9

kph computaware

63 Highlands Road,
Andover, Hampshire,
England, SP10 2PZ

CORTEX USER GROUP NEWSLETTER (FEBRUARY 1987)

Issue Number 9

CONTENTS

Proprietor: K.P.Holloway BSc.

"CORTEX USERS GROUP 1987"

Due to increase in personal commitments by Kevin Holloway the Cortex news letter will be taken over and run jointly by Tim Gray and Ted Serwa.

Tim has a background in the television industry and Ted works in the telecomunications industry. Both have extensive Cortex based systems and have aquired a vast knowledge of the Cortex hardware and software. They are at the moment gathering information on all available hardware and software in the aim of making it all available from the same source. Tim is designing a 512K DRAM card for the E.Bus and will be carrying on with his E.Bus articles. Ted is working on a Cortex E.Bus compatible 80 colum / high definition graphics card with on board processor.

The user group magazine will continue to include as much member supplied material as possible and will encourage software and hardware exchange between members at minimal cost.

The membership renewal for 1987 is now due, the cost will remain the same as last year, £5.00 for twelve months. Cheques made payable to "CORTEX USERS GROUP" should be forwarded to:-

"CORTEX USERS GROUP"
C/O Tim Gray,
1 Larkspur Drive,
Featherstone,
Wolverhampton,
West Midlands,
WV10 7TN.
ENGLAND.

PROGRAMS.

Our first two programs were sent in by W.D.Eaves from Caithness. The first of these is a program called KEYS which can be used to create a user defined key set on the top row of keys when used with the GRAPH key. If a disk drive is used then the program can be autorun at BOOT time(see newsletter 6 page 13 to autoload a given filename).

Once the program has been run then a string of characters can be printed by pressing one key. Users can define their own set of labels simply by altering the data in lines 130 & 135.

The program stores the data and machine code at locations 5fb0h to 600ch. Because I use the program at BOOT time I have included lines 55 and 60 to load other programs. If this option is used it is important that subsequent m/code programs do not overwrite the above locations. I relocated the FIND program at 6010h and the CAT program at 7000h. If using CAT with the keys program then change the CAT buffer from 5fc0h to another value or the key label m/code will be corrupted. The BASIC start address needs to be raised to use the locations I have mentioned; see newsletter 7 page 13 or if not using a disk drive system set MWD(0ed04h)=7114h and MWD(ed06h)=7114h.

KEYS

```
10   DIM $C[2]
15   TEXT : COLOUR 1,13:
20   ? : ? "   Key           Label": ?
25   A0=05FB0H: KD=05F00H: RESTOR 105
30   FOR F=0 TO 47: READ A: MWD[A0+F*2]=A: NEXT F
35   MWD[048EH]=0460H: MWD[0490H]=A0
40   FOR N=176 TO 185: X=N-176: GOSUB 65: NEXT N
45   N=173: X=N-163: GOSUB 65
50   FOR N=219 TO 223 STEP 2: X=(N-197)/2: GOSUB 65: NEXT N
55   ? "Loading CAT Command": LOAD 0,"CAT"
60   ? "Loading FIND Command": LOAD 0,"FIND"
65   F=KD+12*(X): MEM[F]=N: READ $C[0]: GOSUB 90
70   FOR G=1 TO 10: MEM[F+G]=ASC[$C[0;G]]: NEXT G: MEM[F+G]=0
75   $A=%13: P=POS[$A,$C[0]]: IF P THEN $C[0;P]="[Return]"
80   $A=%(N-128): ; "[GRAPH]-";$A;"      ";$C[0]
85   RETURN
90   P=POS["©",$C[0]]: IF P THEN $C[0;P]=%13%0
95   P=POS["▨",$C[0]]: IF P THEN $C[0;P]=%34
100   RETURN
105   DATA 0420H,A0+0CH,04DDH,0D778H,0460H,0492H,A0+010H,A0+030H
110   DATA KD,0,0,0,0,0,0,0,0,0,0,0,0,0,0
115   DATA 0C22DH,010H,0C040H,0202H,11,09631H,01304H,0A042H
120   DATA 0D451H,016FBH,0380H,0DE31H,0D451H,01601H,0380H,0DE31H
125   DATA 05A0H,0EDA8H,010F9H,0DE03H,05A0H,0EDA8H,0380H,0
130   DATA "RUN©","LIST","LOAD 0,▨","LDIR▨","DATA ","GOSUB","RETURN"
135   DATA "SAVE 0,▨","▨,REP,EX©","CONT©","COLOUR","GOTO","GRAPH","TEXT"
```

Our second program from Mr.Eaves is called MEMDUMP and will create a BASIC m/code loader. The user enters the start and finish addresses of the code/data and the program creates a BASIC program to reproduce the memory pattern. With a bit of ingenuity the program can be used to relocate m/code programs.

The first data line contains the start and finish addresses and the subsequent lines contain the memory image. PURGE the original program lines before storing the m/code loader.

```
 20   DIM $LIN[10],$IP[10]
 25   COLOUR 1: TEXT : ? "<0C>"
 30   INPUT "Start Address",$A: GOSUB 140: A1=A2
 35   INPUT "Finish Address",$A: GOSUB 140
 40   ? : INPUT "BASIC Start Line",B1
 45   IF B1<200 THEN ? "THIS WILL OVERWRITE MEMDUMP....RE-ENTER": GOTO 40
 50   INPUT "BASIC Step Length",B2
 55   ? : ? "Reading Memory & Creating Program Lines": ?
 60   $B=B1: $LIN[0]=$B+"READ C,C1:FOR I=C TO C1 STEP 2"
 65   ? $LIN[0]: ENTER $LIN[0]
 70   B1=B1+B2: $B=B1: $LIN[0]=$B+"READ C:MWD[I]=C:NEXT I"
 75   ? $LIN[0]: ENTER $LIN[0]
A80   B1=B1+B2: $B=B1: $A=A1: $LIN[0]=$B+"DATA "+$A: $A=A2: $LIN[0]=$LIN[0]+",
 85   ? $LIN[0]: ENTER $LIN[0]
 90   B1=B1+B2
 95   Z=0: $IP[0]="": FOR I=A1 TO A2 STEP 2
 100    A=MWD[I]: $A=A: $IP[0]=$IP[0]+$A+","
 105    Z=Z+1: IF MOD[Z,5]=0 THEN GOSUB 125
 110   NEXT I
 115   IF MOD[Z,5]<>0: GOSUB 125
 120   END
 125   L=LEN[$IP[0]]: $B=B1: $LIN[0]=$B+"DATA "+$IP[0],L-1
 130   ENTER $LIN[0]
 135   ; $LIN[0]: $IP[0]="": B1=B1+B2: RETURN
 140   IF  NOT POS["H",$A] THEN $A=$A+"H"
 145   A2=$A,B: RETURN
 2000   READ C,C1: FOR I=C TO C1 STEP 2
 2010    READ C: MWD[I]=C: NEXT I
 2020   DATA 24576,24608
 2030   DATA 1440,-4696,4345,-8701,1440
 2040   DATA -4696,896,0,512,-5367
 2050   DATA 513,24726,-9104,5886,4000
 2060   DATA 21871,1217
```

These lines were created by t
above code and are only sho
for example.

The next program was written by Tim Gray and allows the use of expansion memory on the E.bus as a RAM DISK. The program relies on the fact that all disk access is made via the routine that starts at 6180h. A patch is put into the main disk access routine and when an access is made the RAMDISK program checks if the drive number is 3. If it is then RAM is used as a disk, otherwise normal disk access is made.

To use the program, LOAD the RAMDISK code having first set XMEM to the correct address for the start of your external RAM, then change memory word 6182h to the entry point of RAMDISK. From then on drive 3 will be RAM.

There must be enough RAM to take the disk capacity, ie.86k fot 40T SS SD, and drive 3 parameters have to be set correctly using CONFIG. DI and FORMAT don'ty y use the access routine at 6180h so they can't be used. This makes it difficult to clear the directory of the RAMDISK so it is better if drive 3 is set to the same configuration as another drive then DISKCOPY can be used to transfer the whole disk contents to RAM.

```
;RAMDISC BY TIM GRAY
; SET MWD 6182 TO ENTRY ADDR TO USE
; DRIVE 3 BECOMES RAM
                                ORG    >FDD0
                        ENTRY   EQU    >FDD0
                        XMEM    EQU    >2000                  ;START OF EXT RAM
FDD0 D32D 0002          ENTRY:  MOVB   @>0002(R13),R12        ;GET DRIVE
FDD4 098C                       SRL    R12,8                  ;MOVE TO LOW BYTE
FDD6 028C 0003                  CI     R12,>0003              ;CHECK IF DRIVE 3
FDDA 1302                       JEQ    START                  ;YES
FDDC 0460 61A4                  B      @>61A4                 ;NO BRANCH BACK
FDE0 C32D 0002          START:  MOV    @>0002(R13),R12        ;DRIVE+ADDR1
FDE4 0A8C                       SLA    R12,8                  ;ISOLATE ADDR 1
FDE6 C2ED 0004                  MOV    @>0004(R13),R11        ;GET ADDR 2
FDEA C28B                       MOV    R11,R10                ;MAKE A COPY
FDEC 024B 0FFF                  ANDI   R11,>0FFF              ;ISOLATE LOWER 4K
FDF0 098A                       SRL    R10,8                  ;
FDF2 D28C                       MOVB   R12,R10                ;CALCULATE PAGED 4K
FDF4 0A4A                       SLA    R10,4                  ;
FDF6 022B 2000                  AI     R11,>2000              ;OFFSET TO PAGED 4K
FDFA 022A 2000                  AI     R10,XMEM               ;START OF EXT RAM
FDFE D80A F104                  MOVB   R10,@>F104             ;SETUP MAPPER
FE02 03A0                       CKON                          ;MAPPER ON
FE04 C0ED 0006                  MOV    @>0006(R13),R3         ;R/W BUFFER
FE08 C12D 0008                  MOV    @>0008(R13),R4         ;NO OF BYTES
FE0C D06D 0001                  MOVB   @>0001(R13),R1         ;R/W FLAG
FE10 1603                       JNE    WRITE
FE12 0209 DCFB          READ:   LI     R9,>DCFB               ;MOVB *R11+,*R3+
FE16 1002                       JMP    EXEC                   ;JUMP EXECUTE
FE18 0209 DEF3          WRITE:  LI     R9,>DEF3               ;MOVB *R3+,*R11+
FE1C 0489               EXEC:   X      R9                     ;MOVE THE DATA
FE1E 0604                       DEC    R4                     ;CHECK TRANSFER END
FE20 130A                       JEQ    RET1                   ;JUMP RET1
FE22 028B 2FFF                  CI     R11,>2FFF              ;END OF 4K BLOCK ?
FE26 12FA                       JLE    EXEC                   ;NO BACK FOR MORE
FE28 020B 2000                  LI     R11,>2000              ;RESET POINTER
FE2C 022A 0100                  AI     R10,>0100              ;INC MAPPAR
FE30 D80A F104                  MOVB   R10,@>F104;            "
FE34 10F3                       JMP    EXEC                   ;BACK FOR MORE
FE36 03C0               RET1:   CKOF                          ;MAPPER OFF
FE38 020A 0200                  LI     R10,>0200              ;RESTORE MAPPER
FE3C D80A F104                  MOVB   R10,@>F104             ;     "
FE40 04C0                       CLR    R0                     ;CLEAR ERROR CODE
FE42 D740                       MOVB   R0,*R13                ;     "
FE44 0380                       RTWP                          ;RETURN
```

```
ENTRY   FDD0            XMEM    2000            ENTRY   FDD0            START   FDE0
READ    FE12            WRITE   FE18            EXEC    FE1C            RET1    FE36
```

CDOS File Description Utility By Rm.LEE.

This CDOS utility program prompts for a drive number and file name, it then produces a full file description of the named file, based on the information found in the disc directory. This includes File type (BASIC or M/C program or data file), File length or format, Record size, Load and autorun address for M/C, and file fragmentation information (Where the file is stored on disc). Also space allocated and space usage information is given (The allocated space can be larger than the used space, when a file has been REPlaced with a smaller file).

```
 100  DIM $F[1]
 110  PRINT "<0C>CDOS File Description Utility 1.0 1986"
 120  PRINT
 130  INPUT "Drive "%1;D
 140  INPUT "Filename "£8;$F[0]
 150  OPEN D,$F[0],F1
 160  DE=F1+32  !DIRECTORY ENTRY
 170  PRINT
 180  IF MWD[DE+10]<>0 AND (MWD[DE]=05A5AH OR MWD[DE]=0A5A5H): P
RINT "BASIC Program": GOTO 240
 190    ELSE IF MWD[DE]=05A5AH OR MWD[DE]=0A5A5H: PRINT "M/C Pro
gram": GOTO 240
 200  IF MWD[DE]=0FFFFH: PRINT "Sequential Data File"
 210    ELSE PRINT "Random Access file": PRINT MWD[DE];" Byte Re
cord Size": ? MWD[DE+18]/MWD[DE];" Records"
 220  PRINT "File Length";MWD[DE+18];" Bytes"
 230  GOTO 270
 240  PRINT "Program Length";MWD[DE+16]" Bytes"
 250  IF MWD[DE]=0A5A5H: PRINT "Auto-run"
 260  IF MWD[DE+10]=0: GOSUB 400
 270  PRINT
 280  BT=0
 290  PRINT "Sector No.      No.. Blocks"
 300  FOR N=0 TO 7
 310   IF MWD[DE+32+4*N]=0: GOTO 350
 320   PRINT MWD[DE+32+4*N],,MWD[DE+34+4*N]
 330   BT=BT+MWD[DE+34+4*N]
 340  NEXT N
 350  PRINT
 360  IF MWD[DE]=0A5A5H OR MWD[DE]=05A5AH: PRINT INT[MWD[DE+16]/
MWD[06362H+D*2]+127/128];
 370    ELSE PRINT INT[MWD[DE+18]/MWD[06362H+D*2]+127/128];
 380  PRINT " Blocks Used Out of";BT;" Allocated"
 390  END
 400  PRINT "Load Address ";£,MWD[DE+12]"H"
 410  IF MWD[DE]=0A5A5H: PRINT "Auto-run Address ";£,MWD[DE+14]"
H"
 420  RETURN
```

HARDWARE MODIFICATIONS.

A number of hardware modification ideas have been sent in and we hope that they are of interest to some users. We would, however, add a word of caution about such mods, in that much damage can be caused even while making minor changes to wiring and PCBs. Unless you are 100% certain of what you are doing we would not recommend that you try any of the ideas that publish. Having said that, we are sure that the originators of these suggestions have taken great care in their designs.

Prem Holdaway sent us in a description of the changes which he has made to his system. He add thicker(approx .7mm) wires from the power supply board to the main board, and also added separate power wires to IC48 to improve stability. Prem also added the circuit from issue 4 to improve the display. With some careful adjustments and setting up this proved to be succesful.

Prem also suggests bringing the size and density jumpers out to swithes on the front panel.

John Mackenzie suggests the following mods to improve disk reliability. They have been tried by John, and he also points out that his is a first edition board, and he has replaced the RP2 4K7 bank of resistors with individual components.

HARDWARE:

MODS TO DISK CONTROLLER INTERFACE

The IC and component numbers are as per the original PCB and not the ETI numbers.

1. Cut track to IC 13 pin 3 (on top of board)
2. Cut track to IC 13 pin 6 (on top of board)
3. Link IC 13 pins 2 & 3
4. Link IC 13 pins 6 & 11
5. Link IC 13 pin 10 to IC 12 pin 11
6. Link IC 13 pin 9 to IC 5 pin 5
7. Link IC 27 pin 6 to IC 16 pin 13
8. Link pin 14 - 8" drive socket to pin 32 - 5" drive socket
9. Link pin 18 - 8" drive socket to pin 2 - 5" drive socket
10. Change R68 from 4K7 to 2K7
11. Change R69 from 10K to 5K6 (3K9 or 3K6 which ever works best)
12. Change R70 from 18K to 12K
13. Change C4 from 330p to 150p

SHORT TIPS

Prem Holdaway has the following tip for anybody experiencing problems with intermittent disk drives. His drive would not read or write, the LED began flickering and then gave up all together. The problem turned out to be the disk select switching IC (IC 85a 74LS139), so he recommends checking this if you have similar problems.

Robert Lee sent the following item leading on from the article by C.M.Gale in issue six, on the CDOS directory system.
Each directory entry is 64 bytes long, each word and its functions being listed below. A directory entry can be accessed from BASIC by OPENing any file, the 64 byte entry can then be indexed by adding 32 to the file variable, this memory location being the first word of the directory entry, as used in the File Description Utility.

Directory Entry Format

Byte	Function

0-1	Auto-run flag 5A5AH=No auto-run.
	A5A5H=Auto-run.
	FFFFH=Sequential Data.
	Any other number is Record size for random acess file.

2-9	8 Byte Name of File.
10-11	Zero for M/C program. Otherwise a BASIC pointer, similar to cassette header block.
12-13	Load address for M/C program. Otherwise a BASIC pointer.
14-15	Auto-run address for M/C program. otherwise a BASIC pointer.
16-17	Number of bytes in BASIC or M/C program.
18-19	EOF address for relative data or Sequential data file.
20-31	No apparent use! Could be used for time and date stamping of files.

Fragmentation list.

32-33	Sector number.	Segment 1
34-35	Number of Sectors.	
36-37	Sector number.	Segment 2
38-39	Number of Sectors.	
40-41	Sector number.	Segment 3
42-43	Number of Sectors.	
44-45	Sector Number.	Segment 4
46-47	Number of Sectors.	
48-49	Sector Number.	Segment 5
50-51	Number of Sectors.	
52-53	Sector Number.	Segment 6
54-55	Number of Sectors.	
56-57	Sector Number.	Segment 7
58-59	Number of Sectors.	
60-61	Sector Number.	Segment 8
62-63	Number of Sectors.	

CORTEX USER GROUP FEATURE   TIM GRAY 19.03.86

Here is an extra statement that allows you to list the directory of a disc without having to load the basic programme "LDIR" which would overwrite your current programme .

The code is loaded into high memmory assuming you have done the mod to be able to use it .
Add the statement name and start addr to the tables :-

```
MWD[3A92H]=9248H
MWD[4030H]=0FEE0H
```

The statement is used in the form DIR 1 for a list of the files on drive 1 and can be used from within a programme.

DIRECTORY DETAILS :-

```
1st word file type
next 8 bytes name
next 5 words pointers in basic :-
       word 6   : offset to statement location table
       word 7   : offset to variable definition table
       word 8   : next variable pointer
       word 9  : next variable definition pointer
       word 10 : load addr

  or in machine code
       word 6  : allways zero
       word 7  : load addr
       word 8  : run addr
       word 9  : length
       word 10 : load addr
```

From word 16 to 31 is a disc alocation map for the file with the first word in each entry giving the track and sector number and the second word the number of sectors used from this start point . A total of 8 entries is possible for a segmented file .

DIR PRINT EXAMPLE :-

```
DIR 0
 LDIR    .AB  DI      .AB  FORMAT  .AC
 SYSTEM$ .AC  DELETE  .AB  RENAME  .AB
 CONFIG  .AB  AUTOEXEC.AB  FILECOPY.AB
 DISKCOPY.AB  CODE 1  . C  COPYFILE.AB
 CDOS1.20.AB  RAMDISC . C  DIR     . C
```

```
      DIR STATEMENT

START      FEE0 0203 LI   R3,>FE50        : DATA BUFFER (40 BYTES)
           FEE4 C803 MOV  R3,@>FFEA       : DATA BUFFER POINTER
           FEE8 2EC1 XOP  R1,11           : GET DRIVE NUMBER
           FEEA 0203 LI   R3,>FE80        : DIRECTORY BUFFER (64 BYTES)
           FEEE 0281 CI   R1,>0003        : MAXIMUM DRIVE NUMBER ?
           FEF2 1202 JLE  >FEF8           : NO
           FEF4 2FA0 XOP  @>002E,14       : YES ERROR "Invalid device number"
           FEF8 0A11 SLA  R1,1            : DRIVE NUMBER MULTIPLIED BY 2
           FEFA C161 MOV  @>6362(R1),R5   : SECTOR SIZE
           FEFE C1A1 MOV  @>6382(R1),R6   : CONFIG DATA ADDR FOR THIS DRIVE
           FF02 C226 MOV  @>0006(R6),R8   : NUMBER OF ENTRIES POSSIBLE
           FF06 C266 MOV  @>0004(R6),R9   : DIRECTORY START SECTOR
           FF0A 3A45 MPY  R5,R9           : CALCULATE DIRECTORY START ADDR
           FF0C C24A MOV  R10,R9          : AND MOVE IT INTO R9
           FF0E 0A71 SLA  R1,7            : DRIVE NUMBER TO HIGH BYTE
           FF10 0204 LI   R4,>0040        : 64 BYTES PER ENTRY TO TRANSFER
           FF14 0207 LI   R7,>0000        : START DIRECTORY ENTRY NUMBER
GET ENTRY  FF18 C287 MOV  R7,R10          : THIS DIRECTORY NUMBER TO R10
           FF1A 3A84 MPY  R4,R10          : CALCULATE THIS DIRECTORY ADDR
           FF1C A2C9 A    R9,R11          :
           FF1E C08B MOV  R11,R2          : AND MOVE IT TO R2
           FF20 0420 BL   @>FFEC          : READ DIRECTORY ENTRY TO BUFFER
           FF24 D000 MOVB R0,R0           : CHECK FOR ERROR
           FF26 1302 JEQ  >FF2C           : NO
           FF28 0460 B    @>6550          : YES BRANCH TO PRINT ERROR ROUTINE
           FF2C 0420 BLWP @>FF38          : BRANCH TO PRINT FORMAT SUBROUTINE
           FF30 0587 INC  R7              : INCREMENT TO NEXT ENTRY
           FF32 8207 C    R7,R8           : CHECK FOR MAXIMUM ENTRY NUMBER
           FF34 1AF1 JL   >FF18           : NO , GET NEXT ENTRY
           FF36 1044 JMP  >FFC0           : YES , RETURN
PRINT      FF38 FEC0                      : WORKSPACE POINTER (32 BYTES)
FORMAT     FF3A FF3C                      : PROGRAMME COUNTER
           FF3C C06D MOV  @>0006(R13),R1  : DIRECTORY BUFFER TO R1
           FF40 C031 MOV  *R1+,R0         : CHECK IF FILE EXISTS
           FF42 1601 JNE  >FF46           : YES
           FF44 0380 RTWP                 : NO , RETURN

           FF46 0202 LI   R2,>0008        : 8 BYTES PER NAME
           FF4A C0E0 MOV  @>FFEA,R3       : CURRENT DATA BUFFER ADDR
           FF4E D131 MOVB *R1+,R4         : MOVE 8 BYTE NAME TO BUFFER
           FF50 1602 JNE  >FF56           : AND FILL WITH SPACES
           FF52 0204 LI   R4,>2000
           FF56 DCC4 MOVB R4,*R3+
           FF58 0602 DEC  R2
           FF5A 16F9 JNE  >FF4E
           FF5C 0204 LI   R4,>2E20        : ASCII DOT , SPACE
           FF60 DCC4 MOVB R4,*R3+         : SEND DOT TO BUFFER
          FF62 06C4 SWPB R4               : MOVE SPACE TO HIGH BYTE
           FF64 0280 CI   R0,>A5A5        : CHECK FOR AUTO RUN PROGRAMME
           FF68 1603 JNE  >FF70           : NO , NOT AUTO
           FF6A 0205 LI   R5,>4100        : YES , ACII "A" TO R5
           FF6E 1005 JMP  >FF7A
NOT AUTO   FF70 0280 CI   R0,>5A5A        : CHECK FOR PROGRAMME
           FF74 160A JNE  >FF8A           : NO , NOT PROG
           FF76 0205 LI   R5,>2000        : ASCII SPACE
          FF7A C1B1 MOV  *R1+,R6        : CHECK FOR BASIC
          FF7C 1303 JEQ  >FF84          : NO , NOT BASIC
          FF7E 0225 AI   R5,>0042       : ADD ASCII "B"
          FF82 1002 JMP  >FF88
NOT BASIC FF84 0225 AI   R5,>0043       : ADD ASCII "C"
          FF88 1008 JMP  >FF9A
NOT PROG  FF8A 0280 CI   R0,>FFFF       : CHECK FOR SEQUENTIAL DATA
          FF8E 1603 JNE  >FF96          : NO , NOT SEQ
          FF90 0205 LI   R5,>5344       : ASCII "SD"
          FF94 1002 JMP  >FF9A
NOT SEQ   FF96 0205 LI   R5,>5244       : ASCII "RD"
SEND TYPE FF9A DCC5 MOVB R5,*R3+        : SEND FILE TYPE TO BUFFER
          FF9C 06C5 SWPB R5
          FF9E DCC5 MOVB R5,*R3+
          FFA0 DCC4 MOVB R4,*R3+        : PLUS 2 SPACES
          FFA2 DCC4 MOVB R4,*R3+
          FFA4 0283 CI   R3,>FE70       : CHECK FOR FOR BUFFER FULL
          FFA8 1A08 JL   >FFBA          : NO , NOT FULL
          FFAA 04E3 CLR  @>FFFF(R3)     : SEND NULL BYTE TO BUFFER
          FFAE 0002                     : MID OPCODE FOR PRINT CR,LF
          FFB0 0F04 WRIT R4             : WRITE A SPACE
          FFB2 0FA0 MSG  @>FE50         : PRINT THE BUFFER
          FFB6 0203 LI   R3,>FE50       : RESET THE BUFFER POINTER
NOT FULL  FFBA C803 MOV  R3,@>FFEA      : STORE BUFFER POINTER
          FFBE 0380 RTWP                : RETURN
END OF    FFC0 C0E0 MOV  @>FFEA,R3      : CHECK IF BUFFER EMPTY
DIRECTORY FFC4 0283 CI   R3,>FE50       :
          FFC8 130C JEQ  >FFE2          : YES , BUFFER EMPTY
          FFCA 04E3 CLR  @>FFFF(R3)     : SEND NULL BYTE
          FFCE 0204 LI   R4,>2000       : LOAD ASCII SPACE
          FFD2 0002                     : WRITE  CR,LF
          FFD4 0F04 WRIT R4             : WRITE SPACE
          FFD6 0FA0 MSG  @>FE50         : PRINT THE BUFFER
          FFDA 0203 LI   R3,>FE50       : RESET BUFFER POINTER
          FFDE C803 MOV  R3,@>FFEA      : STORE BUFFER POINTER
BUF'EMPT  FFE2 0460 B    @>3F30         : BRANCH BACK TO BASIC
          FFE6 DAI  DATA
          FFE8      ::
READ      FFEC      CLR  RØ             : CLEAR RØ TO FORCE
                    BLWP @>6180         : DISK READ
                    RT
```

Note the original code used to BLWP @6180 direct from FF20 the extra code at FFEC is to ensure that RØ is clear and that the disk access is a read and not a write

POINTS TO NOTE from previous newsletters.

John Mackenzie has sent in one or two corrections to points made in previous issues.

1) In issue 3 page 11, first paragraph last line, add .56 to list of lines to change.

2) In issue 7 page 7, line 704 should read;

```
704 IF $Q="Y" THEN GOTO 100
```

3) In Issue 6 page 14, sub paragraph 3, we ommited the listing mentioned, and so include it here..

```
30 TEXT : COLOUR 1,15
40 ? :? ;"   Auto file load from disc 0": ?
110 DIM B(100),$N(2),X(20),$PGM(30,2),$DOS(14,2)
120 AX=ADR(X(0)): AB=ADR(B(0))
130 DATA 0420H,06180H,0D000H,01601h
140 DATA 0380h,0460h,06550h,04f2h
150 DATA 04d2h,0c0f1h,0704h,0a13h
160 DATA 01701h,0592h,0600h,01601h
170 DATA 0380h,0a14h,016f8h,010f5h
180 FOR I=AX TO AX+38 STEP 2
190  READ IAQ: MWD(I)=IAQ
200 NEXT I
201 READ XX
202 FOR I=0 TO XX
204  READ $DOS(I,0)
206 NEXT I
210 D=0
220 DC=MWD(06382h+D*2)
230 BS=MWD(DC): NB=MWD(DC+4)
240 DS=MWD(DC+4): ND=MWD(DC+6)
245 BPS=MWD(06362h+D*2)
300 CO=1
310 FOR E=0 TO ND-1
320  DA=DS*BPS+E*64
330  CALL AX,0,D*256,DA,AB,64
340  IF MWD(AB)=0 THEN GOTO 420
350  FOR II=1 TO 8
360   $N(0;II)=%MEM(AB+
370  NEXT II
380  FOR I=0 TO XX
390   IF $N(0)=$DOS(I,0) THEN GOTO 420
400  NEXT I
410  $PGM(CO,0)=$N(0)
411  IF CO>16 THEN AA=CO-14: ?@(20,AA);CO; TAB (6);$N(0): GOTO 415
412  ? TAB (2);CO; TAB (8);$N(0)
415  CO=CO+1
420 NEXT E
425 ?@(0,20);
430 ? TAB(10);"31"; TAB (18); "Disk 1"
440 ? TAB(10);"32"; TAB (18); "Disk 0": ?
460 INPUT "       Select afile number ";#2;S
470 IF S=31 THEN LOAD 1,"AUTO3"
475 IF S=32 THEN LOAD 0,"AUTO2"
480 LOAD 0,$PGM(S,0)
490 STOP
2000 REM * FILTER FILE *
2010 REM Increase No in DATA when files added. If more than 14 added increase $DOS
         line 110
2030 DATA 2,"AUTO2","AUTO3","SYSTEM$"
```

9.12 + 9.13

WORTEX
inc
SPELTEX
Version 2:1
Jan 87
[C] HALMAC Computing. September 1984

The January re-issue of Wortex is now available. The new system includes the spelling checker Speltex, and more additions to the main word processor program. The system menus are shown below.

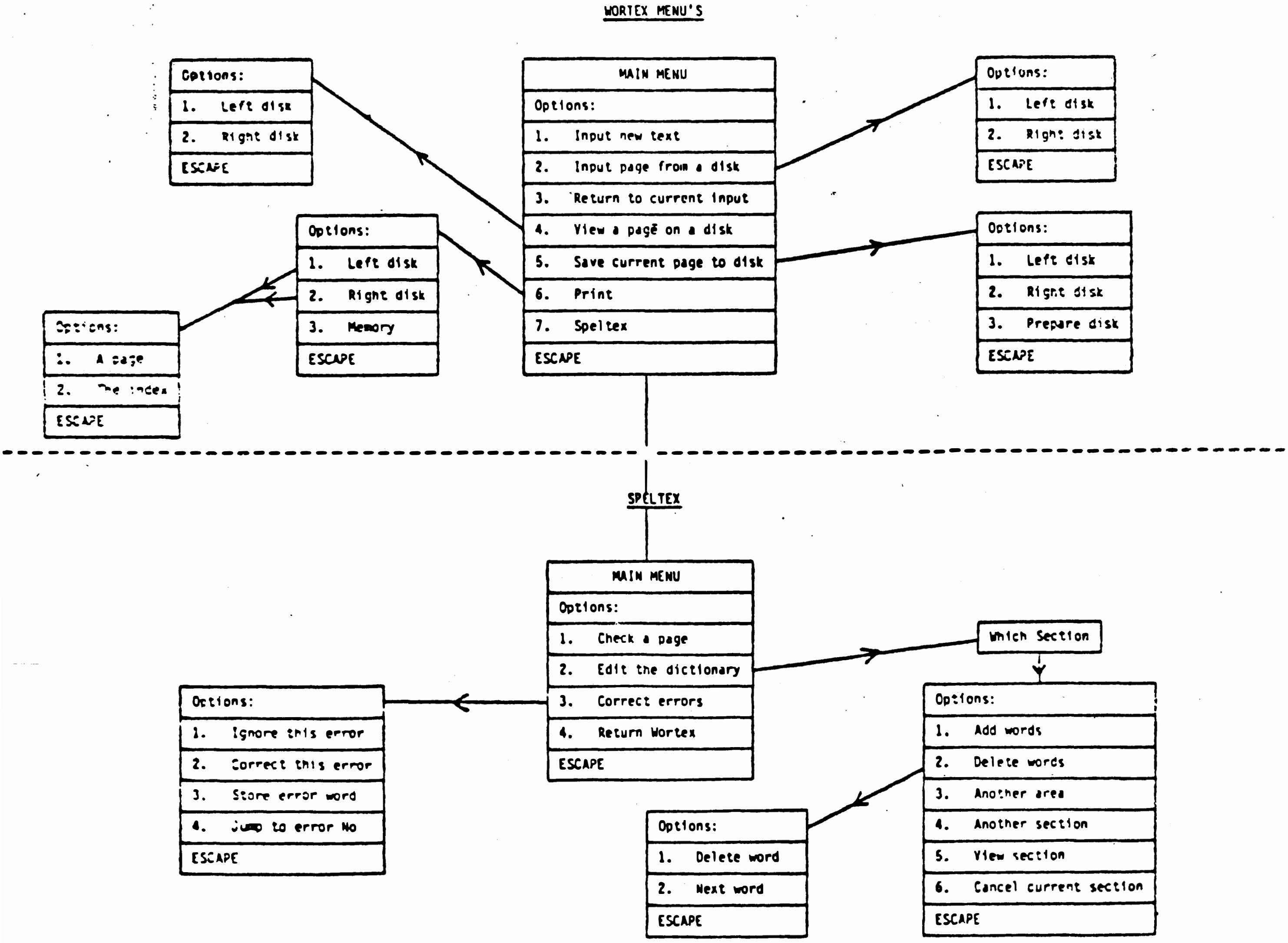
Users can get the re-issue Free by sending back the original Wortex Disk. Note you will have to inclued a disk for the Speltex dictionary if you do not have Speltex.

Non-users can get a copy by sending:

£15.00 plus two 5¼ DD disks to:

J S Mackenzie
4 Werstan Close
MALVERN
WR14 3NH

Querries call 06845-65619 evenings.



WORTEX MORE INFORMATION

This Word Processor for the Cortex runs under CDOS 1.20. The system uses two 40 track single sided drives. Drive 1 must be capable of double density operation for the dictionary of the Spelling checker.

FUNCTIONS

Full text input:

Character input.
Character replacement.
Character deletion.
Character Insertion.

Full page formating:

Automatic page numbering.
Automatic left justify.
Automatic word wrap.
Automatic/manual RETURN.
Centre text.
Right justify text.
Set Left margin.
Set Right margin.
Set Tab markers
Line delete.
Line clear.
Line insert.
Line copy.
Copy text from disk.
Page clear.

On screen monitor of the text:

40 Chars: Two lines on the screen.
80 Chars: One line, 40 chars on, 40 chars off.

Spelling checker:

Check page.
Edit dictionary.
Correct errors.

Hurry order now before January price rise!

We hope he means January 1988 ED!