LD      (FNCNT),A
F6F8 20DC               JR      NZ,WORD3        ; Not finished
                        ;
F6FA D1                 POP     DE
F6FB C9                 RET                     ; Back to basic
                        ;
                        ; Count words in file
                        ;------------------------
                        ;
                        COUNT
```

```
F6FC D7                 RST     010H
F6FD 09                 DB      TAB
                ;
F6FE 21E9D8             LD      HL,FCB5+1       ; HL -> filename
F701 CD97F7             CALL    PRINT           ; Print it
F704 D7                 RST     010H
F705 82                 DB      082H
F706 3D20               DB      "= "
                ;
F708 AF                 XOR     A
F709 32E1F7             LD      (STATUS),A      ; Signal in word
F70C 0E0F               LD      C,OPENF         ; Open file
F70E 11E8D8             LD      DE,FCB5
F711 CD805F5            CALL    BDOS
F714 FEFF               CP      0FFH            ; If #FF then unable
F716 CA7BF7             JP      Z,NOFILE        ; to open file
                ;
F719 010100             LD      BC,01           ; Word count to 1
                COUNT1
F71C C5                 PUSH    BC
F71D 0E14               LD      C,READF         ; Read a sector into DMA
F71F 11E8D8             LD      DE,FCB5
F722 CD805F5            CALL    BDOS
F725 C1                 POP     BC
F726 B7                 OR      A               ; If 0 sector read ok
F727 2032               JR      NZ,COUNT5       ; Non 0 end of file
                ;
F729 2180E6             LD      HL,DMA          ; HL -> DMA ( #E680 )
                COUNT2
F72C 7E                 LD      A,(HL)          ; Get a character
F72D E67F               AND     07FH
F72F FE1A               CP      EOF             ; End of file byte
F731 2828               JR      Z,COUNT5        ; Yes
F733 FE20               CP      " "             ; Space
F735 2817               JR      Z,COUNT3        ; Yes
F737 FE0D               CP      CR              ; Carrage return
F739 2813               JR      Z,COUNT3        ; Yes
F73B FE0A               CP      LF              ; Line feed
F73D 280F               JR      Z,COUNT3        ; Yes
                ;
                ; Here if non space character (word)
                ;
F73F 3AE1F7             LD      A,(STATUS)      ; Check if still scanning
F742 FE00               CP      WFLAG           ; the same word.
F744 280D               JR      Z,COUNT4        ; Yes don't inc count
                ;
F746 03                 INC     BC              ; Increase count
F747 3E00               LD      A,WFLAG
F749 32E1F7             LD      (STATUS),A      ; Signal inside word
F74C 1805               JR      COUNT4
                COUNT3
F74E 3E01               LD      A,SPFLAG
F750 32E1F7             LD      (STATUS),A      ; Signal space
                COUNT4
F753 23                 INC     HL              ; Next in DMA
F754 7C                 LD      A,H             ; Check for end of DMA
F755 FEE7               CP      0E7H            ; #E680 + #80 = #E700
F757 20D3               JR      NZ,COUNT2       ; Not end of DMA
F759 18C1               JR      COUNT1          ; Read next sector
                COUNT5
F75B C5                 PUSH    BC
F75C 0E10               LD      C,CLOSEF        ; Close the file
F75E 11E8D8             LD      DE,FCB5
```

```
F761 CDB0F5             CALL    BDOS
F764 C1                 POP     BC              ; Word count
F765 CDD00D             CALL    BCASC           ; Convert to ascii
F768 CDE802             CALL    BCDEC           ; Print decimal number
F76B C9                 RET
                ;
                ; Clear FCB -> by HL to 0
                ;-------------------------
                ; Entry         HL -> FCB to be cleared
                ;
                ; Exit          A  =  00
                ;               HL -> End of FCB+1
                ;
                CLRFCB
F76C C5                 PUSH    BC
F76D D5                 PUSH    DE
F76E AF                 XOR     A
F76F 77                 LD      (HL),A
F770 E5                 PUSH    HL
F771 D1                 POP     DE
F772 13                 INC     DE
F773 012900             LD      BC,029H         ; Length of FCB
F776 EDB0               LDIR
F778 D1                 POP     DE
F779 C1                 POP     BC
F77A C9                 RET
                ;
                ; Here if no files in directory
                ;-------------------------------
                ;
                NOFILE
F77B D7                 RST     010H
F77C 8A                 DB      08AH
F77D 0D0A               DB      CR,LF
F77F 4E6F2046
F783 696C6573           DB      "No Files"
F787 C35002             JP      BASIC           ; Basic Ready prompt
                ;
                ; Find filename in DMA
                ;----------------------
                ; Up to 4 directory entries are in the DMA
                ; each 32 bytes long. Filename starts at
                ; directory +1.
                ;
                ; Entry         A = 0,1,2 or 3
                ;
                ; Exit          HL -> start of filename
                ;
                FADR
F78A 2181E6             LD      HL,DMA+1        ; HL -> filename
F78D 112000             LD      DE,020H         ; DE = length of directry
F790 A7                 AND     A
F791 C8                 RET     Z               ; If 0 then HL correct
F792 47                 LD      B,A
                FADR1
F793 19                 ADD     HL,DE           ; Calc. adr. for HL
F794 10FD               DJNZ    FADR1
F796 C9                 RET
                ;
                ; Print filename in FCB.
                ;-----------------------
                ; Entry         HL -> filename
                ;
                ; Exit          A  =  Space (32)
```

```
                        ;               B  =  00
                        ;               HL -> end filename+1
                        ;
                        PRINT
F797 0608                       LD      B,08
F799 CDACF7                     CALL    DISPLAY         ; Print filename
F79C 3E2E                       LD      A,"."
F79E CDAB0C                     CALL    PRINTX
F7A1 0603                       LD      B,03
F7A3 CDACF7                     CALL    DISPLAY         ; Print file type
F7A6 3E20                       LD      A," "           ; Space
F7A8 CDAB0C                     CALL    PRINTX
F7AB C9                         RET
                        ;
                        ; Display filename.type to screen
                        ;-----------------------------------
                        ; Bit 7 stripped off so that file type letters
                        ; are displayed correctly.
                        ;
                        DISPLAY
F7AC 7E                         LD      A,(HL)          ; Get letter
F7AD E67F                       AND     07FH            ; Strip off bit 7
F7AF CDAB0C                     CALL    PRINTX          ; Print it
F7B2 23                         INC     HL
F7B3 10F7                       DJNZ    DISPLAY
F7B5 C9                         RET
                        ;
                        ; Get filename and place in FCB.
                        ;-----------------------------------
                        ; Entry          DE -> filename in basic
                        ;                IX -> FCB for filename
                        ;
                        ; Exit           DE -> end of name+2
                        ;
                        GETUF
F7B6 3AD2FA                     LD      A,(PAGE)
F7B9 F5                         PUSH    AF              ; Save calling page / rom
F7BA F630                       OR      030H
F7BC 32D2FA                     LD      (PAGE),A
F7BF D300                       OUT     (00),A          ; Switch in disc rom
F7C1 CD2F2F                     CALL    UFILEN          ; Get user filename
F7C4 F1                         POP     AF
F7C5 32D2FA                     LD      (PAGE),A
F7C8 D300                       OUT     (00),A          ; Ret. to calling page / rom
F7CA C9                         RET
                        ;
                        ; Test for valid command name
                        ;-----------------------------------
                        ; Entry          DE -> Name in basic
                        ;                HL -> Valid name data
                        ;
                        ; Exit           DE -> Byte after name
                        ;
                        UNAME
F7CB 46                         LD      B,(HL)          ; Length of name -1
                        UNAME1
F7CC 23                         INC     HL
F7CD 13                         INC     DE
F7CE 1A                         LD      A,(DE)
F7CF BE                         CP      (HL)            ; Test letters
F7D0 C0                         RET     NZ              ; Failed to match
F7D1 10F9                       DJNZ    UNAME1
F7D3 C9                         RET
```

```
                ;
                ; Valid names / buffers etc...
                ;-------------------------------
                ;
F7D4 EF         UNDEF   RST     028H
F7D5 26                 DB      38
                ;
F7D6 054F5244
F7DA 532C       WORDN   DB      05,"ORDS,"
F7DC 0000       SNAME   DB      00,00
F7DE 0000       FNBUF   DB      00,00
F7E0 00         FNCNT   DB      00
F7E1 00         STATUS  DB      00
                DIRBUF                  ; Requires #2C0 bytes
                        END
Workarea - 5D1A to 60A3
ORG  end - F7E2
LOAD end - B1CA
```

Promzap Corrections

John Hudson

Well folks it's that time again.

Once again I have to say 'sorry'; I am afraid that there are two mistakes in the logic drawing for the prom blower published in in issue 7 of the mag.

The mistakes are:-

(1) Pin 1 on both 74LS374's should be shown connected to 0volts (pin 10)

(2) Pin 20 on the ZIF socket should be shown as being connected to 0volts (pin 14)

I really am sorry for any problem these mistakes have caused to any one who is attempting to build the unit, and would like to thank Richard (? sorry I didn't make a note of your surname) who rang to ask why pin 1 on the 374's and pin 20 on the ZIF were not shown on the drawing.★

SDX 250k Disk Drive, as new and still under warranty!
'Tape-to-Disk' software to transfer all your favourite cassette-based software to disk!
15 floppy disks!

TOTAL COST NEW £239 - THIS FOR ONLY £179 - SAVE £60 !!

Contact **Chris Moore** on (0622) 679093

at 1 Newenden Close, Maidstone, Kent, ME14 5RU



Mastermind by *Fredrik Lundkvist*



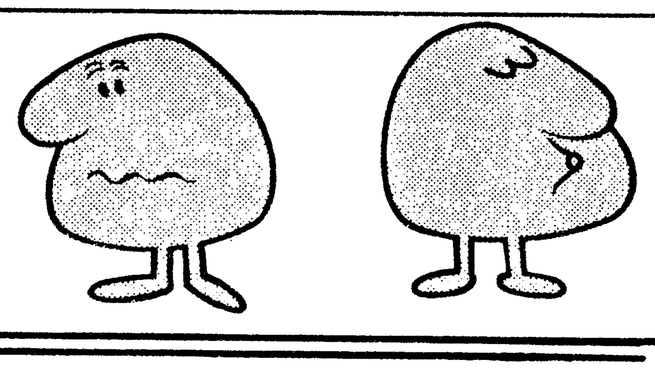
```
1 REM ================================
2 REM
3 REM
4 REM            Made by
5 REM         Fredrik Lundkvist
6 REM              1986
7 REM
8 REM
9 REM ================================
10 REM
11 REM
12 REM
13 REM
14 RAND -1
15 DIM C(9),CX(9),HL(9),HG(9),CG(9),RT(9),CH(9)
20 CTLSPR 2,1: CTLSPR 0,1: CTLSPR 6,0
30 VS 4: CLS : PRINT CHR$(31): GOSUB 20000: CSR 5,5: PRINT "please wait": GOSUB 3300
40 VS 4: PAPER 1: INK 7: COLOUR 4,1: CLS : PRINT CHR$(31): SPRITE 1,0,100,200,0,0,15
42 CSR 8,1: PRINT "Master Mind"
44 CSR 3,3: PRINT "Solve the hidden code."
46 CSR 3,4: PRINT "^he computer will give ": CSR 3,5: PRINT "you hints as follows:"
48 CSR 3,7: PAPER 6: INK 1: PRINT "J": INK 15: CSR 3,9: PRINT "J": PAPER 1: INK 7
49 CSR 5,7: PRINT "= right colour at right": CSR 7,8: PRINT "position.": CSR 5,9: PRINT "= right
 colour at wrong": CSR 7,10: PRINT "position."
50 CSR 3,12: INPUT "How many colours (4-9)?";A$
52 LET AC=VAL(A$): IF AC<4 OR AC>9 THEN  GOTO 50
54 CSR 3,14: INPUT "How many positions (4-9)?";A$
56 LET AH=VAL(A$): IF AH<4 OR AH>9 THEN  GOTO 54
60 RESTORE 61: FOR I=1 TO 9: READ J: LET CX(I)=J: NEXT
61 DATA    3,4,5,7,9,10,12,13,15
70 FOR I=1 TO AC
72 FOR CXX=1 TO 9: IF CX(CXX)<>0 THEN  GOTO 75
73 NEXT
75 CSR 3,16: PRINT "^hoose colour ";I;" (</>)"
77 CSR 4,18: INK CX(CXX): PRINT "I": INK 7: LET TET=1: GOSUB 500: LET TET=0
80 LET CXX=CXX+XF
81 IF CXX>9 THEN  LET CXX=1 ELSE  IF CXX<1 THEN  LET CXX=9
82 IF CX(CXX)=0 THEN  GOTO 80
84 IF TA=26 THEN  GOTO 90
85 SOUND 2,CXX*2+6,0: FOR Y=15 TO 5 STEP -2: SOUND 3,3,Y: PAUSE 10: SOUND 3,0,0: PAUSE 75: GOTO
77
90 LET C(I)=CX(CXX): LET CX(CXX)=0
91 GOSUB 600
94 IF ASC(INKEY$)<>-1 THEN  GOTO 94
95 NEXT I
100 PAPER 1: INK 7: CLS
101 LET X=10-AH
103 GOSUB 3000
105 CSR X,16: PRINT "[ change": CSR X,17: PRINT "] colour": CSR X,19: PRINT "< left": CSR X,20:
PRINT "> right": CSR X,22: PRINT "/ enter"
109 LET XM=2*AH+3
110 CRVS 6,1,30-XM,0,XM+1,24,32
120 VS 6: PAPER 6: INK 1: CLS : CSR 0,0: PRINT "A": CSR XM,0: PRINT "B": CSR 0,23: PRINT "C";: C
SR XM,23: PRINT "E";
130 LET B$="FFFFFFFFF": LET C$="GGGGGGGGG": LET A$=C$(1,AH)+" "+B$(1,AH): FOR Y=3 TO 23: CSR 1,Y
: PRINT A$;: NEXT
140 LET B$="       ": LET C$=B$(1,INT((AH-4)/2+.5)): LET D$=B$(1,INT((AH-4)/2)): LET A$=C$+"KNPRU
L"+D$: LET B$=C$+"  OQTV"+D$
150 PAPER 9: INK 6: CSR AH+1,0: PRINT A$: CSR AH+1,1: PRINT B$: PAPER 6: INK 1
155 GOSUB 1000
160 LET RO=23
162 SPRITE 1,0,100,200,0,0,0
190 FOR I=1 TO AH: LET HG(I)=C(1): LET CG(I)=1: NEXT
200 FOR I=1 TO AH: LET HL(I)=HG(I): NEXT
205 LET XT=AH+1: LET XC=1
210 SPRITE 1,0,(30-XM+XT+XC)*8+4,187-RO*8,0,0,15: CSR XT+XC,RO: INK HL(XC): PRINT "I";
220 GOSUB 500
230 IF TA=26 THEN  GOSUB 600: GOTO 300
240 LET XC=XC+XF
250 IF XC>AH THEN  LET XC=1 ELSE  IF XC<1 THEN  LET XC=AH
255 LET COL=CG(XC)
260 LET COL=COL+YF
270 IF COL>AC THEN  LET COL=1 ELSE  IF COL<1 THEN  LET COL=AC
275 LET CG(XC)=COL
280 LET HL(XC)=C(COL): LET HG(XC)=HL(XC)
290 PAUSE 100: GOTO 210
300 FOR I=1 TO AH: CSR XT+I,RO: INK HL(I): PRINT "I";: NEXT
```

```
305 GOSUB 2000
310 IF AS=AH THEN  GOTO 400
320 IF AS=0 THEN  GOTO 330
325 FOR I=1 TO AS
326 CSR I,RO: INK 1: PRINT "J";
327 SOUND 2,239,14: PAUSE 40: SOUND 2,119,14: PAUSE 40: SOUND 2,60,14: PAUSE 40: SOUND 2,0,0
328 PAUSE 100
329 NEXT
330 IF AV=0 THEN  GOTO 340
335 FOR I=1 TO AV
336 CSR AS+I,RO: INK 15: PRINT "J";
337 SOUND 2,60,14: PAUSE 40: SOUND 2,119,14: PAUSE 40: SOUND 2,239,14: PAUSE 40: SOUND 2,0,0
338 PAUSE 100
339 NEXT
340 VS 4: CSR X,1: PRINT "    ": CSR X,3: PRINT "     ": CSR X,4: PRINT "     ": VS 6
342 LET RO=RO-1: IF RO>4 AND RND>.94 THEN  VS 4: CSR X,1: PRINT "#uy": CSR X,3: PRINT " ^he": CS
R X,4: PRINT "$all": VS 6
345 IF RO>2 THEN  GOTO 200
350 FOR I=0 TO 2: RESTORE 389
360 FOR J=1 TO 7: READ LJ,JL: FOR Y=15 TO 5 STEP -3: SOUND 2,LJ*2^I,Y: SOUND 1,JL*2^I,Y: NEXT :
NEXT
370 NEXT
380 SOUND 2,0,0: SOUND 1,0,0
389 DATA     127,142,142,159,159,179,179,196,196,213,213,239,239,253
390 GOTO 450
400 FOR I=1 TO 4
410 FOR Y=400 TO 100 STEP -18: SOUND 1,Y,14: SOUND 2,Y+4,15: NEXT
42^ FOR Y=100 TO 400 STEP 18: SOUND 1,Y+4,14: SOUND 2,Y,15: NEXT
43  EXT
440 SOUND 2,0,0: SOUND 1,0,0
450 LET B$="YYYYYYYYY": LET A$="X"+B$(1,AH)+"Z": CSR AH+1,1: INK 1: PRINT A$
452 CSR AH+1,0: PRINT " ";: FOR I=1 TO AH: INK RT(I): PRINT "I";: NEXT : PRINT " ";
459 VS 4: CSR X,1: PRINT "Do you": CSR X,2: PRINT "want to": CSR X,3: PRINT "play": CSR X,4: PRI
NT "again?"
460 CSR X+6,4: PRINT CHR$(30);
470 LET A=ASC(INKEY$): IF A=-1 THEN  GOTO 470
475 CSR X+6,4: PRINT CHR$(31)
480 IF CHR$(A)="y" OR CHR$(A)="Y" THEN  GOTO 40
485 IF CHR$(A)<>"n" AND CHR$(A)<>"N" THEN  GOTO 460
487 RESTORE 496: READ N: FOR I=1 TO N
489 READ LJ,JL,ST: FOR J=LJ TO JL STEP ST: SOUND 2,J,15: SOUND 1,J+3,14: NEXT : NEXT
491 SOUND 2,0,0: SOUND 1,0,0: SOUND 0,0,0
495 VS 5: CLS : STOP
496 DATA    8,127,152,2,142,169,2,159,189,2.6,179,206,2.4,196,223,2.4,213,249,3.2,239,109,-14,11
9,60,-8
500 LET TA=ASC(INKEY$)
502 IF TET=0 AND RND>.9997 THEN  VS 5: PAUSE 600: VS 6
505 IF TA=-1 THEN  GOTO 500
510 IF TA=8 THEN  LET XF=-1: GOTO 560
520 IF TA=25 THEN  LET XF=1: GOTO 560
530 IF TA=10 THEN  LET YF=1: GOTO 570
540 IF TA=11 THEN  LET YF=-1: GOTO 570
550 LET XF=0: LET YF=0: RETURN
560 LET YF=0: RETURN
57   ET XF=0: RETURN
600 OR Y=15 TO 0 STEP -1: SOUND 1,900,Y: SOUND 0,905,Y: PAUSE 5: NEXT : SOUND 0,0,0: SOUND 1,0,
0
699 RETURN
700 DATA  etfetdfteiiiefdtfefdtfetfetdfetdftedfdte
705 DATA  etfetdfteidtifdileidtietietdfetdftedfdte
710 DATA  eretfetdfiiidteidtiedifeifetdfetdftedfdo
715 DATA  etfetdfteidtifdifeidtfiiietdfetdftedfdte
720 DATA  etfetdfteiiiifdoiioitfetietdfetdftedfdte
725 DATA  etfetdftefdtefdtfefdtiiiejtdfetdftedfdte
730 DATA  etfetdftefdtefdtfefdtfetfetdfetdftedfdte
735 DATA  etfetdiiiiiiefiii|kiii|kiiiiiiidftedfdte
740 DATA  etfetditiidiefdiipoiigytuiidewidftedfdte
745 DATA  etfetdftiidtefdiihuiiftrtiidfetdftedfdte
750 DATA  etfetdftiidtefdiiiiiifetfiiiiitdftedfdte
755 DATA  etfetdftiidtefdiitfiifetfiidfetdftedfdte
760 DATA  etfetdftiidtefdiiytiifetfiidfeidftedfdte
762 DATA  etfetdfiiiitefiiitriiietiiiiiiidftedfdte
765 DATA  etfetdftefdtefdtfefdtfetfetdfetdftedfdte
770 DATA  eetiiiefdiiidtfeiitfetiiidfetdfiiifdteed
772 DATA  ettdiiefdiifdtfiiiifetfiidfetdftiifdteed
775 DATA  etfdiiefdiifdtiifdiietfiidfetdftiifdtede
780 DATA  ettdiiefdiifdtiifdiietfiidfetdftiifdtede
785 DATA  eetdiieidiifdtiiiiiietfiidfetdftiifdtede
790 DATA  ettdiieidiifdtiifdiietfiidfeitftiifdtide
795 DATA  etdefiifiiefdtiifdiietfiiiiiidftiiiiiide
1000 FOR I=1 TO AH: LET SL=INT(RND*AC)+1: LET RT(I)=C(SL): NEXT : RETURN
2000 LET AS=0: LET AV=0
```



```
2010 FOR I=1 TO AH: IF RT(I)=HL(I) THEN  LET AS=AS+1: LET CH(I)=1: LET HL(I)=100 ELSE  LET CH(I)
=0
2020 NEXT
2030 FOR I=1 TO AH: IF CH(I)=1 THEN  GOTO 2050
2040 FOR J=1 TO AH: IF RT(I)=HL(J) THEN  LET AV=AV+1: LET HL(J)=100: GOTO 2050
2045 NEXT J
2050 NEXT I
2099 RETURN
3000 IF AH<6 THEN  LET A$="Micro": GOTO 3050
3010 IF AH<8 THEN  LET A$="Medium": GOTO 3050
3020 LET A$="Macro"
3050 IF AC<6 THEN  LET B$="Mini": GOTO 3100
3060 IF AC<8 THEN  LET B$="": GOTO 3100
3070 IF AC<9 THEN  LET B$="Super": GOTO 3100
3080 LET B$="Mega"
3100 CSR X/2.5+1,7: PRINT B$: CSR X/1.5+1,8: PRINT A$: CSR X+1,9: PRINT "Master": CSR X*1.5+1,10
: PRINT "Mind"
3199 RETURN
3300 RESTORE 700: POKE 64862,0: OUT 2,0: OUT 2,92: FOR Q=1 TO 22: READ A$: FOR W=1 TO 40: IF A$(
W,1)="i" THEN  OUT 1,124 ELSE  OUT 1,32
3310 NEXT : NEXT
3315 POKE 64862,15
3320 RETURN
20000 REM RUNDA HORN
20010 GENPAT 0,ASC("A"),248,224,192,128,128,0,0,0: GENPAT 0,ASC("B"),31,7,3,1,1,0,0,0: GENPAT 0,
ASC("C"),0,0,0,128,128,192,224,248
20020 GENPAT 0,ASC("E"),0,0,0,1,1,3,7,31
20030 REM HOLES AND BUTTONS
20040 GENPAT 0,ASC("F"),0,0,0,24,24,0,0,0: GENPAT 0,ASC("G"),0,0,0,16,0,0,0,0: GENPAT 0,ASC(")
,0,60,126,126,126,126,60,0
20050 GENPAT 0,ASC("J"),0,0,56,56,56,0,0,0
20060 REM MASTER MIND
20070 GENPAT 0,ASC("K"),0,32,16,16,42,34,36,36: GENPAT 0,ASC("L"),0,16,232,136,16,208,144,136: G
ENPAT 0,ASC("N"),0,132,138,138,129,157,144,160
20080 GENPAT 0,ASC("O"),0,32,16,16,42,34,36,36: GENPAT 0,ASC("P"),0,2,13,16,0,1,145,130: GENPAT
0,ASC("R"),0,2,108,8,8,8,8,8
20085 GENPAT 0,ASC("Q"),0,146,145,144,146,146,146,146
20090 GENPAT 0,ASC("T"),0,32,36,164,164,36,36,32: GENPAT 0,ASC("U"),0,4,120,64,16,96,64,4: GENPA
T 0,ASC("V"),0,128,64,64,64,64,64,128
20100 REM KANTEN RUNT
20110 GENPAT 0,ASC("X"),0,16,16,16,16,0,63,0: GENPAT 0,ASC("Y"),0,0,0,0,0,0,255,0: GENPAT 0,ASC(
"Z"),0,2,2,2,2,2,252,0
20120 REM PILAR OCH GREJS
20130 GENPAT 0,ASC("["),16,56,124,16,16,16,16,0: GENPAT 0,ASC("]"),16,16,16,16,124,56,16,0: GENP
AT 0,ASC("<"),0,32,96,254,96,32,0,0
20140 GENPAT 0,ASC(">"),0,8,12,254,12,8,0,0: GENPAT 0,ASC("/"),174,234,174,0,174,236,168,174
20150 REM TECKNEN
20160 GENPAT 0,ASC("D"),60,82,18,34,34,162,68,120: GENPAT 0,ASC("H"),34,34,34,76,116,68,132,132:
 GENPAT 0,ASC("M"),34,54,42,66,66,66,130,130
20170 GENPAT 0,ASC("S"),56,68,64,48,8,132,132,120: GENPAT 0,ASC("?"),56,68,4,8,16,0,32,0
20175 GENPAT 0,ASC("^"),62,208,16,16,16,32,32,0: GENPAT 0,ASC("."),0,0,0,0,0,24,24,0
20177 GENPAT 0,ASC("#"),60,82,18,44,34,34,194,124: GENPAT 0,ASC("$"),18,34,66,66,130,146,84,40:
GENPAT 0,124,255,255,255,255,255,255,255,255
20180 REM SMO BOKST.
20190 GENPAT 0,ASC("a"),0,0,24,36,68,76,52,0: GENPAT 0,ASC("b"),32,32,56,68,68,72,48,0: GENPAT 0
,ASC("c"),0,0,24,32,64,64,56,0
20200 GENPAT 0,ASC("d"),4,4,28,36,68,76,52,0: GENPAT 0,ASC("e"),0,0,24,36,88,64,56,0: GENPAT 0,A
SC("f"),8,20,16,56,32,32,32,0
20210 GENPAT 0,ASC("g"),0,0,28,34,34,26,4,56: GENPAT 0,ASC("h"),32,32,56,36,68,68,72,0: GENPAT 0
,ASC("i"),8,0,16,32,32,32,16,0
20220 GENPAT 0,ASC("j"),4,0,8,8,8,8,8,48: GENPAT 0,ASC("k"),32,32,40,48,96,80,72,0: GENPAT 0,ASC
("l"),16,16,16,32,32,32,16,0
20230 GENPAT 0,ASC("m"),0,0,20,42,42,84,84,0: GENPAT 0,ASC("n"),0,0,24,36,36,72,72,0: GENPAT 0,A
SC("o"),0,0,24,36,68,72,48,0
20240 GENPAT 0,ASC("p"),0,0,24,36,36,56,64,64: GENPAT 0,ASC("q"),0,0,28,36,36,56,8,8: GENPAT 0,A
SC("r"),0,0,44,48,32,64,64,0
20250 GENPAT 0,ASC("s"),0,0,28,32,16,8,112,0: GENPAT 0,ASC("t"),16,16,16,56,32,32,16,0: GENPAT 0
,ASC("u"),0,0,36,36,68,72,52,0
20260 GENPAT 0,ASC("v"),0,0,100,36,36,40,112,0: GENPAT 0,ASC("w"),0,0,36,36,84,84,40,0: GENPAT 0
,ASC("x"),0,0,36,40,16,40,72,0
20270 GENPAT 0,ASC("y"),0,0,34,34,18,12,4,56: GENPAT 0,ASC("z"),0,0,60,8,16,32,120,0
20280 REM NUMBERS
20290 GENPAT 0,ASC("4"),18,18,34,36,124,4,4,0: GENPAT 0,ASC("1"),8,8,8,8,16,16,16,0: GENPAT 0,AS
C("2"),24,36,4,8,16,32,120,0
20292 GENPAT 0,ASC("3"),60,4,8,4,4,68,56,0: GENPAT 0,ASC("5"),30,16,56,4,4,68,56,0: GENPAT 0,ASC
("6"),8,16,32,120,68,68,56,0: GENPAT 0,ASC("7"),62,2,4,4,8,8,16,0
20294 GENPAT 0,ASC("0"),28,34,34,66,68,68,56,0: GENPAT 0,ASC("8"),24,36,40,24,36,68,56,0: GENPAT
 0,ASC("9"),28,34,34,28,8,16,32,0
20300 GENPAT 0,ASC("("),4,8,16,16,16,16,8,0: GENPAT 0,ASC(")"),16,8,8,8,8,16,32,0: GENPAT 0,ASC(
"-"),0,0,0,60,0,0,0,0
20310 GENPAT 0,ASC("~"),14,16,32,32,64,64,34,28
20800 REM SPRITE
```

```
20810 GENPAT 3,0,255,255,195,195,195,195,255,255
20999 RETURN
49998 REM This is the autostart line
49999 REM
50000 SAVE "Master Mind": RUN ★
```



Shell Sort by Neil Emsen

This shell sort uses Eric Roy's USER BASIC as amended to include the EOF routine at Line 1. It is written in this format to enable the program to be used as a base program to create new programs.★

```
0 CODE

4007          DS 250     ;--USER BASIC--
4101          DS 250
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

4D35 SETUP:   LD A,02
4D37          LD (£4314),A
4D3A          LD HL,UBEOF
4D3D          LD (£FA8D),HL
4D40          RET
4D41 UBEOF:   LD (£4BBD),DE
4D45          LD A,(DE)
4D46          CP "E"
4D48          JR Z,EOFCOM
4D4A EXIT:    LD A,(£FAD2)
4D4D          LD DE,(£4BBD)
4D51          JP £F5B6
4D54 EOFCOM:  LD HL,EOFNAM
4D57          CALL SYNTAX
4D5A          JR NZ,EXIT
4D5C          INC DE
4D5D          RST 30
4D5E          LD A,B
4D5F          AND A
4D60          JR NZ,RANGE
4D62          OR C
4D63          JR Z,RANGE
4D65          CP 05
4D67          JR NC,RANGE
4D69          PUSH DE
4D6A          LD DE,£0028
4D6D          LD IX,£D818
4D71 FCBLP:   ADD IX,DE
4D73          DEC A
4D74          JR NZ,FCBLP
4D76          POP DE
4D77          BIT 7,(IX+£27)
4D7B          JP Z,£27AA
4D7E          JP £4189
4D81 RANGE:   RST 28
4D82          DB £22
4D83 SYNTAX:  LD B,(HL)
4D84 SYN1:    INC HL
4D85          INC DE
4D86          LD A,(DE)
4D87          CP (HL)
4D88          RET NZ
4D89          DJNZ SYN1
4D8B          RET
4D8C EOFNAM:  DB 03,"OF£"
4D90          RET
4D91          RET

Symbols:
SETUP   4D35    UBEOF   4D41
EOFCOM  4D54    EXIT    4D4A
EOFNAM  4D8C    SYNTAX  4D83
RANGE   4D81    FCBLP   4D71
SYN1    4D84
```




```
10 DIM A$(MAXNUM)
100 USER BASIC
49999 STOP
60000 USER @SHELLSORT
60010 PRINT : PRINT "SORTING NOW - PLEASE WAIT": PRINT
60020 LET G=MAXNUM
60030 USER REPEAT
60040 LET G=G/2
60050 LET C=MAXNUM-G
60060 USER REPEAT
60070 LET F=FALSE
60080 FOR I=1 TO C
60090 LET P=I+6
60100 REM       AMEND NEXT LINE TO SUIT PROGRAM
60110 IF A$(I)>A$(P) THEN  LET T$=A$(I): LET A$(I)=A$(P): LET A$(P)=T$: LET F=TRUE
60120 NEXT
60130 USER UNTIL F=FALSE
60140 USER UNTIL G<=1
60150 RETURN
```

MTX Tokens

A F Wilson

Have you ever wondered how the MTX's basic interpreter stores a basic line in memory? Well the MTX like the Sinclair ZX spectrum and Commodore machines stores the basic keywords as tokens. The keywords are converted into a ascii code between 128 and 255. If the user presses any of the function keys which represent ascii codes 128-136 and press enter then the keyword is displayed, i.e. F1=REM. This is a convenient starting point. Type in the basic line in fig 1. Now enter the front panel by typing PANEL (a full description of the front panel is found in issue 4 of Memopad). Now press D for display and enter 4000 (or 8000 on the 500, from now on the 500 owners should change any initial '4' with an '8'); at this point the memory block will be updated from 0000 to 4000 (or 8000) and the memory location and its contents will be displayed at the bottom of the screen. At this point press the <BRK> key in order to avoid changing any contents.

Fig 1:

```
10 REM
```

From location 4000 you will see the following 6 bytes of data:- 06 00 0A 00 80 FF.

0600 = The number of bytes used to describe line 10.
0A00 = The basic line number in hexadecimal. The line number is stored as LSB/MSB format. Note that the allocation of two bytes to represent a basic line number means that the MTX can have line numbers between 0 and 65535.
80 = This equals 128 decimal and represents the keyword for REM. A list of the main keywords, tokens and their address in rom is given later.
FF = This marker is used to indicate the end of this basic line.

Figure 2 shows where the next basic line would go.

Fig 2:

```
10 REM
20 GOTO 1
```

Line 20 gave 07 00 14 00 96 31 FF

The first 5 bytes are similar to above i.e. the length of memory used, basic line number and the token for GOTO. But what is #31. Well this is equal to the number 1. If you look at the ascii table in your MTX manual, appendix section, you will see that 31 hex or 49 dec equals the number 1. This means that if we were to goto line 100 for instance then we would need 3 ascii codes to represent this 3-digit number, i.e. 96 31 30 30 = GOTO 100.

Already this insight into the computers internal workings has given us a tip to save memory. Basic is generally greedy and inefficient as memory goes and this has telling affects when writing long pograms or programs like word processors which need as much free memory as possible. Therefore if we have many GOSUB's or GOTO's in a program then it would be more economical to have them at the beginning of the program, i.e. at low line numbers 0-99, and save a number of valuable bytes.

As you will soon see there are a number of ways to save memory by looking at the MTX's method of storing Basic.

One of these savings is to do with the GENPAT command. Genpat stores the data after the command as ascii codes as did the GOTO command. This means that the maximum number of bytes after the token will be 10x3+9=39, the 9 representing the commas. If you have ever used the built in assembler then you will know that a DB command plus 10 data only uses up 10 bytes as the data is stored as a hex number and not an ascii code. Therefore we need an extra 29 bytes of data not including the memory for the length, line number and token. It would be easier to store genpat data in the aassembler format and peek the data when needed or write a short machine code routine to do the same. All in all the savings are immense, e.g. 10 genpat statements will consume 390+50 bytes of data (maximum), whereas 10 DB

assembler commands use up about 100 bytes, a saving of 340 bytes.

Multistatement lines

Take the example given in figure 3 and the resulting memory data. As you can see only the REM keyword is assigned a token number whereas the GOTO command is stored as ascii codes. The number of bytes used to represent fig 3 equals the number used in fig 2 . This is because the 5 initial bytes used at the start of any basic line in this case is cancelled out by the 4 bytes for the GOTO and 1 for the colon. If we use Multistatements with keywords less than 4 characters long, i.e. PHI,CLS,REM, etc. then we will be saving a few bytes per line. But if the keyword is greater than 4 characters then we are eating memory up. In this case it would be better to use a separate line for keywords:-GOSUB, GENPAT and RESTORE which will save us 1, 2 or 3 bytes respectively, i.e. 1(colon)+7(for keyword,RESTORE)=8 compared with only 5 opening bytes for new line.

Fig 3:

```
10 REM:GOTO 1

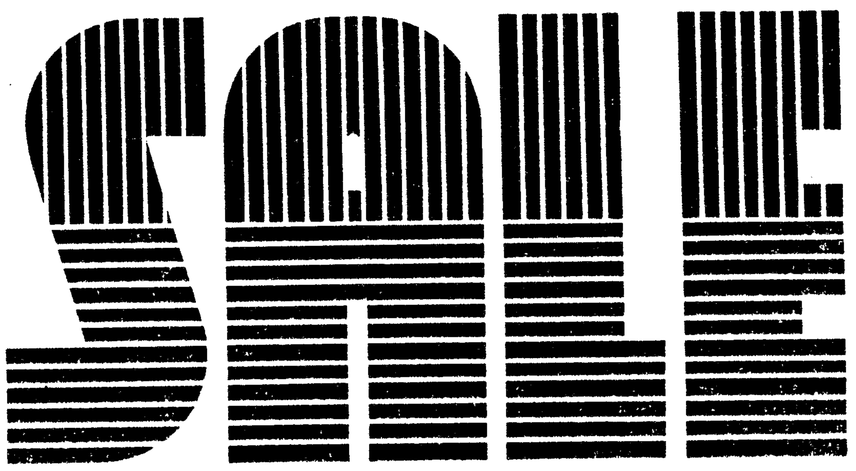
0D 00 0A 00 80 3A 47 4F 47 4F 20 31 FF
          REM :  G O T O sp  1 end     (sp=space)
```

As you can see, there are a numer of ways to make memory savings by taking a closer look at the MTX. I have only discussed a few. By experimenting you will soon see how to write efficient basic programs. The only complaint I have of Memotech is using two bytes to represent the length of line in relation to memory. I know we can redefine the editor window, but who would require a line of over 6 lines of 40 characters (>255). Not me. I think they should have limited the user to only the 4 lines of 40 characters as at present. This would save a lot of memory space.

Here is a sample of the main keywords their addresses in rom and their token. (Note that the MTX basic rom is on page 1 and this means that if we want to disassemble it we will have to move the rom into ram on page zero to get into the front panel. See appendix 8, Memory map in manual).

Getting Bored With Playing Games!

Why not try something new and exciting, MOC has plenty to offer, like:
MTX Fig-Forth (including Ram Disk) for a MTX512
Program £7.00
Tech. Data £2.50
Tutorial £8.00 (240 Packed Pages)
!!! or perhaps a DIY interfacing project !!!
For Beginners:
An LED kit, all parts for only £7.50
For the more ambitious:
A Speech Synthesiser kit, only £19.00
Internal MTX port connector cable £5.00 (needed for above). (fits into internal port 7, transfers to 20 way IDC connector via ribbon cable to outside of computer case).
or
Programs galore - The MOC program library!!
Over 50 programs only 75p each, for full list send a S.A.E. today, to:
MOC 23 Denmead Road, Harefield Southampton. SO2 5GS.
For more discounts and 10 great newsletter/magazines a year why not become a member, it's only £7.00 a year. (UK).





View Letters TO THE EDITOR point



Letter received from Mr F Harrison, Morley, Leeds

Here is my reply to Mr J Waller's letter in Issue No. 7 with regard to the M/C code re-number program published in the November P.C.W.

Yes I did notice it, it is an excellent M/C code program, it works exceedingly well and I can recommend it.

He also says, if he can do it why don't we submit similar programs.

I would dearly like to compile M/C code programs like that, but I haven't the faintest idea how to start. Some of it seems recognisable but only just.

Please Mr Waller - where do I purchase the down to earth book (and what is its title) which will show me (in simple terms) step by step, with sample programs and mnemonic tables how to compile such programs on my MTX.

Hopefully trying to keep the MTX flag flying (as yet only in Basic).

P.S. The new Memotech operator's Manual is an excellent book and far more informative and helpful than the original - a good buy.

Letter received from John Barnett, Sapcote, Leics.

Having just received MEMOPADS 7 and 8 I felt I ought to put pen to paper to thank Keith Hook for all the work he has put in and is continuing to put in, on behalf of we Memotech users - what phenomenal energy he must possess. Thanks must also go to all other people in the Syntaxsoft/Genpat Camp who must similarily be putting in a great deal of effort.

I must admit that my initial pessimism about the future for Memotech (the company and MTX users) has now been overtaken by some optimism and a good deal of excitement.

Letter received from J Nellis, Cawood Common, Nr Selby, North Yorks.

I thought I should pull my finger out and give the MTX and Memopad the support it deserves. I have written this 64 column character program, for consideration of publication in your magazine.

The concept has been kept as simple as possible so that MTX users can incorporate it into their own programs.

The program is simply a splicing mechanism, and data containing the '64 print' ASCII characters.

I have added a small Basic/machine code program, so that you can quickly test that the program works.

Anyway to operate the program on its own you need to know the following:

Poke 16507,0 or 1 : Position of the 64 character in the normal 32 character block.

```
Poke 16508,X        : Current X position
Poke 16509,Y        : Current Y position
Poke 16394,A        : ASCII character value to be printed
RAND USR(16391)     : Print the character
```

The program was written on the MTX 512, so if you wish to use it on the MTX 500 the Basic program should be accordingly changed.

I hope you do publish the program. If you write me any correspondence please be patient in waiting for a reply as I probably won't be home.

Letter received from S A March, Kilsyth, Glasgow

Like every other Memotech owner I am anxiously waiting to hear whether Memotech will be re-born or buried ignominiously. However, whatever may be the final outcome may I, on behalf of all other members of the "Club", thank you most sincerely for all your efforts to keep Memotech alive. I think special thanks are due to Keith Hook. Speaking personally, he helped me the most in those early days when I was beginning to despair of ever getting any real use out of my MTX 512 and gave the support and encouragement that I needed. However, I also want to thank all of you for the way that you have been so helpful in dealing with the queries of computer illiterates like me.

Now I am a simple soul and still have no idea what Macro Assembler, Edasm, RST 10 or RST 28 are. I have no idea how to use the Assembler and even when I enter the Panel I can't for the life of me understand it. The point that I am making is that there are many of us out here who are not computer buffs. Over and over again I have read in Memopad that the Memotech is a superior machine. I believe it, but when I see the cheap business software available for Spectrums, Amstrad, CBM 64's etc. I feel that Memotech has missed the boat.

It is heartening to read that Sagesoft's business packages may be available on the MTX disc system but that is only a "maybe". If Geoff Boyd is able to re-launch Memotech (and I sincerely hope that he can) then may I suggest that he gives serious consideration to negotiating with software houses for some of the excellent software that is available on other machines. I personally would like to see WIMPS and something like the capabilities offered on the Apple Mackintosh or Digital Research's Dr Draw or maybe GEM. Have I got my head in the clouds? Is all this impossible on the Memotech? My feeling is that if such things can be emulated on Amstrad etc. then Memotech can do them better .... but then I am only a simple soul.

Letter received from P R Boardman, Northallerton, N. Yorks

Ever since I first explored the Memotech computer, about 18 months ago, I have had difficulty with the GENPAT mode where p=2, i.e. a multi-coloured character. The original manual, articles and programs in Memopad and now the Bateson manual all tell me that one defines the ink and paper colours by making d1 etc. = 16*PAPER + INK. I do this but consistently get the colours the opposite

way round. Is there something wrong with my brain? or is it the ROM? or is everybody copying the original mistake without trying it out?

I have enclosed a very simple program to test it and would be obliged if you would get someone to try it out and let me know where the error lies.
May I take this opportunity of thanking you and all concerned in the production of Memopad, which improves with every issue. I do hope that Geoff Boyd and his team will make a success of their venture which must be fraught with dangers in the current climate.

Is there any sales literature on the 500K SDX, package ONE? as I wish to purchase one but would like to know the specification beforehand. Can you help?

```
10 GENPAT 1,147,0,0,0,0,255,255,255,255
20 GENPAT 2,147,166,166,166,166,166,166,166,166
30 REM Let PAPER=10 (Dark yellow) and  INK=6(Dark red) therefore p=16*10+6=166.
No need to type this in and don't ever use F1 instead of typing REM.  Try it.
40 VS 4: CLS
50 CSR 12,15: PRINT CHR$(147);CHR$(147);CHR$(147)
100 GOTO 100
```

Pascal *by N Joynson*

This article is aimed at users of Hisoft Pascal on the MTX, and particularly at those contemplating writing fast, efficient programs using it. In addition, some sections may be of use to machine code programmers wishing to know more about the MTX.

Writing software in assembler is all very well, but is extremely time consuming. Research shows that the average (professional) programmer will only produce 10 lines of reliable code per day, regardless of the language used. Thus it is sensible to write software in a high level languge, as each line of code does much more than, for example, a line of assembler. Also, the source code is compatible across a range of machines, so conversion (if required) is made relatively easy.

Hisoft Pascal is a reasonable implementation of Pascal and will allow programs developed using it to run without the ROM board present. Furthermore, it allows the full 64K of the '512s memory to be used, with up to 48K of that being pure object code if required.

For those wishing to write fast arcade type games Pascal is a good choice, as the object code produced is fast, and also because of the portability factor. However, input/output is not well supported by Pascal, and varying implementations give different levels of i/o support. In practice this means machine specific i/o needs to be written. 'Porting' to another machine then simply requires a rewrite of the i/o routines, which may amount to perhaps 10% of the total program code.

There are several ways of providing i/o via Hisoft Pascal:

1. Using the drivers built into the MTX's Basic ROM.

It is a little known fact that virtually all of the MTX's Basic graphics commands are available by printing control codes to the screen. For example, outputing CHR(18), followed by a list of parameters will issue the 'SPRITE' command. By outputing these control codes from Pascal using the format:

write(chr(<1st control code>),chr(<2nd control code>), etc...);
graphics may be created and animated at the same level as is possible from Basic. All full list of the commands available follows:

To PLOT	output	1,X,Y	To ADJSPR	output	17,P,N,V
" LINE	"	2,X1,Y1,X2,Y2	" SPRITE	"	18,N,PAT,XP,YP,
" CSR	"	3,X,Y			XS,YS,colour
" PAPER	"	4,colour	" MVSPR	"	19,P,N,D
" INK	"	5,colour	" VIEW	"	20,DIR,DISTANCE
" COLOUR	"	16,P,N	" ATTR	"	27,65,P,STATE
" CTLSPR	"	14 ,P,X	" GENPAT	"	15,P,N,D1,D2,D3,
					D4,D5,D6,D7,D8

However, there are some disadvantages to this method of i/o. Firstly, Hisoft Pascal intercepts certain codes, in particular CHR(16), which it uses to toggle the printer output. If this wasn't intercepted it would activate the COLOUR command. This problem can be circumvented to a certain extent by using INK and PAPER and to set the border addressing the VDP directly as follows:

```
procedure border (colour: 0..15);
begin
   out(2,chr(colour));out(2,chr(#87));(this selects a write to reg7)
end;
```

Secondly, sound and keyboard reading is not possible using this method. True, the INCH procedure can be used for the keyboard, but this is very slow - try running a program using the INCH procedure and then running it without to see the speed increase.

Thirdly, the screen output routines are relatively slow. Hisoft Pascal seems to take some time to decode the WRITE procedure before control is passed to the Basic ROM (using RST10), which isn't particularly quick by any means.

2. Using assembler to access the ROM routines.

The second method available is to use the routines published in Memopad volume 1 , issue 4. These provide full support of the RST10 vector from Pascal, by poking the parameters into memory, the performing a RST10 on them from assembler.

These routines work well, but again provide no support of sound, and also force the programmer to use the INCH procedure for keyboard reading. Also, they are not that quick, and so will slow down a program which uses them noticeably.

3. Writing i/o routines in assembler.

A third method for implementing low level i/o is to write the i/o you require in assembler, and interface it to the Pascal program.

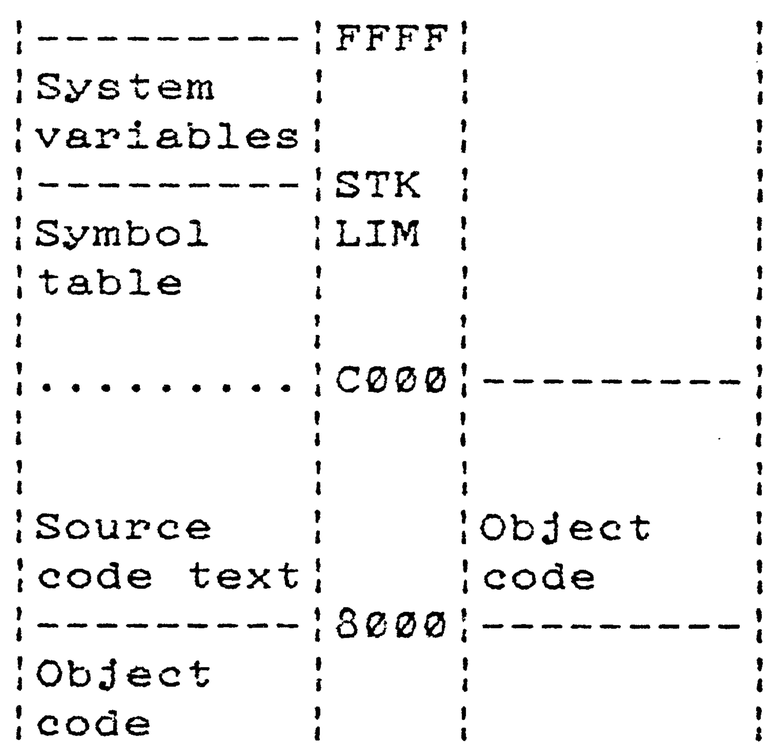
This method has advantages in that only the routines which are required for a given program need to be written, and most importantly, they complement the speed of the compiled Pascal. However, it does of course require the programmer to be able to program in assembler as well as Pascal.

First question - where can you put the assembled code. Well, taking a quick look at the memory map while running Pascal:

As it shows, the object code can span 2 pages, so any machine code routines must be located in memory that is common to both pages-that is between £C000 and £FFFF.

The MTX has a system variable called STKLIM located at £FA92 which effectively defines the top of available memory. Moving this down will allow us to reserve some space between the system variables and the symbol table.

Nominally STKLIM is set to £F8F2, so setting it to £F4F2 will give us 1K to play with. Go into the PANEL and alter STKLIM to the value required, and memory will



be reserved. The next problem is to get your assembler routines up into the reserved memory. If you own EDASM then simply change the ORG statement to the start of the free space. If you haven't got EDASM and you only have the built in assembler, it is possible to relocate the code, but it is not easy. You could simply move the assembled code using the M command, and then change the CALLs and JUMPs by hand. Alternatively:

	6269	
Runtime routines		
	4000	
ROMS		ROMS
	0000	

A. Set STKLIM as high as possible - it will go to #FA52 without crashing.
B. If your assembler routines are in line 10, then create a line 1 of code. Create as many DS 254s as possible to this line.
C. After a short time the MTX will give a No Space error. This means your routines are now at the very top of memory on page 0. Now create a line 2 as a REM statement and fill it (all 4 lines) with characters.
D. Do this several time, creating more lines the same as line 2, between line 1 and your own routines. When you hit RET on the 4th or 5th REM statement there will be a short pause as your routines are moved onto page 1. Listing your routines should show them to be at around #8007 .
E. Now create a line 9 of code and again put as many DS 254s in it as possible.
F. The MTX will eventually give a No Space error again. This indicates your routines are at the top of memory. Now list your routines, noting the start address. Now enter the PANEL and change STKLIM to just below the start address of your routines.
G. Exit to Basic and type NEW.
H. Now enter the PANEL and set STKLIM back to the start address of your routines. Your routines are now stored above STKLIM, with all the JUMPs and CALLs correctly assembled.

Although no Basic program is present, the code may now be saved using the Basic SAVE command, and loaded using the corresponding LOAD command. Now you have your routines stored above STKLIM, these should b loaded in before running any Pascal programs.

In most cases parameters will have to be passed to these routines, and occasionally parameters will be passed out. Probably the best method for doing this is to use blocks of memory as parameter blocks and poke parameters into the block from Pascal, then retrieve them in assembler.

This memory block could be in the area you reserved for your routines, or in part of the system variables. A good area is where the Basic USER command and jump is stored, at #FA85. If this is being used then possibly one of the unused virtual screens could be used (#FF5D-#FFD4).

If you have 6 bytes to pass values in this will allow 6 bytes or 3 addresses to be passed, which should be adequate for most purposes. For routines which only have parameters passed to them, a procedure could be written which pokes the parameters into memory and calls the routine required, e.g.:

```
procedure callmc(startaddr,param1,param2,param3:  integer);
const
     paramblock =     #   f  a85;
begin
     poke(paramblock,param1);
     poke(paramblock+2,param2);
     poke(paramblock+4,param3);
     user(startaddr);
end;
```



If single bytes are to be passed, this would be achieved by passing a parameter as

```
x+256*y
```

which will put the x into the 1st location and y into the 2nd.

As an example, if you had a routine at #F600 which places a sprite on the screen with X,Y,PTTERN,COLOUR attributes, using the above procedure to call it would use a call like this:

```
callmc(#F600,x+256*y,pattern+256*colour,0);
```

The assembler routine at #F600 would then have to retrieve the parameters from memory at #FA85-FA88.

For assembler routines returning parameters, such as tests for sprite collision or keyboard reding, individual interface procedures or functions should be written. For example, to read a key directly, if the assembler routine was:

```
readkey:    ld hl,(ffa85)       ;  get drive lines and port to read from
            ld c,5              ;  output port
            di
            out (c),1           ;  drive required keyboard line(see
            ld c,h              ;  Memopad,vol2,issue1)
            in a,(c)            ;  get keys pressed on drive line
            ei
            cpl                 ;  complement
            ld c,a
            ld a, (#fa87)       ;  get mask for particular key
            and c               ;  test actual key
            ld (#fa85),a        ;  returns a 1 or a 0 for key pressed -
            ret                 ;  corresponds to TRUE and FALSE in Pascal.
```



This could be called from:

```
function readkey (sensebyte, readport, bitselect: 0..255):boolean;
begin
      poke(paramblock, sensebyte); poke(paramblock+1,readport);
      poke(paramblock+2,bitselect); user(<readkey address>);
      readkey:=peek(paramblock,boolean);
end;
```

This could now be used as a standard function from Pascal.



Special Features

The MTX's interrupts work fully from Pascal, so it is quite possible to use interrupts from Pascal programs. The most obvious use of this is to drive sound routines which use ADSR techniques to provide enveloping. The actual routines are beyond the scope of this article, but with a little assembler it is quite feasible to produce music of the quality of 'ROLLA BEARING' and to control it from Pascal.

Activating interrupts from Pascal is exactly the same as for Basic: simply make the interrupt vector at #FA98 point to your interrupt service routine, then poke #FD5E with #1F to enable it. Your routine will then be executed every 20ms.

Another use of interrupts is to provide flicker free sprites. The interrupt mentioned above is provided by completion of the raster scan via the VDP chip. Basically, while the raster beam flies back to the top left of the screen, the VDP can be driven a lot faster than normal. It is capable of outputting 512 bytes to the screen during flyback.

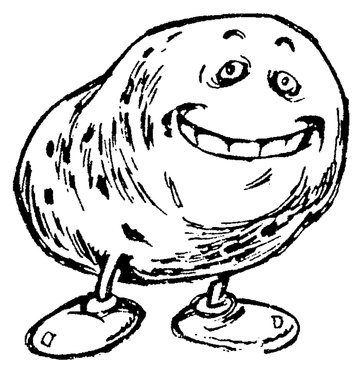
The major use of this is to manipulate sprites. Normally to move sprites the sprite attribute table in the video memory must be changed by using the VDP. This is not veery quick and can cause sprite flicker. A far better way of manipulating sprites is to use a section of normal memory as a carbon copy of the sprite attribute table, and poke your sprite attributes into it when any sprite movement is required. Meanwhile, every 20ms an interrupt service routine outputs the 'shadow' sprite attribute table to video memory during the raster flyback. Thus the real sprite attribute table is kept up to date constantly.

This requires 128 bytes (32*4) of spare memory to act as the shadow sprite attribute table. If ROM routines are not being used to mnipulate sprites then an ideal place is the ROM's own sprite map at #FE55, which is 256 bytes long.

A suitable assembler routine to provide this sprite update follows:

```
isr: ld  de,#7f00    ;  address in vram of attr table + ctrl bits
     ld  c,2
```

```
      out (c),e
      out (c),d
      ld hl,ffe55     ;  address of sprite table in memory
      ld bc,#8001     ;  set o/p port to 1,number of bytes to 128
loop:oti              ;  output (HL) to port 1
      nop             ;  wait a bit-even flyback takes time
      nop
      djnz loop       ;  do it again
      ret
```



I hope this article will be of use to MTX Pascal users. As a finale I will summarize a few of the limitations of Hisoft Pascal which users should be aware of. These limitations are present in my ROM (v1.4) which I assume is the up to date one.

1. If the GET command is used and the last basic command executed was VERIFY then Pascal will continually give a MISMATCH error. To get around this enter PANEL and alter #FD67 to a 00.

2. If subrange types are used, subranges cannot be declared more than 32767. Also subranges less than 256 still take 2 bytes and poking them also pokes 2 bytes.

3. The RST 38 vector, which can be used as a breakpoint in assembler is used and redirected by Pascal, so using this may crash the machine.

4. There is a major bug in the pointer routines in Pascal. If a pointer is set to point to another record containing a pointer, occasionally the program is corrupted and the machine crashes after running.

Attention Getter by D Clement

```
10 CLS
20 LET A$ = "MESSAGE"      :    REM ** WHATEVER YOU NEED
30 LET B$ = "-------"      :    REM UNDERLINE CHARACTERS
40 FOR A = 1 TO 6 <---     OR WHATEVER LENGTH YOU LIKE
50 CSR 7,7 :  PRINT B$
60 SOUND 0,200,15
70 SOUND 1,270,15
80 PAUSE 200
90 SOUND 0,0,0 : SOUND 1,0,0
100 CSR 7,7:PRINT A$  PAUSE 350
110 NEXT
```

B$ SHOULD BE THE SAME LENGTH AS A$; AND IF IT WAS A SUBROUTINE
USED A LOT THEN THE ' MESSAGE ' COULD BE GOSUBBED AND CONVERTED
TO A$ AT THE SUBROUTINE, E.G. LET A$ = X$
A GOOD EXCERCISE WOULD BE TO WORK OUT A ROUTINE TO PACK B$ WITH
LEN(A$) UNDERLINE CHARACTERS
ALSO HERE IS A ONE LINER TO DISPLAY A CURSOR FOR THE INVISIBLE
INKEY$ FUNCTION

```
10 LET A$ = INKEY$ : IF A$ = ""  THEN PRINT ">";: PAUSE 300
   PRINT CHR$(8);: PRINT "Δ";:  PRINT CHR$(8);: PAUSE 300 : GOTO 3000
```

AND ANOTHER INKEY$ FUNCTION TO ALLOW A TWO CHARACTER INPUT.
YOU COULD USE THE INPUT COMMAND OF COURSE BUT WITH THIS ROUTINE
YOU DONT HAVE TO PRESS RETURN

```
5 LET G$ = ""    REM ** NULL CHARACTER                    REM ** "Δ" IS A SPACE
10 GOSUB 30000             REM *** REMAINDER OF PROGRAM.....
29999 STOP
30000 LET A$ = INKEY$
30005 PAUSE 100   REM ** JUST ENOUGH NOT TO GET KEY ROLLOVER
30010 LET G$=G$+A$ : IF LEN  (G$)=2 THEN G = VAL(G$) ELSE GOTO 30000
30015 PRINT G
30020 RETURN
```



SOFTWARE



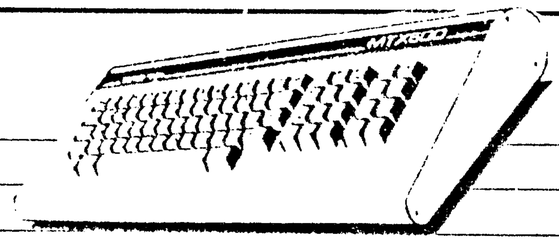
00106	26X26 SPREAD SHEET	UTIL	SYNT	7.95	I	ANY
00057	3D TACHYON FIGHTER	ARC	CONT	6.95	I	ANY
00135	9 ELECTRIC. PROGS	EDUC	SSFT	13.95	U	ANY
00062	ADVENTURE QUEST	ADV	LVL9	8.75	I	ANY
00033	AGROVATOR	ARC	SYNT	5.95	I	512
00125	AIRCRAFT MAGNETISM	FLGT	AVTN	?	I	ANY
00120	AIRCRAFT PAYLOADS	FLGT	AVTN	?	I	ANY
00122	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
00141	ASSEM LANG COURSE	EDUC	GRAY	8.95	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	BLOBBO	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	U	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	U	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
00037	FLUMMOX	ARC	SYNT	5.95	I	512
00132	FRACT, PERCENTAGES	EDUC	SSFT	5.95	U	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
00139	HIGHWAY ENCOUNTER	ARC	VORT	7.95	I	512
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
00086	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	U	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 B.	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
00140	SNOOKER	SPRT	MAG7	7.95	I	512
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	U	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	U	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	U	ANY
00006	TOADO	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	U	ANY
00060	WORD & PICTURE	EDUC	CONT	8.75	I	ANY

PLEASE ONLY ORDER THOSE MARKED " I "

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

KEY:- STOCK NUMBER TITLE TYPE HOUSE MEMBER PRICE STOCK

HARDWARE



PRICES EFFECTIVE FROM 1ST DECEMBER 1985

MTX 512 ..... #119.00 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

PRINTER CABLE .....# 8.95

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

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

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

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.

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

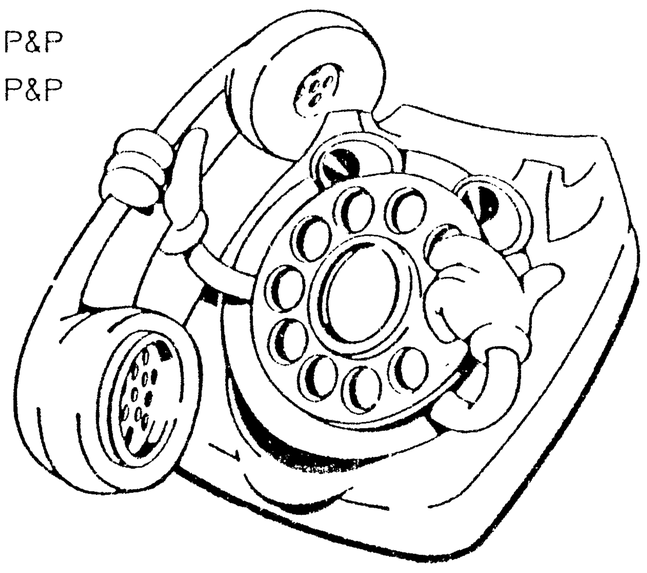
PACKAGE FOUR
AS ABOVE BUT WITH 1MEG DRIVE .....#489.00

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)

ALL HARDWARE SUBJECT TO £5.00

POSTAGE AND PACKING



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

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.

275 JADO SPEED SPLITTER WITH SOFTWARE £20.70

This file was downloaded from

www.primrosebank.net

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.

Unless otherwise noted, all data on the website is deemed to be ***Copyright (c) Dave Stevenson, 2009-2013***

You are hereby granted permission to download data and software from the website ***for your own personal use. Redistribution of any content from the website without written authorisation from Dave Stevenson is expressly forbidden. You are also expressly forbidden from offering for sale any material obtained from the website.***

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.