* CORTEX USER GROUP *

NEWSLETTER ISSUE NO.7 June/July 1986

CONTENTS

2 .. EDITORIAL Software and Hardware News.

3 .. BUG BYTES Problems in Cortex hardware/programming.

4 .. PROGRAMS Rotating 3D pyramid
CDOS modifications
Fractals
List Directory

10 .. USER INFO Your requests and information exchange.

11 .. FEATURE Cassette Interface Modifications.

12 .. SHORT TIPS Programming and hardware suggestions.

14 .. MACHINE CODE Part three : Arithmetic and logical operations.

16 .. USERS ADVERT

---

kph computaware

63 Highlands Road, Andover, Hants. SP10 2PZ

EDITORIAL

Welcome to the seventh issue of the Cortex Users Group Newsletter. We would like to thank all of you who have written to us including tips, programs,..etc. Please keep these letters coming, and feel welcome to write on any matters concerning the Cortex.

SOFTWARE NEWS

We have three new software tapes for you this issue, commencing with an amazing game. (Sorry about the pun.)

'MAZE-3D' is an adventure in three dimensions, in which you are the carrier of a secret document which must be delivered to the authorities to help in a battle against an alien invasion. The only problem is that you are seemingly trapped in an enormous 3 dimensional maze, with only a pair of magnetic boots to help you escape. The screen shows the scene ahead of you as you walk along passageways, and even up walls with your special boots. The graphics are very quickly drawn and easy to resolve.
(£6.00 tape)

'BIDEHE' is a utility program for conversion between number bases. You enter a number in binary, decimal or hexadecimal and the equivalent is displayed in all three bases.
(£4.00 tape)

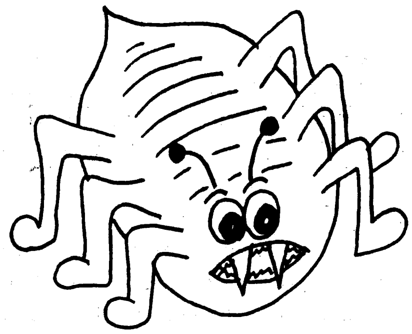
NEWSLETTER 6&7 PROGRAMS: A collection of programs from this and the previous issue. Why spend ages typing when we've done it for you?
(£2.50 tape)

HARDWARE NEWS

We are currently preparing the artwork to produce a new batch of RGB interface boards, and plan to market a kit complete with all components. All enquiries are welcome, and a list will be made of those users who wish to be notified when the details are available.

TMS9911 and 74LS612: We will shortly be receiving a very small number of these chips, which we will sell for £30 each. (NB any orders already received will of course take priority, and the prices at time of ordering will be honoured.)

We have access to most electrical/hardware components via trade accounts, and will be happy to try and obtain items which users have difficulty in buying from normal sources. (Please write for quotations on particular items, including SAE.)



BUG BYTES

Once again we present a collection of problems from Cortex Users. We are always willing to print questions about the Cortex, and appreciate any answers or suggestions which you make. Please specify when writing whether you wish us to print your address, so that individual correspondence can be entered into.

Julian Terry wrote in to inform of the following errors which appeared in his "plane plotter" program in issue 6.

i) line 1820 add ")" after (DIR<>88 (our mistake-ed.)
ii) add line 1825 GOTO 1920
iii) remove line 1860
iv) in line 2260 replace (1 TO 30) by (1 TO 100)

Dave Hunter from Kent wrote with solutions to two of the problems raised in issue 6. (Thank you Dave)

Firstly Julian Terry's 'ILLEGAL DELIMITER' when using 0F020H as the CALL WORKSPACE. MWD(01F20H)=XXXX (16), sets the CALL WP to XXXX (16). To solve the problem in hand, he would also have to type MWD(01F24H) =XXXX (16) +24 (10). ( ) = BASE

ie. Type MWD(01F20H)=0F020H
MWD(01F24H)=0F038H

Mr.J.Stephens of Northumberland can solve his problem by modifying his CDOS boot disc in the following way:-

Using Disc Inspect (DI), display Track 0, sector 8 (assuming single density). This sector is easily found since it is the same sector that has the "SYSTEM$" file name starting at byte 038H.

Once this sector has been found, the bug in CDOS 1.11 can be corrected by changing byte 043H to 0AEH. Thus the line will now read;

40 C8 20 68 AE etc

Now re-boot the system , and CSAVE and CLOAD will work correctly.

Prem Holdaway writes from London to inform us that the following correction should be made to the program in issue 3, page 19.

line 2120 should read:- CE=(FRA(CB))*8

and not :- CE=(FAC(CB))*8

In addition to this Prem would also like to know if anyone has noticed that in Tim Gray's 3D Graph program (issue 2) there would appear to be too many characters on the line.

PROGRAMS

The following programs and routines have been sent in by Cortex Users. We will try and include all the programs that we receive, but we are obviously restricted by the amount of room alloted per newsletter. All of the programs listed here will be available on tape. (see page 7.2)

Our first program comes from our most prolific contributor, *Tim Gray,* and demonstrates the computing and graphics capability of the Cortex. It produces an animated, 3D, rotating pyramid with hidden line removal.

```
 10 REM **** PYRAMID ****
 11 REM ** BY TIM GRAY **
 12 REM
 20 COLOUR 15,1: TEXT
 30 PRINT " PROG 8.2 (ROTATING PYRAMID)": ;
 40 PRINT " THREE DIMENSIONAL ANIMATION ": ;
 50 PRINT " WITH HIDDEN LINE ELIMINATION ": ;
 60 PRINT " AND PAGE MAPPING ": ;
 70 PRINT "   CALCULATING 36 POSITIONS "
 80 PRINT
 90 DIM SP[912]: AD=1: DIM E[6,3]
100 RH=15: TH=0.5: PH=0.9: D=400
110 CX=170: CY=96: S1=SIN[TH]: C1=COS[TH]: S2=SIN[PH]: C2=COS[PH]
120 TN=-0.1: TT=0.1: CT=COS[TT]: ST=SIN[TT]: SO=SIN[TN]: CO=COS[TN]
130 TP=-0.1: SP=SIN[TP]: CP=COS[TP]
140 DATA 0,0,3
150 DATA 1,0,0
160 DATA -0.2,1,0
170 DATA -0.2,-1,0
180 DIM V[4,3]: DIM SV[4,2]
190 FOR I=1 TO 4: READ X,Y,Z
200  V[I,1]=X: V[I,2]=Y: V[I,3]=Z
210  XE=-X*S1+Y*C1: YE=-X*C1*C2-Y*S1*C2+Z*S2: ZE=-X*S2*C1-Y*S2*S1-Z*
     C2+RH
220  SV[I,1]=D*(XE/ZE)+CX: SV[I,2]=-D*(YE/ZE)+CY
230 NEXT I
240 DATA 1,4,2,1
250 DATA 1,2,3,1
260 DATA 1,3,4,1
270 DATA 2,4,3,2
280 DIM S[4,4]
290 FOR I=1 TO 4
300  FOR J=1 TO 4
310   READ S[I,J]
320  NEXT J: NEXT I
330 DIM N[4,3]
340 FOR RO=1 TO 36
350  FOR I=1 TO 6: E[I,3]=0: NEXT I
360  FOR I=1 TO 4
370   U1=V[S[I,2],1]-V[S[I,1],1]
380   U2=V[S[I,2],2]-V[S[I,1],2]
390   U3=V[S[I,2],3]-V[S[I,1],3]
400   V1=V[S[I,3],1]-V[S[I,1],1]
410   V2=V[S[I,3],2]-V[S[I,1],2]
420   V3=V[S[I,3],3]-V[S[I,1],3]
430   N[I,1]=U2*V3-V2*U3
440   N[I,2]=U3*V1-V3*U1
```

```
450   N[I,3]=U1*V2-V1*U2
460  NEXT I
470  XE=RH*S2*C1: YE=RH*S2*S1: ZE=RH*C2
480  N=1
490  FOR I=1 TO 4
500   E2=S[I,1]
510   WX=XE-V[E2,1]
520   WY=YE-V[E2,2]
530   WZ=ZE-V[E2,3]
540   IF N[I,1]*WX+N[I,2]*WY+N[I,3]*XZ<=0 THEN GOTO 650
550   E1=S[I,1]
560   FOR J=2 TO 4
570    E2=S[I,J]
580    FOR K=1 TO N
590     IF E[K,1]=E2 AND E[K,2]=E1 THEN E[K,3]=2: GOTO 630
600    NEXT K
610    E[N,1]=E1: E[N,2]=E2: E[N,3]=1
620    N=N+1
630    E1=E2
640   NEXT J
650  NEXT I
660  FOR I=1 TO 6
670   IF E[I,3]=0 THEN GOTO 700
680   J=E[I,1]: K=E[I,2]
690   SP[AD]=SV[J,1]: SP[AD+1]=SV[J,2]: SP[AD+2]=SV[K,1]: SP[AD+3]=
      SV[K,2]
700   AD=AD+4
710  NEXT I
720  FOR I=1 TO 4
730   T1=CP*CT*V[I,1]-(ST*CP+SO*SP)*V[I,2]+(SO*ST*CP-SP*CO)*V[I,3]
740   T2=ST*V[I,1]+CO*CT*V[I,2]-SO*CT*V[I,3]
750   T3=SP*CT*V[I,1]+(SO*CP-CO*ST*SP)*V[I,2]+(ST*SO*SP+CO*CP)*V[I,
      3]
760   V[I,1]=T1: V[I,2]=T2: V[I,3]=T3
770   X=T1: Y=T2: Z=T3
780   XE=-X*S1+Y*C1: YE=-X*C1*C2-Y*S1*C2+Z*S2: ZE=-X*S2*C1-Y*S2*S1-
      Z*C2+RH
790   SV[I,1]=D*(XE/ZE)+CX: SV[I,2]=-D*(YE/ZE)+CY
800  NEXT I
810  PRINT RO,
820 NEXT RO
830 FOR I=1 TO 48: SP[I+864]=SP[I]: NEXT I
840 PRINT
850 PRINT
860 INPUT "   READY, PRESS RETURN ";$A
870 COLOUR 15,1: GRAPH: DP=0
880 GOSUB 1100
890 AD=1
900 FOR I=1 TO 6
910  IF SP[AD]=0 THEN GOTO 930
920  UNPLOT (SP[AD])-DP,SP[AD+1] TO (SP[AD+2])-DP,SP[AD+3]
930  AD=AD+4
940 NEXT I
950 AD=AD+24
960 FOR I=1 TO 6
970  IF SP[AD]=0 THEN GOTO 990
980  PLOT (SP[AD])-DP,SP[AD+1] TO (SP[AD+2])-DP,SP[AD+3]
990  AD=AD+4
```

```
1000 NEXT I
1010 AD=AD-48
1020 IF AD=865 THEN AD=1
1030 IF DP=0 THEN MP=0
1040   ELSE MP=1
1050 MEM[0F121H]=6+MP: MEM[0F121H]=082H
1060 DP=128-DP
1070 GOTO 900
1080 STOP
1090 REM *** PAGE MAP SUBROUTINE ***
1100 MEM[0F121H]=0: MEM[0F121H]=058H
1110 FOR MP=0 TO 23
1120  FOR MPC=0 TO 7
1130   MEM[0F120H]=0
1140  NEXT MPC
1150  FOR MPD=8 TO 23
1160   MEM[0F120H]=MPD+8+(32*MP)
1170  NEXT MPD
1180  FOR MPE=24 TO 31
1190   MEM[0F120H]=0
1200  NEXT MPE
1210 NEXT MP
1220 MEM[0F121H]=0: MEM[0F121H]=05CH
1230 FOR MP=0 TO 23
1240  FOR MPC=0 TO 7
1250   MEM[0F120H]=0
1260  NEXT MPC
1270  FOR MPD=8 TO 23
1280   MEM[0F120H]=MPD-8+(32*MP)
1290  NEXT MPD
1300  FOR MPE=24 TO 31
1310   MEM[0F120H]=0
1320  NEXT MPE
1330 NEXT MP
1340 RETURN
```

Our next offering was sent in by *John Mackenzie* , and consists of a couple of modifications that can be made to the CDOS utility programs. Early versions of CDOS can be modified, but the line numbers may differ from those shown here.

Ref.CDOS file copy utility 1.2

a) The following changes and/or additions to the BASIC listing will allow you to step through the Directory of the disk which you want to copy from, and select whether or not to copy the files.

```
341 ? @(1,9);"Do you want to copy this file ? ";
342 INPUT ?349,#1;$ANS
343 ? @(1,9);"                                      ";
344 IF $ANS="Y" THEN GOTO 350
345 IF $ANS="y" THEN GOTO 350
346 IF $ANS="N" THEN GOTO 710
347 IF $ANS="n" THEN GOTO 710
348 GOTO 341
349 POP: GOTO 341
```

Now resave the program as say "COPYFILE".

b) The following changes and/or additions to the BASIC listing will allow you to select a particular file by name to copy, which can be easier than stepping through all of the files.

```
120 DIM X[10],B[20],$S[2],M[4096],$N[2]
195 INPUT "File name";#8;$N[0]
345 IF $S[0]<>$N[0] THEN GOTO 710
700 INPUT "Another file ";#1;$Q
704 IF $Q="Y" OR $Q="y" THEN GOTO 100
706 STOP
730 ? @(0,20);"End of directory, file not found."
740 GOTO 700
```

Now resave the program as say "SFILECOP".

Tony Roberts from South Australia sent in the following two programs and the accompanying description.

These two programs draw some interesting and beautiful mathematical sets, called *Fractals.*

The first program draws approximations to the so-called *Koch curve* , which is an example of a line of infinite length, and has been used as a model of coastlines. In fact, it is convenient to think of the Koch curve as a set of points which has a dimension between one (the dimension of a smooth curve) and two (the dimension of a planar figure). By its construction it can be argued that the Koch curve has a dimension of log4/log3=1.2619 . The program to draw the Koch curve is recursive, and the depth of recursion is controlled by the input parameter MAX. The plotted curve becomes a better approximation for larger MAX, and is exact for MAX=infinity. In practise MAX=6 will draw the curve to the maximum resolution of the screen, and lower values will show how the curve is constructed. (Further references : New Scientist, 4 April 1985; B.B.Mandelbrot, The Fractal Geometry of Nature, 1982.)

```
 10 REM *** Koch Curves ***
 15 DIM XA[9], YA[9], XB[9], YB[9], DX[9], DY[9]
 20 INPUT "Draw a Koch Curve of order",MAX: MAX=MAX+1
 30 GRAPH
 40 L=0
 50 XA[1]=3: YA[1]=50
 60 XB[1]=249: YB[1]=50
 70 GOSUB 100
 80 UNPLOT 1,1
 90 GOTO 20
100 L=L+1
110 IF L=MAX THEN PLOT XA[L],YA[L] TO XB[L],YB[L]: L=L-1: RETURN
120 DX[L]=(XB[L]-XA[L])/3
130 DY[L]=(YB[L]-YA[L])/3
140 XA[L+1]=XA[L]: YA[L+1]=YA[L]
150 XB[L+1]=XA[L]+DX[L]
160 YB[L+1]=YA[L]+DY[L]
170 GOSUB 100
180 XA[L+1]=XB[L+1]: YA[L+1]=YB[L+1]
190 XB[L+1]=(XA[L]+XB[L]-SQR[3]*DY[L])/2
200 YB[L+1]=(YA[L]+YB[L]+SQR[3]*DX[L])/2
210 GOSUB 100
```

```
220 XA[L+1]=XB[L+1]: YA[L+1]=YB[L+1]
230 XB[L+1]=XB[L]-DX[L]
240 YB[L+1]=YB[L]-DY[L]
250 GOSUB 100
260 XA[L+1]=XB[L+1]: YA[L+1]=YB[L+1]
270 XB[L+1]=XB[L]: YB[L+1]=YB[L]
280 GOSUB 100
290 L=L-1
300 RETURN
```

The second program, which draws the *Mandelbrot Set* takes a couple of hours to run! The Mandelbrot Set is a set of points in the plane whose boundary is another amazing fractal curve (symmetric about the line y=0). The program plots the set in some rectangular region of the plane, and consequently has three input parameters: the first two being the bottom left corner coordinates; and the third being the horizontal length of the rectangle. (NB these coordinates and length do not correspond to screen pixels). For a first go, try -1, 0, 2 . Then re-run the program with appropriate parameters to look more closely at any part of the boundary of the set, and see the incredibly intricate detail of the Mandelbrot Set. (NB: If you look too closely MAX may need to be increased.)

```
100 REM Draw part of the Mandelbrot Set
105 MAX=100
110 INPUT "Bottom left corner coord. and x-length", ERM, EIM, SCL
120 SCL=SCL/256
130 COLOUR 15,1
140 GRAPH
150 FOR H=0 TO 255
160  ER=ERM+SCL*H
170  FOR V=0 TO 191
180   EI=EIM+SCL*(191-V)
190   X=0
200   Y=0
210   FOR N=1 TO MAX
220    Z=X*X-Y*Y+ER
230    Y=2*X*Y+EI
240    X=Z
250    IF X*X+Y*Y>4 THEN GOTO 280
260   NEXT N
270   PLOT H,V
280  NEXT V
290 NEXT H
300 IF KEY[0]=0 THEN GOTO 300
```

Our last program for this issue was written by *R.M.Lee* using his two pass assembler, as advertised in issue 5. The line number shown on the far left is produced by this assembler, and should otherwise be used only as a reference.

This machine code program will list the directory of a disc. It should be set up to autorun from location 6084H.

Adjust value loaded into R5 in line 9 as follows;

    =30 for SSSD
    =60 for DSSD

Machine code program to list disc directory.

```
                              ORG   >6000             ;SYNTAX IS..
                      BYTEIO  EQU   >6180             ;CAT n <cr>
                      BASIC   EQU   >3F2C             ;WHERE n IS THE..
                      OS      EQU   >3F30             ;DRIVE NO.
6000 0FA0 605E                MSG   @MSG1             ;PRINT HEADING
6004 04C0                     CLR   R0                ;DISC READ
6006 2EC1                     XOP   R1,11             ;GET DRIVE NO.
6008 0241 0003                ANDI  R1,>0003          ;DRIVE 0-3 VALID
600C 06C1                     SWPB  R1                ;HIGH BYTE IS DRIVE
600E 0205 003C                LI    R5,60             ;60 FILES ON DISC
6012 0202 0880                LI    R2,>0880          ;DIRECTORY
6016 0203 5FC0        NEWPRO: LI    R3,>5FC0          ;BUFFER ADDRESS
601A 0204 000A                LI    R4,>000A          ;TRANSFER 10 BYTES
601E 0420 6180                BLWP  @BYTEIO           ;CALL BYTEIO
6022 C1A0 5FC0                MOV   @>5FC0,R          ;CHECK FOR A..
6026 1315                     JEQ   NOPROG            ;PROGRAM ENTRY
6028 04E0 5FCA                CLR   @>5FCA            ;LAST BYTE NULL
602C 0206 5FC2                LI    R6,>5FC2          ;START OF NAME
6030 0207 2000                LI    R7,>2000          ;ASCII SPACE
6034 DD96             NEXT:   MOVB  *R6,*R6+          ;FIND END OF NAME
6036 1304                     JEQ   EON               ;JUMP TO NAME END
6038 0286 5FCA                CI    R6,>5FCA          ;IS IT 8 BYTES
603C 1306                     JEQ   FULNAM            ;GOTO FULLNAME
603E 10FA                     JMP   NEXT              ;
6040 0606             EON:    DEC   R6                ;END OF NAME
6042 DD87             FILL:   MOVB  R7,*R6+           ;PAD OUT NAME TO..
6044 0286 5FCA                CI    R6,>5FCA          ;TABULATE OUTPUT
6048 16FC                     JNE   FILL              ;
604A 0FA0 5FC2        FULNAM: MSG   @>5FC2            ;PRINT NAME
604E 0FA0 6072                MSG   @MSG2             ;TABULATE SCREEN
6052 0222 0040        NOPROG: AI    R2,>0040          ;NEXT ENTRY
6056 0605                     DEC   R5                ;DO THIS 60 TIMES
6058 16DE                     JNE   NEWPRO            ;GOTO NEWPROGRAM
605A 0460 3F2C                B     @BASIC            ;BACK TO BASIC
605E 0A0D             MSG1:   DATA  >0A0D             ;'Disc catalogue'
6060 4469                     DATA  >4469
6062 7363                     DATA  >7363
6064 2063                     DATA  >2063
6066 6174                     DATA  >6174
6068 616C                     DATA  >616C
606A 6F67                     DATA  >6F67
606C 7565                     DATA  >7565
606E 0A0D                     DATA  >0A0D
6070 0000                     DATA  >0000
6072 2020             MSG2:   DATA  >2020             ;12 SPACES
6074 2020                     DATA  >2020
6076 2020                     DATA  >2020
6078 2020                     DATA  >2020
607A 2020                     DATA  >2020
607C 2020                     DATA  >2020
607E 0000                     DATA  >0000
6080 A046             WORD1:  DATA  >A046             ;'CAT' ENCODED
6082 6000             WORD2:  DATA  >6000             ;ADDRESS FOR ENTRY
6084 C820 6080 3A92   SETUP:  MOV   @WORD1,@>3A92     ;SET UP
608A 04E0 3B3E                CLR   @>3B3E            ;TABLE ENTRY
608E C820 6082 4030           MOV   @WORD2,@>4030
6094 0460 3F30                B     @OS               ;BACK TO O.S.
```

USER INFO

This page is your opportunity to exchange knowledge and opinions about any Cortex related products. If you are in search of specific information then send us the details, and we will include them in the next newsletter. Please also indicate if you wish your address to be included as well. Your appraisals of printers, software, hardware add-ons, etc, are also very welcome.

Prem Holdaway wrote in to tell us that the TEAC 50A 40T or the 50F 80T disc drives fit the Cortex. In addition to this the Tandon TM65-4 80T is also suitable.

P.D.Griffiths of Cambridge has an ESPRINT printer working with his Cortex. This is a serial impact dot matrix type printer, and is easily connected to the Cortex via the RS232C interface port. So far the printer has proved to be very reliable.

Mr.Griffiths would like to write a screen dump routine, but has not had much success in converting the routines previously published in this newsletter, despite replacing the relevant control codes. If anyone has any suggestions, or has written a screen dump for use with this printer, then we will be happy to pass on your comments.

(The Esprint printer is available from Display Electronics, London)

Ladislav Vig from Switzerland is a member of the British Amateur Television Club (BATC), and tells us that problems similar to the video faults of the Cortex are sometimes dealt with in their magazine. We would be interested to hear from any other double members, particularly if they can suggest modifications to the Cortex video circuitry.

Mr.J.Stephens from Northumberland would like to know if there is a cheaper source for Cortex PASCAL, rather than buying MDEX.

Mr.Stephens uses a Intelligent Eprom Programmer (E&W.World 84/85), and has adapted the corresponding software. If ther are any other users with the same device then he would be willing to send his program.

If anyone has any information at all on using the E-bus, then we would be very interested to hear from them. We will refund photocopying charges, or will ensure that any original documents are returned safely. Please send any information to the usual address, marked FAO K.P.Holloway.

FEATURE : CASSETTE INTERFACE MODS (*By P.Moyers*)

With reference to the letter from Albert Reilly of Galway (issue 5, page 4) I had experienced similar problems with the cassette interface which were caused by interference from the cassette deck motor/switching randomly triggering IC70. There are a few ways of tackling this problem.

1) MODIFICATIONS TO THE CASSETTE PORT

On IC70 (74LS123) connect pin 3 to +5v (pin 16) and connect in a ceramic capacitor across pins 1 and 2.

Make C21=4n7 , and C23=2n2.

2) MODIFICATION TO THE CASSETTE DECK

Usually, the source of the problem is noise from the cassette deck motor, and a badly regulated power supply which makes the spikes which occur during switching much worse. One way of curing this is to fit Ni.Cad. rechargeable cells and convert the power supply to a trickle charger. (fig.1b)

fig.1a. Usual cassette power supply configuration.

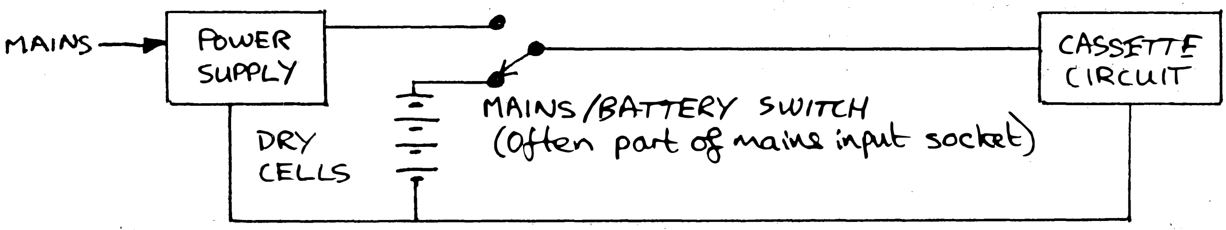
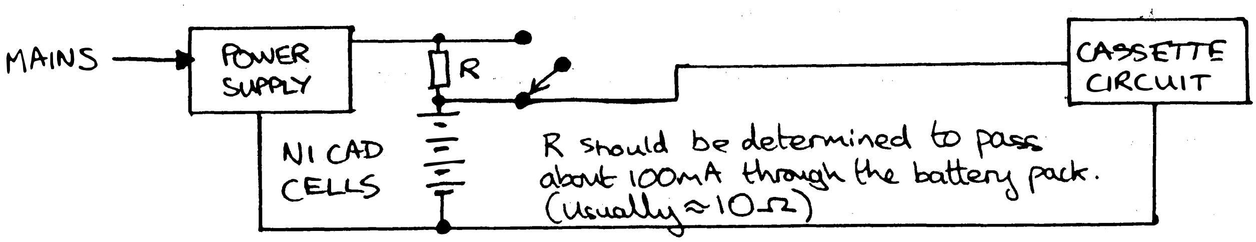


fig.1b. Ni.Cad. Trickle charger modification.

The Ni.Cad. cells have a very low internal impedance, and thus oppose any surges or dips in the power supply rail.



Mr. Moyers has designed a small circuit which acts a cassette signal conditioner and acoustic flag. This circuit is available from Mr. Moyers , and full details can be seen in the advert on page 16.

SHORT TIPS

Tim Gray sent us in some more tips, a few of which are included here;

As standard the DMA controller can't access external memory on the E-bus, which makes it impossible to transfer data direct to disk. The problem can be corrected by performing this simple modification:

1) Isolate pin 5 of IC24 and tie it to +5v ;This stops the signal turning the mapper off.

2) Isolate pin 34 of IC34 TMS9911 and connect it to pin 23 of IC11 TMS9995 ;This allows the TMS9911 to see the E-bus ready signal.

The following program adds two extra monitor commands J and K, which switch the mapper on and off respectively. it is recommended that this routine be included in the Autoexec program if you are using a disk system.

```
100  MWD[06EF0H] = 003A0H
110  MWD[06EF2H] = 00460H
120  MWD[06EF4H] = 00080H
130  MWD[06EF6H] = 003C0H
140  MWD[06EF8H] = 00460H
150  MWD[06EFAH] = 00080H
160  MWD[00A88H] = 04A00H
170  MWD[00A8AH] = 06EF0H
180  MWD[00A8CH] = 04B00H
190  MWD[00A8EH] = 06EF6H
```

Tony Roberts sent us the next item all the way from South Australia.

To disable the autorunning of a program after it has been loaded from tape; change the value of memory location 183AH from 5522H to almost anything else. To restore autorunning change location 183AH back to 5522H again. Tony says that this is particularly useful when transfering programs from tape to disc, since some programs overwrite parts of CDOS upon running.

Chris Young sent in the following tips:-

To stop a BASIC program without any message except CR LF is

MWD[0EFCCH]=0

The cursor position is held as follows;

MEM[0EE36H] for horizontal

MEM[0EE37H] for vertical

NB: For GRAPH mode the values held are multiplied by eight.

Our next item is a collection of tips for CDOS users, supplied by *Syd Champkin* .

1) The very latest version of CDOS 1.20, which Syd received supported the 'star' command for loading files from disk. (ie. * <filename>). Unfortunately, this did not work, and returned an error message 'FILE NOT FOUND'. This was easily corrected by modifying the 'SYSTEM$' file in the following way, using the 'DI' utility.

Change the data byte at location 67 track 004 sector 11 from 8A to 88.

2) Again, using the 'DI' utility, changing the data bytes at location 48 and 4A of track 000 sector 00 from 70 to 80 changes the 'NEW' command on disk from memory address 07000H to 08000H, and also the start of the basic programs from 07000H to 08000H. This modification allows approximately 4K bytes of memory for basic programs supporting assembly language routines.

NB:- When carrying out the modification, the 'DISKCOPY' utility does not work, and returns an error message 'OUT OF MEMORY AT 110'. This can be corrected by changing the following lines in the 'DISKCOPY' utility program.

```
110  DIM X(4), B(4300)
290  DT=INT(4300*6/BPT)
```

This problem is due to the reduction of usable memory space for basic programs.

3) If the contents of memory locations 0FC0H and 01238H are changed from 060F0H to 070F0H, the start address of the 'A' and 'U' commands of the monitor facility will be 070F0H. This fix can be incorporated into the 'AUTOEXEC' program at the time of 'BOOT'.

P.A.Bowman from Switzerland wishes to endorse the suggestions made by Mr.Williams (page 6.3). He also recabled the power supply distribution, and added a more adequate earth cable to the main board.

In addition to this Mr.Bowman also suggests that the proximity of the tv or monitior to the mains PSU in the Cortex can cause a slightly wobbly display.

Our final tip this issue is a lesson learnt from a recent experience of ours. When ordering components from mail order companies, it is always advisable to check on availablity by phoning before placing the order. A recent purchase of ours was taking longer than expected to arrive, and so we phoned to ask how long it would take. The time we were quoted was 16 weeks! Cortex owners should be particularly wary, now that many of the main IC's are becoming obsolete...eg 74LS2001, TMS9909, TMS9911...

MACHINE CODE PROGRAMMING

[3] Arithmetic and Logical Operations (*by Kevin Holloway*)

In this issue we will discuss how to perform simple arithmetic, and about manipulation of data using logical operations. It should be noted that wherever a register or memory address is used to store a result, the previously stored value will be overwritten.

A good starting place would be to add together two numbers, and we will look at a couple of ways of doing this. Say we wish to add a number to the value stored in a register then we would use an 'immediate add' instruction.

eg1) AI R1,6 — *This will add 6 to the value stored in R1. The result will be stored in R1.*

This instruction can also be used to add numbers to any general general memory address, which is a value stored using any of the addressing modes described in part 2. When an instruction can be used on any general memory address, we will give examples using the notation G1, G2, etc, where these can represent R1, R2, @>6234, @>7000(R4), etc.

The other way of performing addition is to add together two general memory addresses. (Remember this may be registers and/or memory locations).

eg2) A G1,G2 — *The values in G1 and G2 are added together, and the result is stored in G2.*

Subtraction can also be performed, although there is no immediate subtraction instruction.

eg3) S G1,G2 — *The value of G1 is subtracted from G2, and the result would be stored in G2.*

Both of the previous instructions operate on whole words of data (ie 2 bytes). If you wish to operate on only the highest (most significant) byte, then there are equivalent instructions AB and SB respectively.

Unlike most assembly languages, the 9995 set includes instructions for multiplication and division: MPY and DIV, respectively.

eg4) MPY G1,R2 — *This multiplies the value of G1 and R2 together and stores the result in R2 and R3. NB the second operand can only be a memory register. Two registers are needed to store the result since the product of two 2 byte numbers may be greater than the maximum number that can be represented by one register (2 bytes). In this example the high (most significant) word of the result would be stored in R2, and the low (least significant) word would be stored in R3.*

eg5) DIV G1,R2 — *This divides R2 by G1, and stores the result in in R2 and R3. The integer part of the result would be stored in R2, and the remainder would be stored in R3.*

For reasons which will be discussed in more detail in later issues, it is sometimes necessary to perform certain logical operations upon data. Two of these operations are given the names 'AND' and 'OR'. We will start with the 'AND' operation, and will discuss what it does.

To understand logical operations it is easiest to imagine them operating on binary numbers, one bit at a time. We usually represent the operation graphically with a table called a 'Truth Table'.

The Truth Table for the operation 'AND'.

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

ie. 0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1

From this we can see where the name 'AND' originates. The result is only 1 when A and B are both 1's.

This can be easily extended to perform an AND operation on two numbers, each of several bits. (8 in the following example)

```
eg6)   57 AND 23     57 = 00111001       }Perform AND column by column.
                     23 = 00010111       }
                          --------
                          00010001 = 17
                          --------
```

ie. 57 AND 23 = 17

This operation is implemented on the Cortex using the instruction ANDI. (AND Immediate)

eg7) ANDI R1,23 — *This performs a logical AND upon the value stored in R1 and the number 23. The result would be stored in R1.*

The 'OR' operation can be used in much the same way, as shown and described below.

The Truth Table for the 'OR' operation.

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

The result is 1 if A or B, or both, are 1's.

Implementation of the OR operation;

eg8) ORI R1,37 — *This performs a logical OR upon the value stored in R1, and the number 37. The result is stored in R1.*

Another useful logical operation is to invert a number. This is given many names, such as NOT, INVERT, 1's COMPLEMENT, etc, but basically means that wherever a '1' appears in the binary equivalent of the number it is replaced by a '0', and vica versus. This can be represented by a truth table, although there is a slight difference, since it only operates on one number (operand).

A	INV A
0	1
1	0

The truth table for logical inversion.

eg9) INVERSE OF 57   57 = 00111001

=198   11000110 = 198

This is implemented on the Cortex using the INV instruction.

eg10) INV G1   *This inverts the value of G1, and stores the result back in G1.*

We have now covered many instructions which we can use in machine code programming. In the next issue we will progress to looking at some example routines, so that we can gain a better understanding of the instructions covered so far.
