* CORTEX USER GROUP *

NEWSLETTER ISSUE NO.4 DECEMBER 1985

Dear Cortex Owner,

Welcome to issue 4 of the Cortex User Group Newsletter. May we take this opportunity to wish you and your Cortex a Merry Christmas and a Happy New Year. As you can see there are a few changes inside. Our aim is principally to provide a useful communication service between users, and to do this does, of course, require your cooperation. Therefore we are prepared to listen to any suggestions, and will consider any items/articles sent in to us. We cannot guarantee to solve all of your problems, but in previous newsletters where problems were featured we are proud to claim a very high success rate.

$A 2 Several people have mentioned that in the past program listings have been typed with mistakes inevitably creeping in. We are now photocopying listings wherever possible. Please do not be put off sending in short programs/routines if you don't have a printer. Another feature which we are introducing so that you don't have to spend your time typing and debugging, is a tape containing all of the programs/routines featured in this newsletter.

Other new features are: A regular quarterly competition
New prices for existing software
4 new games for sale
Blank computer grade cassettes
Announcement of new hardware

We are still maintaining our offer to market your software! If you have written any good programs then we are waiting to hear from you. We pay £1.50 royalties for every copy sold (irrespective of selling price). All you have to do is send a fully working version on tape with any necessary documentation, and we wll do the rest.

Read on - we look forward to receiving your comments!

kph computaware

63 Highlands Road, Andover, Hants. SP10 2PZ

CORTEX USER GROUP NEWSLETTER 4

INDEX

Programs

Once again we have received many interesting programs from Cortex Users. Wherever possible we aim to retain the original listing in the hope of minimising errors. What better way to learn about programming than by example? Please note that any programs or routines that are sent in(no matter how short) are considered.

The first offering this time comes from Brian Harris of Plymouth,Devon.

The following addition to the CDOS LIST DIRECTORY utility. Together with Tim Grays auto run from BOOT, described in newsletter III. Will allow the directory to be listed and a program to be selected from the screen. A cursor is provided which can be moved down the program list using the SPACE BAR. Pressing the RETURN KEY moves the program name to the SYSTEM$ routine and then loads it. Using the DISK INSPECT utility, modify the SYSTEM$ file as follows:-

```
00    00 06 69 9A 02 01 69 C8
08    C0 B1 13 02 C4 B1 10 FC
10    06 A0 6C B0 04 60 69 A8
18    00 00 0D 0A 07 00 00 00
20    00 00 00 00 C0 88 10 02
28    02 02 69 B6 04 C1 02 0B
30    00 80 04 60 65 9C 4C 44    Enter the underlined code.
38    49 52 00 00 00 00 00 00
40    00 00 00 00 00 00 00 00
48    3A 8A 2C 1F 3B 36 00 1C
50    40 28 6A 00 3A 8C 7B 07
58    3B 38 01 66 40 2A 69 00
60    3A 8E A1 4E 40 2C 6A E0
68    3A 90 A5 60 40 2E 6A E4
70    40 56 69 A4 00 00 00 00
78    00 00 00 00 00 00 00 00
```

Add the following lines to the LIST DIRECTORY program.

```
570 SP=168
572 SPUT(SP),127
574 K=KEY[0]:IF K=0 THEN GOTO 574
576 IF K=32 THEN SPUT(SP),32:GOTO 582
578 IF K=13 THEN GOTO 588
580 GOTO 574
582 SP=SP+40:SGET(SP-8),CHR
584 IF CHR=127 THEN GOTO 570
586 GOTO 572
588 SP=SP-8:AD=069B6H
590 SGET(SP),CHR:IF CHR=32 THEN CHR=0
592 MEM[AD]=CHR
594 SP=SP+1:AD=AD+1:IF AD=069BEH THEN CALL 069A8H
596 GOTO 590
```

To display the directory without having to input the drive number, change line 210 to D=0.

Paul Sheridan, of Gloucestershire, has sent in a program for number base conversion which he wrote and finds very useful.

```
6000  DIM $A(10),B(20),BB(16),C(9),C1(9),C2(16),C3(9),C4(9),$AA(9)
6010  DIM $BN(8),$A1(5)
6020  ?@"C"; "NUMBER CONVERSION PROGRAM"
6030  ?@"BD";"WHICH TYPE TO BE CONVERTED": DBH=0 : BDH=0 : DAD=0: RF=0: XX=0
6040  ?@"D4R";"H=HEX": ?@"D4R";"B=BINARY": ?@"D4R";"D=DECIMAL"
6050  A=KEY(0) : IF A=72 : GOTO 6190
6060  IF A=68 : GOTO 6140
6070  IF A=66 : GOTO 6090
6080  GOTO 6050
6090  ?@"C5D";"TYPE OF OUTPUT:-1=HEX": ?@"D17R";"2=DECIMAL": ?@"D17R";"3=BOTH"
6100  A=KEY(0) : IF A=49 : GOTO 6640
6110  IF A=50 : GOTO 6520
6120  IF A=51 : BDH=1 : GOTO 6520
6130  GOTO 6100
6140  ?@"C5D";"TYPE OF OUTPUT:-1=BINARY": ?@"D17R";"2=HEX": ?@"D17R";"3=BOTH"
6150  A=KEY(0) : IF A=49 : GOTO 7000
6160  IF A=50 : GOTO 6720
6170  IF A=51 : DBH=1 : GOTO 7000
6180  GOTO 6150
6190  ?@"C5D";"TYPE OF OUTPUT:-1=BINARY": ?@"D17R";"2=DECIMAL": ?@"D17R";"3=BOTH"
6200  A=KEY(0) : IF A=49 : GOTO 6480
6210  IF A=50 : GOTO 6240
6220  IF A=51 : DAD=1 : GOTO 6240
6230  REM * HEX TO DECIMAL *
6240  ?"INPUT HEX NO.";: INPUT $A(0)
6250  J=LEN($A(0)) : C(0)=1 : AD=16
6260  FOR D=1 TO J : C(D)=AD : AD=AD*16 : NEXT D
6270  MMM=0 : FOR D=1 TO J : $B(D)=$A(0;D),1
6280  BB(D)=ASC($B(D)) : IF BB(D)<48 OR BB(D)>70 : MMM=1
6290  NEXT D
6300  IF MMM=1 : GOTO 6240
6310  FOR D=1 TO J : A=0 : FOR E=48 TO 57
6320  IF BB(D)=E : B(D)=A
6330  IF BB(D)>64 AND BB(D)<71 : GOSUB 6360
6340  A=A+1 : NEXT E
6350  NEXT D : GOTO 6380
6360  X=10 : FOR F=65 TO 70 : IF BB(D)=F : B(D)=X
6370  X=X+1 : NEXT F : RETURN
6380  T=0 : FOR D=0 TO J : S=B(D)*C(J-D) : T=T+S
6390  NEXT D
6400  IF DAD=1 : DN=T : GOTO 7010
6410  IF RF=1 : RETURN
6420  ?:?"DECIMAL EQUIVALENT=" T
6430  ?:?"AGAIN (Y/N)"
6440  A1=KEY(0) : IF A1=89 : GOTO 6020
6450  IF A1=78 : END
6460  GOTO 6440
6470  REM * HEX TO BINARY *
6480  RF=1 : GOSUB 6240
6490  DN=T : GOTO 7010
6510  REM * BINARY TO DECIMAL *
6520  INPUT"INPUT BINARY NO." $BN(0)
6530  C1(1)=1 : T=0 : J=LEN($BN(0))
6540  FOR D=2 TO J : C1(D)=C1(D-1)*2 :NEXT D
```

```
6550  FOR D=1 TO J : $B(D)=$BN(0;D),1
6560  BB(D)=$B(D),X : IF BB(D)=1 OR BB(D)=0 : GOTO 6580
6570  GOTO 6520
6580  NEXT D : FOR D=0 TO J : R=C1(J-D)*BB(D+1) : T=T+R
6590  NEXT D : IF BDH=1 : GOTO 6650
6600  IF XX=1 : RETURN
6610  ?:?"DECIMAL EQUIVALENT=" T
6620  GOTO 6430
6630  REM * BINARY TO HEX *
6640  XX=1 : GOSUB 6520
6650  AF=T
6660  GOSUB 6730
6670  ?@"2D4R";"HEX EQUIVALENT"; : IF $F2>"00" : ?$F2" "$F3
6680  ELSE ?$F3
6690  ?@"2D4R";"DECIMAL EQUIVALENT" T
6700  GOTO 6430
6710  REM * DECIMAL TO HEX *
6720  INPUT"INPUT DECIMAL NO." AF
6730  AB=FRA(AF) : IF AB>0 :GOTO 6720
6740  C3(1)=1 : C3(2)=16 : C3(3)=256 : C3(4)=4096 : C3(5)=65536
6750  DD=5 : AP=AF : FOR D=1 TO 5 : XA=1
6760  AP=AP-C3(DD) :IF AP<0 : AP=AP+C3(DD) : $A1(0;D)=/"0" : GOTO 6830
6770  AP=AP+C3(DD)
6780  AP=AP-C3(DD) : IF AP<0 :AP=AP+C3(DD) : GOTO 6810
6790  IF AP=0 : XA=XA+1 : GOTO 6810
6800  XA=XA+1 : GOTO 6780
6810  XA=XA-1 : IF XA>9 : GOTO 6850
6820  $A1(0;D)=XA
6830  DD=DD-1 : NEXT D
6840  GOTO 6920
6850  IF XA=10 : $A1(0;D)=/"A"
6860  IF XA=11 : $A1(0;D)=/"B"
6870  IF XA=12 : $A1(0;D)=/"C"
6880  IF XA=13 : $A1(0;D)=/"D"
6890  IF XA=14 : $A1(0;D)=/"E"
6900  IF XA=15 : $A1(0;D)=/"F"
6910  GOTO 6830
6920  $F2=$A1(0;2),2 : $F3=$A1(0;4),2
6930  IF DBH=1 : GOTO 7180
6940  IF DAD=1 : RETURN
6950  IF BDH=1 : GOTO 6670
6960  ?:?"HEX EQUIVALENT";: IF $F2>"00" : ?$F2" "$F3
6970  ELSE ?$F3
6980  GOTO 6430
6990  REM * DECIMAL TO BINARY
7000  INPUT"INPUT DECIMAL NO." DN
7010  C2(1)=1 : DD=16
7020  AB=FRA(DN) : IF AB>0 : GOTO 7000
7030  FOR D=2 TO 16 : C2(D)=C2(D-1)*2
7040  NEXT D : AT=DN : FOR D=1 TO 16
7050  AT=AT-C2(DD) : IF AT<0 : AT=AT+C2(DD) : $AA(0;D)=/"0" :GOTO 7070
7060  IF AT>=0 : $AA(0;D)=/"1"
7070  DD =DD-1 : NEXT D
7080  $F=$AA(0;9),4 : $F1=$AA(0;13),4
7090  $FF=$AA(0),4 : $F6=
7100  IF DBH=1 : AF=DN : GOTO 6730
7110  ?"BINARY EQUIVALENT";
```

```
7120  IF $FF="0000" : IF $F6="0000" : GOTO 7140
7130  ?$FF" "$F6" "$F" "$F1
7140  IF $F="0000" : ?$F1
7150  ELSE ?$F" "$F1
7160  IF DAD=1 : ?: ?"DECIMAL EQUIVALENT=" T
7170  GOTO 6430
7180  ? : ?"HEX EQUIVALENT=";: IF $F2>"00" : ?$F2" "$F3
7190  ELSE ?$F3
7200  GOTO 7110
```

Tim Gray has once again sent in reams of information and programs(Thank you Tim). Here is a short routine to access external memory:-

```
  CALL 549EH,in/out,addr 1,addr 2,data/addr 3

   R0  =  IN=0            READ
   R1  =  ADDR 1          MAPPER VALUE (4K BLOCK No)
   R2  =  ADDR 2          XMEM ADDR 0000H TO 0FFFH
   R3  =  DATA            OR RECIVING ADDR
   R12 =                  NUMBER OF PARAMETERS

 549E D120 MOVB @>F104,R4        ! save mapper contents
 54A2 06C1 SWPB R1               ! mapper value to high byte
 54A4 0281 CI   R1,>1000         ! minimum mapper value
 54A8 1A12 JL   >54CE            ! not external addr
 54AA D801 MOVB R1,@>F104        ! set up mapper
 54AE 0282 CI   R2,>0FFF         ! max addr value = 4k
 54B2 1B0D JH   >54CE            ! out of 4K range
 54B4 0222 AI   R2,>2000         ! add offset
 54B8 03A0 CKON                  ! switch mapper on
 54BA C000 MOV  R0,R0            ! check for read/write
 54BC 1303 JEQ  >54C4            ! goto read
 54BE 06C3 SWPB R3               ! data to high byte
 54C0 D483 MOVB R3,*R2           ! write the data
 54C2 1001 JMP  >54C6            ! end
 54C4 D4D2 MOVB *R2,*R3          ! read the data
 54C6 03C0 CKOF                  ! switch mapper off
 54C8 D804 MOVB R4,@>F104        ! restor mapper contents
 54CC 0380 RTWP                  ! return
 54CE 2FA0 XOP  @>0031,14        ! error "illegal addr"
 54D2 0380 RTWP
```

The next program is a good example of how good the graphics capabilities of the Cortex are,(even from Basic). It was sent from Mr A.Lyall of Edinburgh, and was until recently one of our range of programs for sale. It is menu driven, allowing you to choose one of six predefined graphs, or enter your own function.

```
100 DIM LIN(10),FUN(9)
110 REM 3DGRAPH BY A.LYALL
260 TEXT
270 ? "<OC>"
280 ? @(9,2);"FUNCTION MENU"
290 ? @(9,3);"============="
300 ? @(3,6);"1..1/(COS(X/2)*COS(Y/2)+1.1)+2"
310 ? @(3,8);"2..1/(COS(X)*SIN(Y)+1.1)"
320 ? @(3,10);"3..1.5/(COS(X)*SIN(Y/2)+1.1)"
330 ? @(3,12);"4..1.5/(COS(X)*SIN(Y/3)+1.1)"
340 ? @(3,14);"5..1/(COS(X)*COS(Y)+1.1)"
350 ? @(3,16);"6..(SIN(X/3)*2)^3+(SIN(Y/3)*2)^3"
360 ? @(3,18);"7..ENTER YOUR OWN FUNCTION"
370 ? @(2,22);"PRESS NUMBER REQUIRED"
380 A=KEY(0): IF A=0 THEN GOTO 380
390   ELSE IF A=49 THEN DEF FNA=1/(COS(X/2)*COS(Y/2)+1.1)+2
400   ELSE IF A=50 THEN DEF FNA=1/(COS(X)*SIN(Y)+1.1)
410   ELSE IF A=51 THEN DEF FNA=1.5/(COS(X)*SIN(Y/2)+1.1)
420   ELSE IF A=52 THEN DEF FNA=1.5/(COS(X)*SIN(Y/3)+1.1)
430   ELSE IF A=53 THEN DEF FNA=1/(COS(X)*COS(Y)+1.1)
440   ELSE IF A=54 THEN DEF FNA=(SIN(X/3)*2)^3+(SIN(Y/3)*2)^3
450   ELSE IF A=55 THEN GOSUB 710
460   ELSE GOTO 260
470 REM * 3DGRAPH *
480 GRAPH
490 ? "<OC>"
500 FOR Y=0.25 TO 8.75 STEP 0.5
510 K=1
520 FOR X=0.25 TO 8.75 STEP 0.5
530 IF K=1 THEN PLOT -13*(X+Y),99-(Y-X+FNA )*5
540    ELSE PLOT TO -13*(X+Y),99-(Y-X+FNA )*5
550 K=0
560 NEXT X
570 PLOT TO -(13*(Y+X)),99-(Y-X+2)*5
580 NEXT Y
590 FOR X=0.25 TO 8.75 STEP 0.5
600 C=1
610 FOR Y=0.25 TO 8.75 STEP 0.5
620 IF C=1 THEN PLOT -13*(X+Y),99-(Y-X+FNA )*5
630    ELSE PLOT TO -(13*(Y+X)),99-(Y-X+FNA )*5
640 C=0
650 NEXT Y
660 PLOT TO -(13*(X+Y)),99-(Y-X+2)*5
670 NEXT X
680 ? @(2,22);"PRESS ANY KEY FOR MENU"
690 A=KEY(0): IF A=0 THEN GOTO 690
700   ELSE GOTO 260
710 REM *INPUT OWN FUNCTION*
720 ? "<OC>"
730 ? @(11,4);"INPUT YOUR OWN FUNCTION"
740 ? @(11,5);"======================="
750 ?:?:?
760 ? "PLEASE TYPE IN YOUR FUNCTION"
770 ? "EG  1/(COS(X)*SIN(Y)+1.1)"
780 INPUT $FUN(0)
790 $LIN(0)="810 DEF FNA= " + $FUN(0)
800 ENTER $LIN(0)
810 DEF FNA=
820 GOTO 470
```

Finally in this section another program from Tim Gray:-

The second program is a version of RM LEE's epsom printer dump routine. It dumps the graphics screen to the printer using a 3*3 dot matrix for each pixel, to represent the density of the pixel's colour as adifferent grey level. It uses no extra space in memory and is written position independent so that it can be relocated without problems. To use it just include the call routine after the graphics screen has been set up. A full A4 size picture is produced.

CALL "PAINT",(start address(05E00H))

```
5E00 020A LI   R10,>045B            : load R10 with RT opcode
5E04 068A BL   R10                  : branch to R10
5E06 C28B MOV  R11,R10              : R11 = the addr of this instruction
5E08 022B AI   R11,>004A            : offset to data storage
5E0C CA8B MOV  R11,@>00E2(R10)      : setup data return address
5E10 CA8B MOV  R11,@>0110(R10)      :
5E14 101F JMP  >5E54                : jump to main routine
5E16 00E0 DATA >00E0                : colour density data 3 bytes for each
5E18 E0E0 DATA >E0E0                 : colour code
5E1A E0E0 DATA >E0E0
5E1C E0A0 DATA >E0A0
5E1E 40A0 DATA >40A0
5E20 A000 DATA >A000
5E22 A0E0 DATA >A0E0
5E24 A0E0 DATA >A0E0
5E26 A040 DATA >A040
5E28 A0E0 DATA >A0E0
5E2A 40E0 DATA >40E0
5E2C 00E0 DATA >00E0
5E2E 00A0 DATA >00A0
5E30 40A0 DATA >40A0
5E32 A000 DATA >A000
5E34 A000 DATA >A000
5E36 E000 DATA >E000
5E38 0040 DATA >0040
5E3A 00E0 DATA >00E0
5E3C 40E0 DATA >40E0
5E3E A040 DATA >A040
5E40 A000 DATA >A000
5E42 4000 DATA >4000
5E44 0000 DATA >0000
5E46 0000 DATA >0000
5E48 0000 DATA >0000
5E4A 0000 DATA >0000
5E4C 0000 DATA >0000
5E4E 0000 DATA >0000
5E50 0000 DATA >0000
5E52 0000 DATA >0000
5E54 0201 LI   R1,>00BF             : maximum Y value
5E58 CA81 MOV  R1,@>0046(R10)       : set Y
5E5C 1000 NOP
5E5E C060 MOV  @>0026,R1            : check for graph mode
5E62 1602 JNE  >5E68                : yes o/k
5E64 2FA0 XOP  @>0030,14            : no error "illegal in current mode"
5E68 CAA0 MOV  @>001E,@>0042(R10)   : save unit flags
5E6E 0201 LI   R1,>0002             : set unit 2 (4 for centronics)
```

```
5E72 C801 MOV  R1,@>001E               : set new unit flag
5E76 0201 LI   R1,>0A00                : print cr lf
5E7A 0F01 WRIT R1
5E7C 0201 LI   R1,>0D00
5E80 0F01 WRIT R1
5E82 0201 LI   R1,>1B00                : set line spacing to 6/72"
5E86 0F01 WRIT R1
5E88 0201 LI   R1,>4100
5E8C 0F01 WRIT R1
5E8E 0201 LI   R1,>0600
5E92 0F01 WRIT R1
5E94 0201 LI   R1,>1B00                : set colum head to 8
5E98 0F01 WRIT R1
5E9A 0201 LI   R1,>6C00
5E9E 0F01 WRIT R1
5EA0 0201 LI   R1,>0800
5EA4 0F01 WRIT R1
5EA6 04EA CLR  @>0044(R10)             : clear X pointer
5EAA CAA0 MOV  @>EE36,@>0048(R10)      : save cursor
5EB0 04EA CLR  @>004A(R10)             : clear colour data storage area
5EB4 0201 LI   R1,>1B00                : set bit immage mode for 768 bytes
5EB8 0F01 WRIT R1
5EBA 0201 LI   R1,>4C00
5EBE 0F01 WRIT R1
5EC0 0201 LI   R1,>0000
5EC4 0F01 WRIT R1
5EC6 0201 LI   R1,>0300
5ECA 0F01 WRIT R1
5ECC 1000 NOP
5ECE 1000 NOP
5ED0 D82A MOVB @>0045(R10),@>EE36      : set X position
5ED6 D82A MOVB @>0047(R10),@>EE37      : set Y position
5EDC 0201 LI   R1,>F120                : correct fault in COL function
5EE0 C801 MOV  R1,@>1D12
5EE4 0420 BLWP @>1C9E                  : branch to COL function
5EE8 0000 DATA >0000
5EEA D0AA MOVB @>004A(R10),R2          : colour of this pixel to R2
5EEE 1604 JNE  >5EF8
5EF0 D0A0 MOVB @>0548,R2               : if zero get background colour to R2
5EF4 0242 ANDI R2,>0F00
5EF8 0982 SRL  R2,8                    : move it to low byte
5EFA 0203 LI   R3,>0003                : calculate data table offsett
5EFE 38C2 MPY  R2,R3
5F00 0224 AI   R4,>0011
5F04 A10A A    R10,R4
5F06 D174 MOVB *R4+,R5                 : move density data to R5,R6,R7
5F08 D1B4 MOVB *R4+,R6
5F0A D1F4 MOVB *R4+,R7
5F0C B820 AB   @>1D49,@>EE36           : increment X position
5F12 0420 BLWP @>1C9E                  : get colour for this pixel
5F16 0000 DATA >0000
5F18 D0AA MOVB @>004A(R10),R2
5F1C 1604 JNE  >5F26                   : if zero replace with background col
5F1E D0A0 MOVB @>0548,R2
5F22 0242 ANDI R2,>0F00
5F26 0982 SRL  R2,8                    : move to low byte
```

```
5F28 09D5 SRL  R5,13                        : move to low byte
5F2A 09D6 SRL  R6,13                        :       "
5F2C 09D7 SRL  R7,13                        :       "
5F2E 0203 LI   R3,>0003                     : calculate data table offsett
5F32 38C2 MPY  R2,R3
5F34 0224 AI   R4,>0011
5F38 A10A A    R10,R4
5F3A D174 MOVB *R4+,R5                      : get density data to R5,R6,R7
5F3C D1B4 MOVB *R4+,R6
5F3E D1F4 MOVB *R4+,R7
5F40 06C5 SWPB R5                           : rearange data for printer format
5F42 06C6 SWPB R6
5F44 06C7 SWPB R7
5F46 0A45 SLA  R5,4
5F48 0A46 SLA  R6,4
5F4A 0A47 SLA  R7,4
5F4C 0F05 WRIT R5                           : send the data
5F4E 0F06 WRIT R6
5F50 0F07 WRIT R7
5F52 04C7 CLR  R7
5F54 0F07 WRIT R7
5F56 062A DEC  @>0046(R10)                  : decrement Y position
5F5A 0201 LI   R1,>FFFF
5F5E 806A C    @>0046(R10),R1               : check for last Y
5F62 16B6 JNE  >5ED0                        : no go for next pixel
5F64 0002 DATA >0002                        : send cr lf
5F66 0201 LI   R1,>00BF                     : reset Y pointer
5F6A CA81 MOV  R1,@>0046(R10)
5F6E 05EA INCT @>0044(R10)                  : increment X (we print two each pass)
5F72 0201 LI   R1,>00FF                     : check for end of page
5F76 806A C    @>0044(R10),R1
5F7A 129A JLE  >5EB0                        : no setup for next line
5F7C 0201 LI   R1,>1B00                     : reset colum head to zero
5F80 0F01 WRIT R1
5F82 0201 LI   R1,>6C00
5F86 0F01 WRIT R1
5F88 0201 LI   R1,>0000
5F8C 0F01 WRIT R1
5F8E 0201 LI   R1,>1B00                     : reset line spacing
5F92 0F01 WRIT R1
5F94 0201 LI   R1,>3200
5F98 0F01 WRIT R1
5F9A C82A MOV  @>0042(R10),@>001E           : return old unit flags
5FA0 C82A MOV  @>0048(R10),@>EE36           : return old cursor
5FA6 0380 RTWP                              : return to calling programm
```

Competition

In order to encourage all Cortex owners to develop their programming skills we intend to offer a free game for the best short program we receive each quarter.

This time we are looking for a program to demonstrate the graphic capabilities of the Cortex. The actual form of the program is entirely up to you, but it should not be very long. Suggested topics are geometric patterns or imaginative use of coloured sprites.

To enter simply send your program on tape with any additional information we might need to load, run, and list it. We regret that we cannot undertake to return tapes unless return postage is paid. Please state which game you would like to receive if you win. (You may choose any one from the present order form)

The winner will be announced in issue 5, and his/her program will also be featured in the newsletter.

User Info

Information on any of the following points would be gratefully received, and will be passed on to other users via the newsletter.

BOOKS We have received several letters asking about books applicable to Cortex programming. If anyone has found any suitable books then we would be delighted to hear from you. If you feel inclined then why not write a short review for us, and get your name in print!

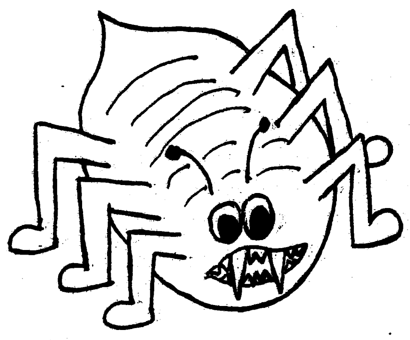
REPAIRS Does anyone know of some organisation which undertakes Cortex repairs. We feel that their is a definite lack of this sort of service, and want users to have more information available to them.

DISC DRIVES A number of people have asked about the compatability of disc drives, and software. We hope to compile a report on this subject for a future newsletter, and would appreciate any additional information.

PRINTERS What printer do you use, and what difficulties have you experienced (if any) ?

HARDWARE ADD-ONS Have you discovered any decent additions to the Cortex, such as memory expansion, interfaces etc ? We are sure that other users would appreciate any knowledge that you have to share.

ANY OTHER POINTS NOT MENTIONED ABOVE



Bug Bytes

This section is concerned with the problems which may arise whilst using your Cortex. If you have any such problems or can offer advice on any points included in previous newsletters, then we will be delighted to hear from you.

Firstly we have a plea from John Mackenzie of Malvern:-

Would some kind person please sort out the bug in the error trap subroutine ! The idea behind the trap is that on a return the line at which the error occurred is re-executed, but as anybody who has used it will know, a normal return is done to the next line, thus missing the original line at which the error occurred. This is easily overcome, but that isn't the point.

John also writes software, and his latest works are a word processor and spelling checker.(see User's advertisments) He also has this to say about the problem Mr Evans had with his video circuit in newsletter 3...

Mr Evans and his screen problem; I don't have an answer to that one but I did at one time have strange happenings on my screen, which I cleared up by soldering the video processor chip staight onto the main Cortex board. This chip runs very hot, and the contacts between the chip legs and the the carrier seem to suffer because of this heat. Propogation delays to the video memory seem to cause the faults. Since soldering the chip to the board I have had no problems.

Paul Sheridan has a few comments on Helge Larsen's keyboard reading routine which was included in newsletter 3.

To stop the screen continually scrolling upwards change line 30 and add line 40 as below;

```
30  IF K=0 : GOTO 20
40  ? K    : GOTO 20
```

The number now stays on the screen after the key has been released. It is also possible to achieve the same result with a 2 line Basic program without the machine code program;

```
10  K=KEY(0) : IF K=0 : GOTO 10
20  ? K  : GOTO 10
```

This program can be further extended to put the character or shape onto the screen as well as it's ASCII code.

```
20  SPUT 610,K  : ? K : GOTO 10
```

Mr Azzopardi of Malta is seeking information on a tape which he purchased a few months ago. The tape has on it a set of four graphics programs entitled "R9", "P15", "SQRS", and "C-SPOTS". He cannot get these to load, and has no documentation with them. If anyone has working versions of these programs and is willing to lend us a copy, then we will be happy to return their tape and refund any postage charges. Any additional information would also be gratefully received.

Mr Teirila of Helsinki has kindly sent in his solution to Mr Radford's problem (Newsletter 3, page 24).

The cause for the problem lies in the fact that Cortex Basic does not store the length of strings, but uses the character 0H as a string terminator, instead. That is why Basic cannot use that character for any other thing.

The simplest solution is to send 80H instead of 0H. This is possible because in Cortex the RS232 is programmed to send only seven bits and the MSB is so ignored. Even if you have reprogrammed the RS232 to send eight bits (as I did) this solution may help, because many printers will accept 80H for 0H in Escape-sequences.

However, if you definately need to send 0H, the only solution is to write a routine that directly manipulates the RS232 port. The following Basic subroutine will send any character (including 0H) whose ascii code is stored in variable CHR.

```
100  REM RS-232  SEND
110  BASE 080H
120  CRB[16]=1
130  A=CRB[22]: IF A=0 THEN GOTO 130
140  A=CRB[27]: IF A=0 THEN GOTO 140
150  CRF[8]=CHR
160  CRB[16]=0
170  RETURN
```

And here is the same program in machine code. It is written in an array in high memory, but is fully relocatable. It is used from Basic by CALL 0EAECH,xx , where xx is the ascii code to be transmitted.

```
EAEC 020C LI   R12,>0080
EAF0 1D10 SBO  16
EAF2 1F16 TB   22
EAF4 16FE JNE  >EAF2
EAF6 1F1B TB   27
EAF8 16FE JNE  >EAF6
EAFA 06C0 SWPB R0
EAFC 3200 LDCR R0,8
EAFE 1E10 SBZ  16
EB00 0380 RTWP
```

Programming Tips

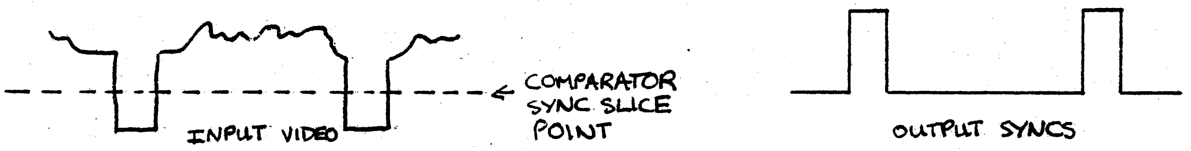
Here is your chance to share your knowledge of your Cortex with other users. If you have discovered any interesting facts related to either software or hardware then write to us and get a mention in the next newsletter. In particular it is extremely useful for inexperienced programmers to have ambiguous or undocumented instructions explained in more detail.

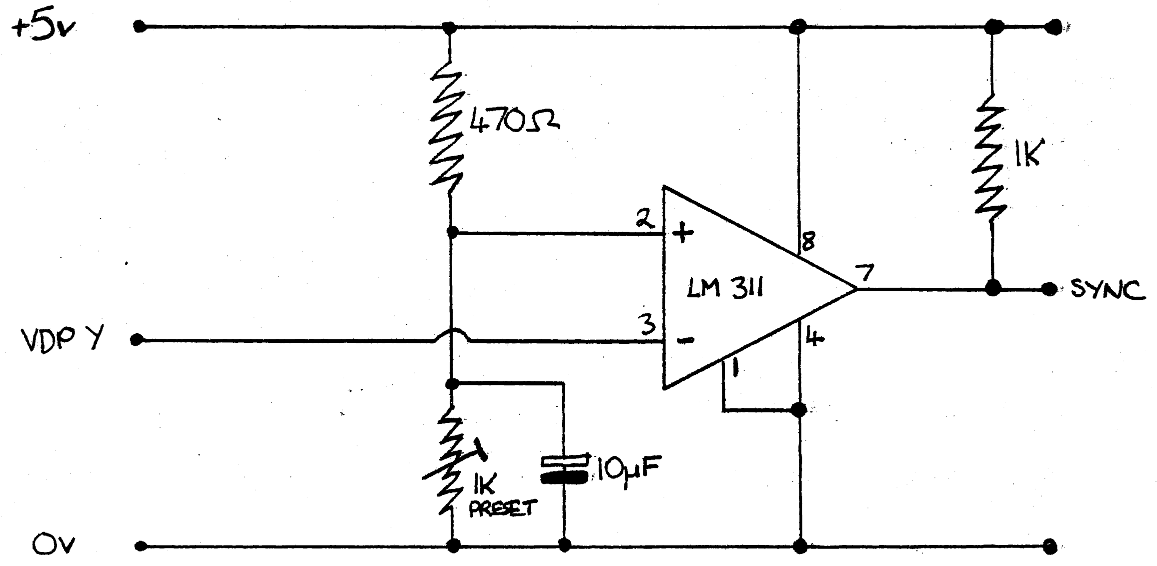
Tim Gray has provided much information in the past and continues to do so with the following points..

The problem of having to change all data memory locations or BL instructions when moving code from one place to another can be avoided by writing position independant code. The following code loads R10 with 045BH, the opcode for the RT instruction, and then performs a branch and link instruction to it. Although the code returns immediately, it leaves the return address in R11. This can then be moved to another register and used as an index for any other BL instructions, or data locations used in the program.

```
LI   R10 >045B                load R10 with RT
BL   R10                      branch and link
MOV  R11, R10                 R11 contains addr of this instruction
BL   @ 0048(R10)    etc....   rest of program
```

Lots of people seem to be having trouble with the video circuits of the Cortex. Most of these problems can be solved by replacing the sync separator transistor Q2 with a comparator IC. The original sync separator stage had problems due to its capacitive coupling causing incorrect output at different brightness levels and over the vertical interval. This in turn caused false triggering of the pal-switch IC which shows itself as pink flashes at the top of the picture or complete loss of colour depending on the type of TV used. The circuit below replaces Q2, R22, R25, and C7, and as it is DC coupled it is completely stable once set up correctly.





In Newsletter three Julian Terry seems to be confused about the reason for the XOP routines . Here are some notes to explain them more fully.

XOP's 0 to 10 are the floating point arithmetic package and are used in the form :- XOP (S),xop number
where S is the start addr of a floating point number Ie 6 bytes as in the address of a basic variable.

XOP 0	LOAD ACC	Loads the floating point accumulator with the 6 bytes starting at S . The accumulator is R0 , R1 , R2 of XOP 0 to 10 , memory addr F09C F09E F0A0
XOP 1	STORE ACC	Stores the accumulator contents to the six bytes begining at S
XOP 2	ADD ACC	Adds the floating point number starting at S to the accumulator . The normalised result will be in the accumulator .
XOP 3	SUBTRACT FROM ACC	Subtracts the floating point number at S from the accumulator . The normalised result will be in the accumulator
XOP 4	MULTIPLY ACC	Multiplys the floating point accumulator by the floating point number begining at S . The normalised result will be in the accumulator
XOP 5	DIVIDE ACC	Divides the floating point accumulator by the floating point number starting at S .The normalised result will be in the accumulator
XOP 6	SCALE	Adjusts the floating point accumulator so that its exponent matches that of the floating point number starting at S
XOP 7	NORMALISE	Adjusts the floating point accumulator to give the greatest number of decimal places :- 5.65 would become .565 E 1 and .000865 E 2 would become .865 E -1
XOP 8	CLEAR ACC	Clears the floating point accumulator
XOP 9	NEGATE ACC	Negates the floatingpoint accumulator by changing the sign bit
XOP 10	FLOAT	Converts the floating point accumulator to floating point if it is an integer

OTHER XOP'S

XOP 11	EVALUATE AND FIX	Used by basic to evaluate parameters and return an integer number
XOP 12	OUTPUT F.P IN ASCII	Converts the floating point number at S to ascii code and stores it at the addr pointed to by R7 on return R7 will contain the end addr of the ascii code to print it add a null byte and use the MSG MID MSG @>start addr
XOP 13	OUTPUT AN INTEGER No	As above but uses a one word integer number starting at S instead
XOP 14	PRINT ERROR MESSAGE	As described in newsletter three

Software Scene

OCT/NOV 1985	TOP 3
1	BURGLAR
2	FROGGER
3	INVADERS & ASTEROID

NEW**NEW**NEW

We are delighted to announce the introduction of four NEW games to our range of software.

OLYMPICS An excellent adaption of one of the newer arcade games, from the author of BURGLAR and FROGGER. Three sections to challenge you in Skeet shooting, archery and weightlifting. This game has the best graphics we have seen on the Cortex, and takes some beating.

FIREBIRD An amazingly fast "shoot 'em up" style arcade game. Swarms of whirling invaders attempt to bomb your solitary craft. Defend yourself from neutron bombs, and ships which require more than one hit to be destroyed. An extremely challenging game which will probably keep you awake all night.

PENGO Another game from the same author as Firebird. Guide your penguin around the screen and attempt to move three diamond blocks together. Crush the marauding monsters with the plentiful ice blocks which litter the screen. Fast action and dazzling special effects.

CENTIPEDE The invasion of the centipedes has begun! Shoot all of their segments before they reach the bottom of the screen. Be careful though, if you hit the centre of one it splits into two smaller centipedes.

Unfortunately two games which appeared on the order form in newsletter 3 were not described.

MOONBASE II Protect your base against the invading spaceships which grow as they get closer. If you lose too many solar panels you will have to undertake to land the supply ship, on which all your hopes depend!

MUNCHER Entertaining Cortex version of Pacman. Large colourful ghosts chase you around the screen, while you try and eat all of the dots. Eat a strawberry to gain a few seconds in which you can chase the ghosts.

NEW SERVICE - AVAILABLE NOW!

We can now supply on tape all of the programs and routines from newsletter 4. Why spend hours bashing at your keyboard when we've done it for you?

As an introductory offer we will also give you a free game!!!

All this for only £1.50 (inc. p&p)

If you have written any good software then why not send it to us (with any necessary documentation), and we will market it. We pay £1.50 royalties for every copy sold (payable every second month). So get writing today!

Cortex Hardware

AVAILABLE NOW !!

BLANK COMPUTER GRADE TAPES DIRECT FROM KPH COMPUTAWARE

C30 TAPES AT 80p EACH (OR 75p EACH FOR 3 OR MORE)

C10 TAPES AT 45p EACH (OR 40p EACH FOR 3 OR MORE)

OFFER AVAILABLE WHILE STOCKS LAST

PRICES INCLUDE P&P

NEW HARDWARE ADD-ONS TO BE RELEASED IN 1986

WE ARE AT PRESENT WORKING ON A NUMBER OF HARDWARE PROJECTS THESE WILL HOPEFULLY INCLUDE:

i) AN INPUT/OUTPUT USER PORT
ii) A SOUND GENERATOR
iii) A SPEECH GENERATOR
iv) A JOYSTICK INTERFACE

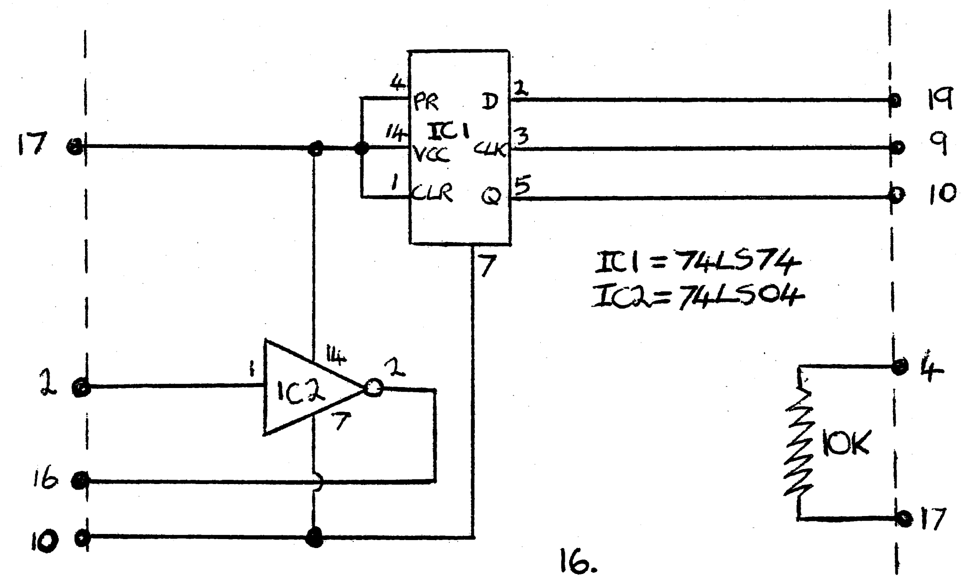
NONE OF THESE PROJECTS WILL INVOLVE SOLDERING ON THE MAIN BOARD

DETAILS OF PROJECTS WILL BE ANNOUNCED IN FURTHER NEWSLETTERS

IF YOU HAVE ANY EXPANSION IDEAS, OR HAVE DESIGNED CIRCUITS YOURSELF, THEN WHY NOT WRITE AND TELL US ABOUT THEM.

74LS2001 REPLACEMENT (E-BUS EXPANSION)

The following circuit should suitably replace the 74LS2001 chip in the E-bus expansion circuit. This circuit does not allow for multiprocessors or bus time-outs.



ORDER FORM

Program title	Price each	Quantity	Cost
ⓐ £6.00 each (or *** £5.00 each)			
BURGLAR			
FROGGER			
INVADERS & ASTEROID			
HUNCHBACK			
MAZE .			
OLYMPICS (new)			
FIREBIRD (new)			
PENGO (new)			
ⓐ £4.00 each (or *** £3.50 each)			
MUNCHER			
GDESIGN			
WALL			
CORTELLO			
ARCHIE			
NIGHT ATTACK			
WINE & FORM1			
SHARES & HISTOGRAM			
MOONBASE II			
CENTIPEDE (new)			
NEWSLETTER 4 PROGRAMS/ROUTINES	£1.50		
C30 BLANK TAPES			
C10 BLANK TAPES			
1986 SUBSCRIPTION (see separate form)	£5.00		
		TOTAL ENCLOSED	

*** SPECIAL CHRISTMAS OFFER ON ORDERS POSTMARKED BEFORE 31-1-86
LOWER PRICE FOR 2 OR MORE PROGRAMS ORDERED (FROM EITHER SECTION)

NAME : ______________________________

ADDRESS: ______________________________

____

____

____

NO VAT TO ADD !!!

NO P&P TO ADD !!!

WORTEX

This is a Word Processor for the Cortex. It runs under CDOS 1.20. The system runs using Twin 40 track single sided single density disk drives. Operation with one drive can be done.

MODES
1. Input text
2. Input page from disk
3. Return input text
4. View disk page
5. Save page to disk
6. Print page/pages
7. Spelling check (requires Speltex)

FUNCTIONS
1. Text input with full character editing
2. Page formating with:
   a. Auto page number
   b. Center text option
   c. Right justify option
   d. Auto left justification
   e. Left margin control
   f. Right margin control
   g. Auto return
   h. Word wrap
   i. 15 Tab markers
   j. Page length control
   k. Page editing
3. Copy from disk page to memory page
4. Multi page Printing

15.00 plus a 51/4 blank disk

SPELTEX

The spelling checker for Wortex. This runs under CDOS 1.20. The system uses twin 40 track single sided disks with drive '0' Single Density and drive '1' Double Density. (NOTE only the most recent version CDOS 1.20 supports Double Density).

This is a must for Wortex users. Comes with about 7000 words and the dictionary can go up to around 20000 words.

MODES
1. Check page spelling
2. Edit the Dictionary
3. Return to Wortex
4. Correct errors

FUNCTIONS
1. View the errors
2. Correct the errors
3. Store the error word in the dictionary
4. Add words to Dictionary dirrect from keyboard
5. Delete words from the Dictionary

10.00 plus two 51/4 DD Disks to

J S Mackenzie
4 Werstan Close
Malvern
WR14 3NH

Querries call 06845-65619 evenings