

CORTEX USERS GROUP

T Gray, 1 Larkspur Drive, Featherstone, Wolverhampton, West Midland WV10 7TN.
E Serwa, 93 Long Knowle Lane, Wednesfield, Wolverhampton, West Midland WV11 1JG.
Tel No: T Gray 0902 729078, E. Serwa 0902 732659

1 5

CORTEX USER GROUP NEWSLETTER (DECEMBER 1987)

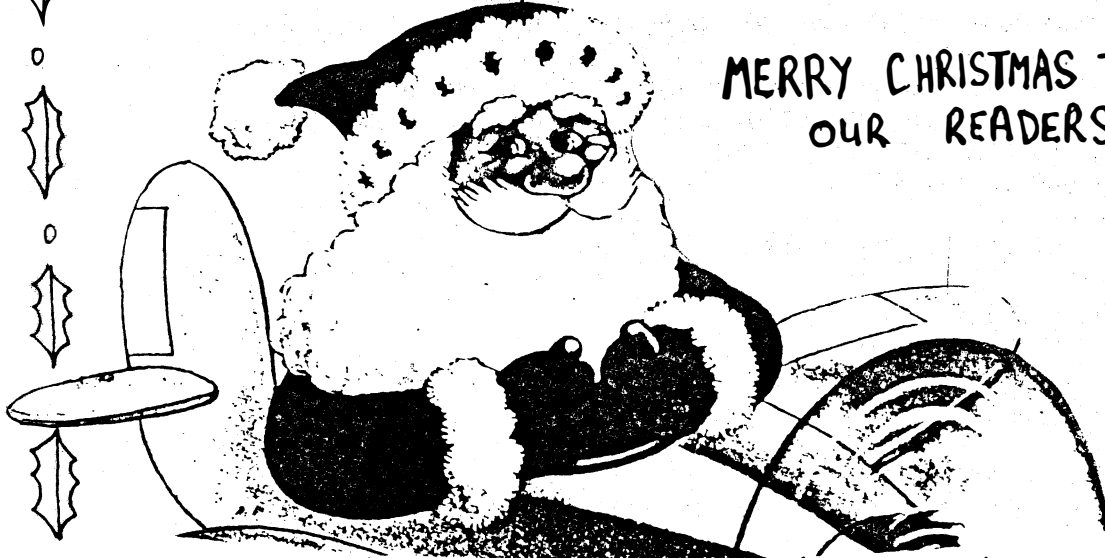
Issue Number 15

CRISTMAS SPECIAL PROGRAMME EDITION.

CONTENTS

1. Index
2. Programme (Missile Command)
6. Programme (Canyon)
8. Programme (Ramdisc format utility)
9. Programme (Diskname utility)
10. Programme (Disk verify utility)
14. Feature MDEX disk operating system
16. Feature using QBASIC part one
19. 9938 VDP interface
20. Extending E. Bus memory
23. Mouse - Keyboard interface

MERRY CHRISTMAS TO ALL
OUR READERS



REMEMBER TO SEND IN YOUR ARTICLES FOR THE NEXT NEWSLETTER

MISSILE COMMAND

MARK RUDNICKI

```

10 MWD[01D12H]=0F120H
20 DIM MX[10],MY[10],BL[2],MD[10]
30 SH=1: K1=1: K9=1: K8=5: M7=5: P6=0: DIM MZ[10]
40 SN=3
50 DIM MH[10],MS[10],M1[10],M2[10],ML[10],MT[10]
60 DIM BP[5],PS[5],PX[5],PY[5],PC[5],BS[5]
70 FOR I=0 TO 5: BS[I]=1: NEXT I
80 MC=3+INT[RND*12.99]
90 BG=K1: FG=3+INT[RND*12.99]: IF BG>15 THEN BG=1
100 IF BG=FG THEN GOTO 90
110 IF MC=BG THEN GOTO 80
120 COLOUR FG,BG: GRAPH
130 MAG 0,0: F6=6
140 SPRITE 1,0,-16,0,0: SPRITE 2,0,-16,0,0
150 SHAPE 0,02H,070FH,0707H,0703H
160 SHAPE 1,0C1EH,037B5H,03D3FH,03B31H
170 SHAPE 2,010H,0387CH,03838H,03870H
180 SHAPE 3,0101H,0107H
190 SHAPE 4,0F161H,061FFH,OFFH,OFFFH
200 SHAPE 5,0E0A0H,0A0F0H,018FCH,0FEFFH
210 SHAPE 6,0404H,0404H,021AH,0397DH
220 SHAPE 7,0,0,020H,04080H
230 SHAPE 8,07756H,05F7FH,OFFE9H,0E9F9H
240 SHAPE 9,0C030H,0F00H,0,080FEH
250 SHAPE 10,0,0,08142H,02119H
260 SHAPE 11,0,0,0204H,0830H
270 SHAPE 12,0701H,02B3FH,01716H,01EFEH
280 SHAPE 13,0C0H,0A8F8H,0F050H,0507FH
290 SHAPE 14,0,0,04H,0201H
300 SHAPE 15,02020H,02020H,04058H,09CBEH
310 SHAPE 16,030CH,0F000H,0,01EFH
320 SHAPE 17,0EE6AH,0FAFEH,0FE97H,0979FH
330 SHAPE 18,OFFFFH,OFFFFH,OFFFFH,OFFFFH
340 SHAPE 32,0,0,0,0
350 SHAPE 33,0,0,0,0
360 SHAPE 34,01010H,010FEH,01010H,01000H
370 SHAPE 40,0,0,03EH,07F7FH: SHAPE 41,0,0,0,0: SHAPE 42,0,0,0,0
380 SHAPE 43,07E1CH,0,0,0: SHAPE 44,0,OFFH,OFFFH,OFFFH
390 SHAPE 45,0,080H,0C0E0H,0E0C0H: SHAPE 46,0,01H,0303H,0100H
400 SHAPE 47,07F3EH,0,0,0: SHAPE 48,OFFH,OFFH,OFFFH,OFFFH
410 SHAPE 49,0E0H,01000H,0C0F0H,0E0C0H: SHAPE 50,03H,0408H,0107H,0301H
420 SHAPE 51,OFFE7H,01800H,0,0
430 SHAPE 60,0,0C370H,03F1FH,0107H: SHAPE 61,0,0E080H,0F8FCH,0C080H
440 SHAPE 62,OFF08H,01C3FH,03F3EH,01C00H: SHAPE 63,08002H,06FEH,0,0
450 SHAPE 70,0FC84H,0BFA5H,0A5FDH,0213FH
460 DEF FNB[X,Y]=(MT[I]-X)/(152-Y)
470 C=0: F5=0: F7=0
480 DEF FNA[X,Y]=INT[(X+0.5)/8]+32*INT[(Y+0.5)/8]
490 SPRITE 0,0,-16,34,FG
500 FOR I=0 TO 5: PS[I]=0: NEXT I
510 DATA 644,647,650,659,662,665
520 FOR I=0 TO 5
530 READ BP[I]
540 NEXT I
550 FOR I=0 TO 255: A=INT[RND*9]
560 A=A+16: IF I<32 OR I>224 OR (I>112 AND I<144) THEN A=A+8

```

"Missile Command"

Cursor Keys - Move target cross
A.S.D - fire from Bases.

```

570 PLOT I,191 TO I,191-A
580 NEXT I
590 COLOUR 13,BG
595 IF BG=13 THEN COLOUR 15,BG
600 FOR I=0 TO 5
610 IF BS[I]=0 THEN GOTO 650
620 FOR J=0 TO 2
630 SPUT BP[I]+J,J: SPUT BP[I]+J+32,J+3
640 NEXT J
650 NEXT I
660 COLOUR 6,BG
665 IF BG=6 THEN COLOUR 14,BG
670 SPUT 608,6: SPUT 609,7: SPUT 640,8: SPUT 641,9
680 SPUT 623,10: SPUT 624,11: SPUT 655,12: SPUT 656,13
690 SPUT 638,14: SPUT 639,15: SPUT 670,16: SPUT 671,17
700 PX=128: PY=80
710 PRINT @(12,3);"GET READY.": ? @(10,5);"X";K1;" SHEET ";SH
720 WAIT 150: PRINT @(12,3);"           ": ? @(10,5);"           "
730 FOR I=0 TO M7: MS[I]=1
740 IF SH>3 AND RND>0.9 THEN MS[I]=2
750 MX[I]=INT[RND*255]: MY[I]=0
760 MD[I]=(((BP[RND*5.99]-638.5)*8)-MX[I])/152
770 M1[I]=MX[I]: M2[I]=MY[I]
780 MH[I]=INT[(250*RND)/SH]
790 NEXT I
800 COLOUR MC,BG
810 GOTO 1050
820 F6=0: FOR I=0 TO M7
830 F6=F6+MS[I]
840 IF MS[I]=0 THEN GOTO 970
850 IF MH[I]<>0 THEN MH[I]=MH[I]-1: GOTO 970
860 IF MS[I]=2          1980: GOTO 970
870 IF MS[I]=3 THEN GOSUB 2040: GOTO 970
880 IF COL[MX[I],MY[I]+1]<>BG AND COL[MX[I]+1,MY[I]+1]<>BG THEN GOTO 980
890 PLOT MX[I],MY[I]
900 MY[I]=MY[I]+K9: IF MY[I]>153 THEN MY[I]=MY[I]-K9: GOTO 980
910 MX[I]=MX[I]+MD[I]*K9
920 PLOT TO MX[I],MY[I]
930 IF RND>0.9 AND MS[I+1]=0 AND MY[I]<88 THEN GOTO 950
940 GOTO 970
950 M1[I+1]=MX[I]: M2[I+1]=MY[I]: MS[I+1]=1: MX[I+1]=MX[I]: MY[I+1]=MY[I]
960 MD[I+1]=(((BP[RND*5.99]-638.5)*8)-MX[I])/((152-MY[I]))
970 NEXT I: RETURN
980 MS[I]=0: UNPLOT MX[I],MY[I] TO M1[I],M2[I]
990 UNPLOT MX[I]-1,MY[I] TO M1[I]-1,M2[I]
1000 UNPLOT MX[I]+1,MY[I] TO M1[I]+1,M2[I]
1010 IF MY[I]>=150 THEN GOSUB 1410: GOTO 970
1020 SC=SC+K1*25: A5=15: IF FG=15 THEN A5=1
1030 COLOUR A5,FG: PRINT @(5,23);SC: COLOUR MC,BG
1040 GOTO 970
1050 A=CRF[0]: B=CRF[0]: IF A<>B THEN GOTO 1130
1060 M=KEY[0]
1070 IF A=16883 OR A=21491 OR A=17651 THEN GOSUB 1190
1080 IF A=2291 AND PX>7 THEN PX=PX-4: GOTO 1120
1090 IF A=2547 AND PX<240 THEN PX=PX+4: GOTO 1120

```

```

1100 IF A=3059 AND PY>8 THEN PY=PY-4: GOTO 1120
1110 IF A=2803 AND PY<152 THEN PY=PY+4
1120 SPRITE 0,PX,PY
1130 C=C+1: IF MOD[C,K8]=0 THEN GOSUB 820: IF F6<>0 THEN GOTO 1050: ELSE STOP
1140 IF F6=0 THEN F7=F7+1: IF F730 THEN GOTO 1630
1150 IF F6=0 AND F7=2 THEN F6=6: GOTO 730
1160 IF P6 THEN GOSUB 1890: GOTO 1180
1170 IF RND>0.997 THEN GOSUB 1810
1180 GOSUB 1290: GOTO 1050
1190 FOR I=0 TO 5: IF PS[I]=0 THEN GOTO 1210
1200 NEXT I: RETURN
1210 PS[I]=1
1220 IF A=21491 THEN GOTO 1270
1230 IF A=17651 THEN GOTO 1280
1240 PLOT 16,152 TO PX,PY
1250 WAIT 3: UNPLOT 16,152 TO PX,PY
1260 PX[I]=PX: PY[I]=PY: PC[I]=0: RETURN
1270 PLOT 127,152 TO PX,PY: WAIT 3: UNPLOT 127,152 TO PX,PY: GOTO 1260
1280 PLOT 232,152 TO PX,PY: WAIT 3: UNPLOT 232,152 TO PX,PY: GOTO 1260
1290 FOR I=0 TO 5: IF PS[I]=0 THEN NEXT I: RETURN
1300 A1=FNA[PX[I],PY[I]]
1310 PC[I]=PC[I]+1
1320 IF MOD[PC[I],4]<>0 THEN GOTO 1340
1330 ON INT[PC[I]/4] THEN GOTO 1360,1370,1380,1390,1400
1340 NEXT I: RETURN
1350 SPUT A1,C1: SPUT A1+1,C2: SPUT A1-1,C3: SPUT A1+32,C4: GOTO 1340
1360 C1=40: C2=41: C3=42: C4=43: GOTO 1350
1370 C1=44: C2=45: C3=46: C4=47: GOTO 1350
1380 C1=48: C2=49: C3=50: C4=51: GOTO 1350
1390 C1=52: C2=53: C3=54: C4=55: GOTO 1350
1400 C1=32: C2=32: C3=32: C4=32: PS[I]=0: GOTO 1350
1410 ES=0: X=INT[(MX[I]/8)+641.5]
1420 FOR H=0 TO 5: IF ABS[X-BP[H]]<4 AND SGN[X-BP[H]]>-1 THEN GOTO 1440
1430 NEXT H: RETURN
1440 A1=BP[H]+1: ES=3: BS[H]=0
1450 ES=ES+1: IF ES=21 THEN GOTO 1550
1460 ON INT[ES/4] THEN GOTO 1500,1510,1520,1530,1540
1470 REM
1480 SPUT A1,C1: SPUT A1+1,C2: SPUT A1-1,C3: SPUT A1+32,C4
1490 GOTO 1450
1500 C1=40: C2=41: C3=42: C4=43: GOTO 1480
1510 C1=44: C2=45: C3=46: C4=47: GOTO 1480
1520 C1=48: C2=49: C3=50: C4=51: GOTO 1480
1530 C1=52: C2=53: C3=54: C4=55: GOTO 1480
1540 C1=32: C2=32: C3=32: C4=32: GOTO 1480
1550 A=0: FOR N=0 TO 5: A=A+BS[N]: NEXT N: IF A>0 THEN RETURN
1560 WAIT 200: COLOUR FG,BG
1570 PRINT @(12,3);"GAME OVER"
1580 SC=0
1590 WAIT 200: ? @(4,5)"PRESS <SPACE> TO RESTART."
1600 IF KEY[32] THEN GOTO 1620
1610 GOTO 1600
1620 WAIT 200: RESTOR : GOTO 10
1630 BC=0

```

```

1640 FOR I=0 TO 5
1650   IF BS[I]=1 THEN BC=BC+100*K1
1660   IF BS[I]=0 THEN GOTO 1700
1670   FOR J=0 TO 2: SPUT J+4*(BC/(100*K1))+160,J
1675     SPUT J+4*(BC/(100*K1))+192,J+3
1680   NEXT J
1690   WAIT 50
1700   PRINT @(10,3);"BONUS ";BC
1710 NEXT I
1720 SC=SC+BC
1730 SN=3
1740 SH=SH+1
1750 IF MOD[SH,3]<>0 THEN GOTO 1780
1760 IF SH>3 THEN K9=K9+1: IF K9>6 THEN K9=5
1770 M7=M7+1: IF M7>9 THEN M7=9
1780 WAIT 150: K1=INT[(SH+1)/2]: P6=0
1790 RESTOR
1800 GOTO 80
1810 P6=1: FY=32+INT[RND*100]
1820 ON 1+INT[RND*1.99] THEN GOTO 1850,1870
1830 SPRITE 1,FX,FY,FV,FC: SPRITE 2,FX+8,FY,FV+1,FC: RETURN
1840 REM
1850 FD=2: FX=2: FV=60: FC=6: IF BG=6 OR BG=8 OR BG=9 THEN FC=10: GOTO 1830
1860 GOTO 1830
1870 FD=-2: FX=238: FV=62: FC=4
1875 IF BG=7 OR BG=5 OR BG=4 THEN FC=15: GOTO 1830
1880 GOTO 1830
1890 IF FD=-2 THEN GOTO 1950
1900 FX=FX+FD
1905 IF FX>240 THEN P6=0: SPRITE 1,0,-16,0,0: SPRITE 2,0,-16,0,0: RETURN
1910 FOR I=0 TO 5: IF PS[I]=0 THEN GOTO 1940
1920   IF ABS[PX[I]-FX-8]<8 AND ABS[PY[I]-FY-4]<4 THEN GOTO 1970
1930 NEXT I
1940 SPRITE 1,FX,FY: SPRITE 2,FX+8,FY: RETURN
1950 FX=FX+FD
1955 IF FX<4 THEN P6=0: SPRITE 1,0,-16,0,0: SPRITE 2,0,-16,0,0: RETURN
1960 GOTO 1910
1970 SC=SC+K1*250: SPRITE 1,0,-16,0,0: SPRITE 2,0,-16,0,0: P6=0: RETURN
1980 MZ[I]=SN: SN=SN+1
1990 MS[I]=3
2000 ML[I]=7: IF BG=7 OR BG=4 OR BG=5 THEN ML[I]=15
2010 SPRITE MZ[I],MX[I],MY[I],70,ML[I]
2020 A=INT[RND*5.99]: MT[I]=(BP[A]-638)*8
2030 RETURN
2040 L9=K9: IF K9>2 THEN L9=2.5
2050 MY[I]=MY[I]+2*(L9): MX[I]=MX[I]+(L9)*MD[I]*2
2060 IF COL[MX[I]+4,MY[I]+4]<>BG AND COL[MX[I]+5,MY[I]+4]<>BG THEN GOTO 2170
2070 IF MY[I]>=150 THEN POP : SPRITE MZ[I],0,-16,0,0: MS[I]=0: GOTO 1410
2080 IF COL[MX[I]+4,MY[I]+10]<>BG AND COL[MX[I]+5,MY[I]+10]<>BG THEN GOTO 2110
2090 SPRITE MZ[I],MX[I],MY[I]
2100 RETURN
2110 MY[I]=MY[I]-INT[RND*8]: MX[I]=MX[I]-4+INT[RND*7.99]
2120 IF MY[I]<1 THEN MY[I]=4
2130 IF MX[I]<4 THEN MX[I]=4: GOTO 2150
2140 IF MX[I]>248 THEN MX[I]=248
2150 MD[I]=FNB[MX[I],MY[I]]
2160 GOTO 2090
2170 SC=SC+K9*100: MS[I]=0: SPRITE MZ[I],0,-16,0,0: RETURN

```

```

10 REM
20 REM
30 BG=1: FG=12
40 COLOUR FG,BG
50 CHAR 023H,0,03FFCH,0
60 CHAR 024H,0FH,OFFFFH,0FOOOH
70 CHAR 025H,03FFH,OFFFFH,OFFCOH
80 CHAR 026H,079E3H,0CFH,0F300H
90 CHAR 027H,0CFH,0FF30H,0C7BFH
100 CHAR 061H,021C2H,01C73H,0FBEBH
110 CHAR 062H,0DD58H,0C18DH,0DFH
120 CHAR 063H,082H,0820H,08200H
130 CHAR 064H,0470H,0431CH,0F7FFH: CHAR 065H,0300H,020C3H,08F3EH
140 CHAR 066H,04F0H,04314H,0947FH: CHAR 067H,0380H,02040H,0813EH
150 CHAR 068H,0410H,0439EH,0FFFFH: CHAR 069H,0,020CBH,0AFBEH
160 CHAR 06AH,0F79CH,0431CH,0F300H: CHAR 06BH,078E1H,0A0C3H,08600H
170 TEXT
180 DIM EM[30,1,6]
190 PO=260: P1=0: P2=0: P=14: Q=18: C=0
200 MA=0: CA=1: B=0: BE=0: B1=0: NL=3: FI=0: FF=0: FX=0
210 FOR I=3 TO 30
220   K=33-I
230   EM[I,0,0]="": EM[K,1,0]=" "
240   FOR J=1 TO INT[0.34+I/3]
250     EM[I,0,0]=EM[I,0,0]+"#": EM[K,1,0]=EM[K,1,0]+"%"
260   NEXT J
270   FOR J=1 TO INT[I/3]: EM[I,0,0]=EM[I,0,0]+"ε"
280     EM[K,1,0]=EM[K,1,0]+"ε": NEXT J
290   FOR J=1 TO INT[0.67+I/3]: EM[I,0,0]=EM[I,0,0]+"%"
300     EM[K,1,0]=EM[K,1,0]+"#": NEXT J
310 NEXT I
320 PRINT @(0,23): ?
330 FOR I=1 TO NL: SPUT PO-2*(I-1),026H: NEXT I: WAIT 50
340 B=0: BE=0: FI=0: FF=0
350 SPUT PO,32
360 PO=PO+P1: P1=0
370 IF B=1 THEN B1=B1+1: IF B1=11 THEN B1=0: B=0: BE=0
380 L=0: SGET PO+40,L: IF L<>32 AND L<>42 THEN GOTO 860
390 PRINT EM[P,0,0]; TAB (Q+6);EM[Q,1,0]
400 SPUT PO,026H+MOD[B1,2]
410 IF Q-P>7 AND RND>0.7 AND C<1500+100*CA THEN ? @(((P+Q)/2),22);"#ε%ε#"
420 C=C+2: IF C>400 AND C<1500+100*CA AND Q-P>6 THEN Q=Q-1: P=P+1
430 IF C>1500+100*CA
440 IF C<>1800+100*CA THEN GOTO 500
450 FG=FG+1: FG=FG LAND 15: IF FG<2 THEN FG=2
460 IF FG=BG THEN GOTO 450
470 COLOUR FG,BG: CA=CA+1: IF MOD[CA-1,4]=0 THEN BG=8-BG: COLOUR FG,BG: NL=NL+1
480 C=0
490 IF BG=7 AND FG=10 OR FG=11 OR FG=3 OR FG=5 THEN FG=12: COLOUR FG,BG
500 IF C<400 OR C>1500+100*CA THEN GOTO 600
510 IF RND>0.95 THEN ? @(P/2+Q/2+RND*4+2,22);"de": MA=MA+1: GOTO 530
520 IF CA>1 AND BE=0 AND RND>0.97 THEN BE=1: ? @(P/2+Q/2+RND*4+2,22);"fg": MA=MA+1
530 IF BE>0 THEN BE=BE+1: IF BE=40 THEN BE=0
540 IF BE>0 AND BE=(24-(INT[PO/40])) THEN B=1: FOR I=-3 TO 3: SPUT PO+I,02DH: NEXT I

```

"Canyon"

← → Move left and right
 R.F Reverse. Forward
 E. Fire.

```

550 IF CA<3 OR FI>0 OR RND<0.95 THEN GOTO 570
560 FI=1: FX=INT[(P/2+Q/2+RND*4+2)+0.5]
565 ? @(FX,21);"hi";: ? @"DLL";"jk": FF=0: MA=MA+1
570 IF FI>0 AND 22-FI>INT[PO/40] AND (MOD[PO,40]=FX OR MOD[PO,40]=FX+1) THEN FF=FF+1
580 IF FI>0 THEN FI=FI+1: IF FI>20 THEN FI=0: FF=0: FX=0: GOTO 600
590 IF FF>6 AND INT[PO/40]<(22-FF) THEN GOSUB 910
600 P=P+INT[RND*2.99]-1: Q=Q+INT[RND*2.99]-1
610 IF Q-P<-1 THEN Q=Q+1: P=P-1
620 IF (C<400 OR C>1500+100*CA) AND Q-P<1 THEN Q=Q+1: P=P-1
630 IF P<3 THEN P=3: GOTO 650
640 IF P>30 THEN P=30
650 IF Q<3 THEN Q=3
660 IF Q>30 THEN Q=30
670 A=KEY[0]: A=CRF[0]: IF A<>CRF[0] OR B=1 THEN GOTO 350
680 IF A=2291 THEN P1=-1: GOTO 350
690 IF A=2547 THEN P1=1: GOTO 350
700 IF A=21235 AND PO>80 THEN P1=-80: GOTO 350
710 IF A=18163 AND PO<800 THEN P1=40: GOTO 350
720 IF A<>17907 THEN GOTO 350
730 P2=PO+40
740 L=0: SPUT P2-40,32: SGET P2,L: SPUT P2,063H
750 IF L<>32 AND L<>42 THEN GOTO 780
760 P2=P2+
770 GOTO 350
780 IF L<026H OR L=98 THEN SPUT P2,062H: GOTO 770
790 SPUT P2,42: IF L=064H THEN SPUT P2+1,32: GOTO 850
800 IF L=065H THEN SPUT P2-1,32
810 IF L=067H THEN SPUT P2-1,32: BE=0
820 IF L=066H THEN SPUT P2+1,32: BE=0
830 IF L=068H THEN SPUT P2+1,32: SPUT P2+40,32: SPUT P2+41,32: FI=0
840 IF L=069H THEN L=065H: SPUT P2+40,32: SPUT P2+39,32: FI=0: GOTO 800
850 MA=MA-1: COLOUR FG,BG: GOTO 350
860 WAIT 150: NL=NL-1: IF NL>0 THEN TEXT : GOTO 320
870 ? "YOU LET THROUGH "MA
880 ? "SCORE=";C*5-50*MA
890 ?
900 WAIT 150: RESTOR : CHAR : TEXT : GOTO 10
910 FOR I=(22-FF) TO INT[PO/40] STEP -1
920 SPUT I*40+FX,02BH: NEXT I: GOTO 860

```

Canyon- use the left and right cursors to move the plane, F to move forwards, R to go back, and E to fire. Destroy the enemy aircraft. Beware the beamers (skeltal outlines) and jet planes (these fire back).

RDFORM

T.Gray.

This programme can be used to initialise Ramdisk with a blank directory. It also works on normal disks and is a quicker way of clearing the directory than using format. It does not remove the boot track. Please use with care.

```
10 TEXT : ? @"C";"CDOS Ramdisk Format Utility 1.0 (C) 1987": ?
20 DIM B[100],$N[2],X[20]
30 AX=ADR[X[0]]: AB=ADR[B[0]]
40 REM *** MACHINE CODE ROUTINE ***
50 DATA 0420H,06180H,0D000H,01601H
60 DATA 0380H,0460H,06550H,04F2H
70 DATA 04D2H,0C0F1H,0704H,0A13H
80 DATA 01701H,0592H,0600H,01601H
90 DATA 0380H,0A14H,016F8H,010F5H
100 FOR I=AX TO AX+38 STEP 2
110 READ IAQ: MWD[I]=IAQ
120 NEXT I
130 REM *** INPUT DRIVE ***
140 INPUT 1"Drive ",D: ?
150 INPUT 1"Are you sure", $ANS
160 IF $ANS="Y" THEN GOTO 190
170 IF $ANS="y" THEN GOTO 190
180 STOP
190 DC=MWD[06382H+D*2] !Drive data
200 BS=MWD[DC] ! sectors / track
210 NB=MWD[DC+2] ! total sectors
220 DS=MWD[DC+4] ! directory start
230 ND=MWD[DC+6] ! number of entries
240 BPS=MWD[06362H+D*2] ! bytes / sec
250 REM *** CLEAR BUFFER ***
260 FOR I=0 TO 600
270 MEM[AB+I]=0
280 NEXT I
290 REM *** CALCULATE NUMBER OF SYSTEM SECTORS ***
300 SS=DS+ND*64/BPS
310 SB=SS/8-1
320 SR=MOD[SS,8]
330 BR=0: TMP=128
340 IF SR=0 THEN GOTO 370
350 BR=BR+TMP: TMP=TMP/2
360 SR=SR-1: GOTO 340
370 REM *** CLEAR SECTORS TO END OF DIR ***
380 FOR I=0 TO SS
390 CALL AX,1,D*256,I*BPS,AB,BPS
400 NEXT I
410 REM *** SET BIT MAP FOR DIRECTORY ***
420 FOR I=0 TO SB
430 MEM[AB+I]=255
431 MEM[AB+I+1]=BR
440 NEXT I
460 CALL AX,1,D*256,BS*BPS,AB,NB/8
470 ? : ? "Done"
480 END
```


DISKNAME

C. J. YOUNG

Puts a name on the disk for use by PDIR etc.

```
10  REM
20  REM *****
30  REM *
40  REM *
50  REM * DISK NAME *
60  REM *
70  REM *
80  REM *****
90  REM
100 TEXT
110 ? "Disk name program"
120 ? "Input Drive ? ";
130 IK=KEY[0]
140 IF IK=0: GOTO 130
150 IF IK<48: GOTO 130
160 IF IK>51: GOTO 130
170 DRV=IK-48
180 ? DRV
190 DIM B[99]
200 DIM $NM[9]
210 DIM MC[10]
220 AMC=ADR[MC[0]]
230 AB=ADR[B[0]]
240 REM
250 DATA 0420H,06180H,0D000H,01601H
260 DATA 0380H,0460H,06550H,0
270 REM
280 FOR X=0 TO 12 STEP 2
290 READ Q
300 MWD[AMC+X]=Q
310 NEXT X
320 REM
330 INPUT £19"Name ";$NM[0]
340 REM
350 CALL AMC,0,DRV*256,0,AB,64
360 IF MWD[AB]: GOTO 430
370 FOR X=0 TO 19
380 MEM[AB+X+2]=MEM[ADR[$NM[0]]+X]
390 NEXT X
400 CALL AMC,0FFH,DRV*256,0,AB,64
410 ? "Ok."
420 END
430 ? " Can't name a SYSTEM Disk"
440 STOP
```

DISK VERIFY PROGRAMME BY C.J.YOUNG

verifies disk and flags any faulty sectors

```
10 REM
20 REM *****
30 REM * *
40 REM * DISK *
50 REM * *
60 REM * Verify *
70 REM * *
80 REM *****
90 REM
100 TEXT
110 ? "CDOS Disk Verify program"
120 INPUT £1;"Drive ",DRV
130 DRV=MOD[DRV,4]
140 ?
150 REM ***
160 REM *** Machine code ***
170 REM ***
180 DIM MC[4]
190 AMC=ADR[MC[0]]
200 X=0
210 READ Q
220 IF Q=0: GOTO 270
230 MWD[AMC+X]=Q
240 X=X+2
250 GOTO 210
260 DATA 0420H,06180H,05C5H,0585H,0D540H,0380H,0
270 REM ***
280 REM *** Get disc pointers ***
290 REM ***
300 P1=MWD[06362H+DRV*2] !Pointer 1
310 P2=MWD[06372H+DRV*2] !Pointer 2
320 P3=MWD[06382H+DRV*2] !Pointer 3
330 REM ***
340 REM *** Get infomation ***
350 REM ***
360 BPS=P1 ! Bytes per sector
370 SID=MWD[P2+6] !Sides
380 SPT=MWD[P3+0] !Sectors per track
390 SEC=MWD[P3+2] !Sectors
400 SOD=MWD[P3+4] !Start Of Dir
410 FIL=MWD[P3+6] !Files
420 REM ***
430 REM *** Calc ***
440 REM ***
450 TRK=SEC/SPT !Tracks
460 BPT=BPS*SPT !Bytes per track
470 BYT=BPS*SEC !Bytes
480 REM ***
490 REM *** Display infomation ***
500 REM ***
```

```

510 ? "Bytes per sector ";BPS
520 ? "Sectors per track ";SPT
530 ? "Tracks ";TRK
540 ? "Number Of Files ";FIL
550 ?
560 REM ***
570 REM *** Buffers ***
580 REM ***
590 BLN=INT[BPS/6]+1 !Buffer length
600 DIM BFT[BLN*16] ! Track Buffer
610 DIM BFS[BLN] ! Sector Buffer
620 DIM MAP[BPS] ! Bit map
630 DIM BDR[64*FIL/6+1] !Dir Buffer
640 DIM $F[2] ! File Name
650 REM ***
660 REM *** Pointers ***
670 REM ***
680 ABT=ADR[BFT[0]]
690 ABF=ADR[BFS[0]]
700 AMP=ADR[MAP[0]]
710 AD=ADR[BDR[0]]
720 ERR=0
730 ERF=1 !Error Flag
740 ERS=0 !No Of Sector Errors
750 ERW=0 !No Of Write Errors
760 ERE=0 !No Of Read Errors
770 AE=ADR[ERR]
780 CAL=0
790 W1=ADR[CAL]+2
800 W2=ADR[CAL]+4
810 MX=65536
820 REM ***
830 REM *** read bit map ***
840 REM ***
850 T=1
860 S=0
870 GOSUB 1150
880 CALL AMC,0,C1,C2,AMP,SEC/8,AE
890 IF ERR: ? "Can't read bit map ": STOP
900 REM ***
910 REM *** Read Directory ***
920 REM ***
930 S=SOD-16
940 GOSUB 1150
950 CALL AMC,0,C1,C2,AD,FIL*64,AE
960 IF ERR: ? "Can't read Directory"
970 REM ***
980 REM *** Track test ***
990 REM ***
1000 FOR T=0 TO TRK-1
1010 S=0
1020 GOSUB 1150
1030 CALL AMC,0,C1,C2,ABT,BPT,AE
1040 IF ERR: GOSUB 1230
1050 NEXT T

```

```

1060 IF ERF: ? "No Errors": STOP
1070 ?
1080 ? £'999'ERS;" Sector Error";
1090 IF ERS<>1: ? "s";
1100 ? " ";
1110 ? £'999'ERW;" Write Error";
1120 IF ERW<>1: ? "s";
1130 ?
1140 END
1150 REM ***
1160 REM *** Calc ***
1170 REM ***
1180 CAL=(16*MX+DRV*256)*MX
1190 CAL=CAL+T*BPT+S*BPS
1200 C1=MWD[W1]
1210 C2=MWD[W2]
1220 RETURN
1230 REM ***
1240 REM *** Track Error ***
1250 REM ***
1260 ERF=0
1270 FOR S=0 TO SPT-1
1280 GOSUB 1150
1290 Q2=T*SPT+S
1300 SIU=BIT[MAP[0],Q2] !In use ?
1310 CALL AMC,0,C1,C2,ABF,BPS,AE
1320 IF ERR: GOSUB 1510
1330 ELSE GOSUB 1360
1340 NEXT S
1350 RETURN
1360 REM ***
1370 REM *** Write sector ***
1380 REM ***
1390 CALL AMC,255,C1,C2,ABF,BPS,AE
1400 IF ERR: GOTO 1420
1410 RETURN
1420 REM ***
1430 REM *** Write Error ***
1440 REM ***
1450 ERS=ERS+1
1460 ERW=ERW+1
1470 GOSUB 1790
1480 ? "Write Error";
1490 GOSUB 1840
1500 RETURN
1510 REM ***
1520 REM *** Read Error-Try Write ***
1530 REM ***
1540 ERS=ERS+1
1550 IF SIU: GOTO 1620
1560 FOR Q=0 TO BPS-1
1570 MEM[ABF+Q]=0
1580 NEXT Q

```

```

1590 CALL AMC,255,C1,C2,ABF,BPS,AE
1600 IF ERR: GOTO 1700
1610 RETURN
1620 REM ***
1630 REM *** Read Error ***
1640 REM ***
1650 ERE=ERE+1
1660 GOSUB 1790
1670 ? "Read Error";
1680 GOSUB 1840
1690 RETURN
1700 REM ***
1710 REM *** Read/Write Error ***
1720 REM ***
1730 ERW=ERW+1
1740 ERE=ERE+1
1750 GOSUB 1790
1760 ? "Read & Write Error";
1770 GOSUB 1840
1780 RETURN
1790 REM ***
1800 REM *** Track/sector number ***
1810 REM ***
1820 ? £'999'T;"/";£'99'S;" ";
1830 RETURN
1840 REM ***
1850 REM *** File Name ***
1860 REM ***
1865 ZZ=SOD+(FIL*64/BPS)
1870 IF T*16+S<ZZ: GOTO 1990
1880 FOR Z=0 TO FIL-1
1890   ZZ=AD+Z*64
1900   IF MWD[ZZ]=0: GOTO 1960
1910   FOR Z1=ZZ+16 TO ZZ+31 STEP 4
1920     IF T*16+S<MWD[Z1]: GOTO 1950
1930     Z2=MWD[Z1]+MWD[Z1+2]
1940     IF T*16+S<Z2: GOTO 2010
1950   NEXT Z1
1960 NEXT Z
1970 ?
1980 GOTO 2000
1990 ? " In Directory"
2000 RETURN
2010 $F[0]=" "
2020 FOR II=2 TO 10
2030   $F[0]=$F[0]+%MEM[ZZ+II]%0
2040 NEXT II
2050 ? " In File:"+$F[0]
2060 RETURN

```

MDEX software for the Cortex.

MDEX (Marinchip Disk Executive) is a disk operating system similar in some respects to CPM. It was originally developed for the T.I. range of computer boards using the TMS9900 processor. It has been modified by M.P.E. in England for use on the Cortex. M.P.E. no longer wish to support MDEX and have handed the system to the Cortex User Group. The Group have been unable to trace Marinchip in the States and are therefor offering the system to Cortex User Group members at a much reduced price, 10% of its original value, assuming no copywrite is payable.

The system available.

MDEX CORE :- £10.00

The operating system includes a Boot disk for loading the nucleus which contains all input output drivers and file handling software. Also included on the system disk are file and disk maintenance utilities, a debug monitor, file copy utilities with wild-card facilities and a disc BASIC similar in many respects to Microsoft Basic. Also included is a simple line orientated context editor EDIT similar to CPM's ED.

ASM & LINK :- Assembler and Linker £10.00

ASM is a relocatable assembler which supports the full instruction set of the TMS9995. It accepts standard TI mnemonics in a form essentially compatible with TI assemblers, and produces a TI standard tag format output file. Conditional assembly is also supported, as is comprehensive expressive evaluation at assembly time. ASM is very fast assembly speed being over 600 lines per minute.

LINK allows several object files generated by ASM or a compiler to be linked together to form an executable programme. Programmes of any size can be built.

SYSGEN :- System generation Kit £10.00

This is a toolkit for modifying MDEX itself and will be needed if you want to reconfigure the terminal, printer, or disk handlers, or if you want to install MDEX for different hardware. Sysgen includes all the source listings for hardware dependant software.

WORD :- Word processor £10.00

Word is a powerfull text processing programme. It takes text from a file produced with EDIT or WINDOW, and produces a neatly formatted document. It fetures user control of all formatting parameters, macros with parameters, library files, user defined stings, automatic section numbering / page numbering, index, headings and footings etc.

MDEX-PDS :- All of the above systems in one package £30.00

SPL :- System programming language £10.00

A systems programming language much in the style of "C". Allows in line assembler and many other tricks needed for systems or industrial programming.

FORTH :- £15.00

Fig forth with file handling extensions, four editors, full 9995 assembler, over 200 screens of source code, decompiler, turn-key compiler etc.

NAUTILUS :- Forth cross compiler £10.00

Compiles Forth systems to varying target hardware configurations. Allows use of multiple source files. This software can produce fully ROM-able code for a range of microprocessors. Chose targets from 6502, 6800, 8080, Z80, 8086/88, 68000, 1802, Z8, 6301/6801, 9900/9995 etc. Targets £10.00 each.

META :- Compiler generator £10.00

An automatic compiler generator. Feed it a description of language and details of the code generating rules and out comes a compiler for the language. META can be used to generate macro-processors, assemblers, compilers and brain strain.

QBASIC :- £15.00

A structured compiler Basic wich is fully code compatible with the popular CBASIC for CPM. Has 31 character names, WHILE-WEND, formatted output, line numbers only required on refered to ststatement lines etc. Output file can be linked with other files produced by QBASIC or ASM.

PASCAL :- £10.00

An adaption of the Per Brinch Hansel sequential pascal.

WINDOW :- £15.00

A fully featured screen editor, incorporating on-line help facilities, a virtual memory system memory system for large files optional automatic indentation and word wrap, global search and replace facilities, horizontal scrolling and some text formatting functions. Very easy to use. This is a superb editor.

SPELL :- £10.00

A spelling checker that can be used with any editor or text file. It comes complete with a large dictionary can be added to by the user.

All the above MDEX software is now available from the Cortex User Group at the normal address.

Qbasic is a true compiler for basic, which runs on the MDEX operating system. It is very fast and lends its self to structured programming. According to the manual it is compatible to CBASIC, i've yet to find out.

The QBASIC disk contains the compiler, plus the QBASIC library which is a set of routines used by QBASIC to form the final executable programme. With interpretive Basic it is resident in the computers memory thus using memory plus the executing programme. By using the Qbasic library QBASIC links only what's needed making a more compact programme.

A feature of QBASIC is the ability to link several QBASIC &/or assembly programmes together into an executable programme, therefore you can have your own set of standard programmes stored on disk to use when designing your new pr

I will describe how to compile your own programmes & the language syntax.

The QBASIC disk is in drive 1 and the work disk is in drive 2. First, three files have to be created on drive 2. If we call our programme TEST then our files will be:-

TEST.BAS	this will hold the source programme
TEST.REL	relocatable code used by the linker
TEST	the executable programme.

Note the suffix .BAS & .REL

Once the executable programme TEST has been compiled it can be transferred, if required to drive 1, to run the programme just type in the name & it will run.

The sequence of events is as follows:-

The file TEST.BAS would be put into one or Edit, you would then write your QBASIC programme into it & save the file, the compiler would be called, ie QBASIC 2/TEST. (the prefix 2/ denotes drive 2/ right/h) the compiler would then write code to TEST.REL. Next you would link the programme, ie QLINKER 2/TEST which would take the file 2/TEST.REL & pick up the necessary library routines plus other QBASIC or assembly programmes & write the executable output to the file 2/TEST. The system linker will then produce a memory map for the programme 2/TEST & exit to MDEX. Simply type in the programme name and the programme will be executed.

Programme Syntax.

Line numbers need not be used except on GOSUB or GOTO which due to structured nature of QBASIC are not used often. If programme lines are very long they may be continued into the next line by the addition of a backslash "\".

The remark statement has several forms, but the most interesting is embedded comments. First a left curly bracket is printed "{" then every thing between it and the right curly bracket "}" is treated as a remark or embedded comment, it can be in the middle of a statement line or continue for several lines. Neither form of comment occupies space or effect the speed in the executable programme, in effect the compiler takes no notice of them so QBASIC encourages lots of remarks !!!

The other interesting feature is the block IF statement.

```
eg:- IF <expression>
      code0
      code0
ELSEIF
      code1
      code1
ELSEIF
      code2
      code2
ELSE
      code3
ENDIF
```

When the block IF statement is executed, the <expression> is evaluated according to the same rules as a single line IF THEN if the value is a non zero the code following the IF statement will be executed upto the ENDIF, if zero the code will be skipped, until an ELSEIF or ELSE or ENDIF is reached. Therefore programmes can be structured very easily.

Variables are of two types and can 31 characters long

Real variables	A	AC	PAYROLL	MAIN.LOOP.COUNTER
Integer variable	A%	AC%	PAYROLL%	MAIN.LOOP.COUNTER%
String variable	A\$	AC\$	PAYROLL\$	MAIN.LOOP.COUNTER\$

Also all the above may be subscripted variables.

Sample pgm.

{Read the keyboard using
the Qbasic library routines
Constat% Conchar%, which bypasses
the normal keyboard INPUT
routine.}

```
1      WHILE NOT CONSTAT%  {Constat%= -1 if keyboard is pressed}
      WEND
      C%=CONCHAR%          {Conchar%=ASCII val of key}
                           {it is stored in C%      }

      IF C%=65             {A pressed ?}
      PRINT "Please continue"
      A.FLAG%=1:R=10
      SCREEN.DISPLAY$="MENU-0"
      GOTO 100
    ELSEIF C%=66           {B}
      SCREEN.DISPLAY$="MENU-2"
      GOTO 100
    ELSEIF C%=67           {C}
      SCREEN.DISPLAY$="MENU-1"
      GOTO 100
    ENDIF

      PRINT CHR$(8);" ";CHR$(8)      {If A,B or c not pressed }
      GOTO 1                          {backstep on input and
                                       return to start  }

100     PRINT CHR$(26);SCREEN.DISPLAY$;\
        R;A.FLAG%

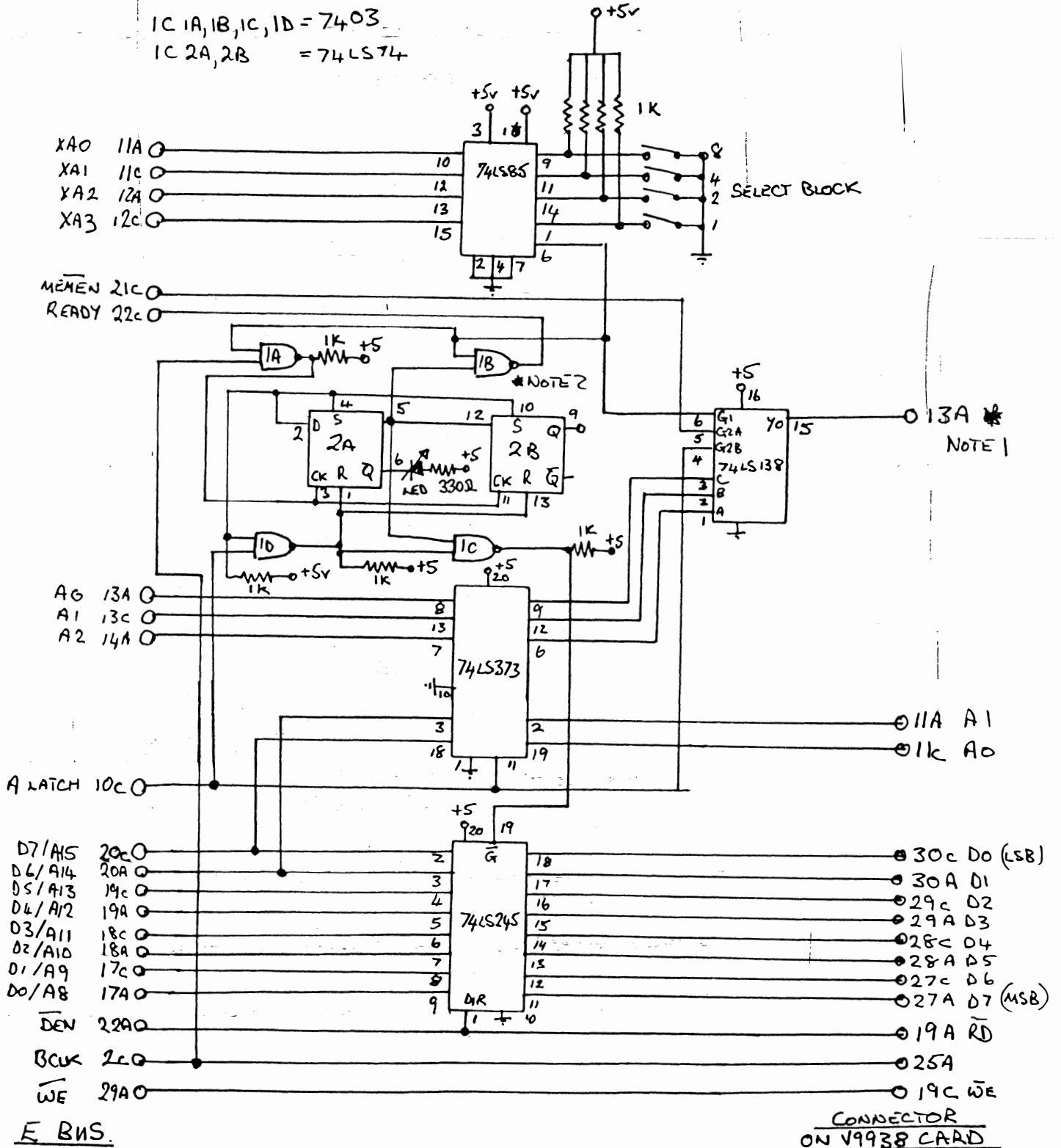
END
```

In the next article i will describe some of the file handling statements used in QBASIC.

80 COLUMN SCREEN FOR THE CORTEX

RECENTLY MAPLIN (THE COMPONENT SUPPLIERS) HAVE PRODUCED A KIT FOR A FRAME STORE SEE MAPLIN MAGAZINE SEPTEMBER TO NOVEMBER 1987 ISSUE

IN THIS KIT THEY USE A V9938 VDP CHIP WHICH IS SEMI COMPATIBLE WITH THE CORTEX VDP BUT WILL GIVE 80 COLUMN ON SCREEN. IF YOU PURCHASE THE KIT AND BUILD IT UP ON THE MAPLIN PCB THE CIRCUIT DIAGRAM WILL SHOW YOU HOW TO INTERFACE IT TO THE CORTEX E BUS. FURTHER ARTICLES WILL FOLLOW ON HOW TO PROGRAM THE NEW VDP FROM THE CORTEX



NOTE 1 * FIT LINK 7 ON I/O BLOCK ON V9938 CARD

NOTE 2 STRAPPED UP FOR I WAIT STATE

EXTENDING E BUS MEMORY

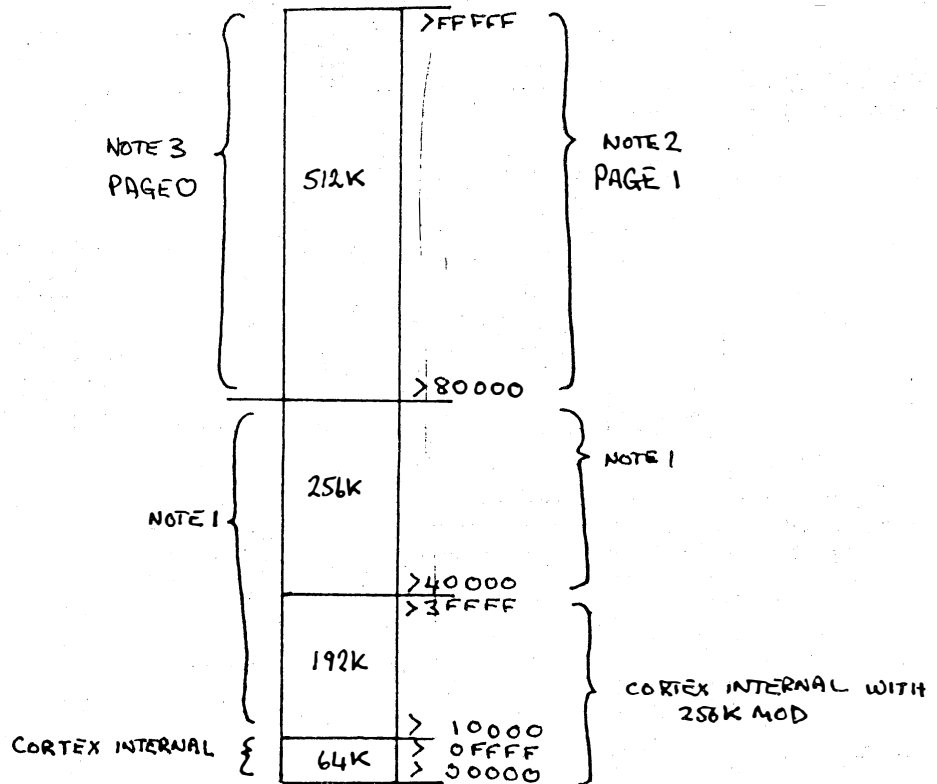
THIS ARTICLE DEALS WITH EXTENDING THE ADDRESS SPACE ON THE E BUS TO 2 M/B AND HOW TO MODIFY MEMORY CARDS TO USE THE EXTRA ADDRESS LINE

1 CORTEX MAIN BOARD MODIFICATION

- (a) CONNECT PIN 1 AND PIN 19 OF IC 94(74LS244)
- (b) CONNECT PIN 17 OF IC 94 TO PIN 12 OF IC 64 (THIS IS A SPARE CRU OUTPUT BIT)
- (c) CONNECT PIN 3 OF IC 94 TO E BUS CONNECTOR PIN 9C

THIS IS ALL THAT IS REQUIRED TO BE DONE ON THE CORTEX MAIN BOARD

E BUS MEMORY MAP USE



- NOTE 1 FOR USE WITH EXPANSION CARDS THAT DO NOT DECODE EXTRA ADDRESS LINE
- NOTE 2 512K MEMORY PAGE 1 FOR USE WITH EXPANSION CARDS THAT HAVE EXTRA LINE DECODE
- NOTE 3 512K MEMORY PAGE 0 DEFAULT BLOCK ON SWITCH ON

MODIFICATION TO MEMORY CARDS

2 MODIFICATION TO MPE MEMORY CARDS 'MOD 3' TO PAGE IN AT PAGE 1
NOTE-THE ORIGINAL DIAGRAM FOR THIS CARD IS INCORRECT SO MODIFY
AS FOLLOWS

- (a) CUT TRACK LEADING TO G2 LINK FROM E BUS CONNECTOR 6C
- (b) CUT TRACK BETWEEN E BUS CONNECTOR 12A AND IC 1 PIN 3
- (c) CUT TRACK BETWEEN IC 1 PIN 4 AND IC 4 PIN 6
- (d) CONNECT E BUS CONNECTOR 9C TO IC 1 PIN 3
- (e) CONNECT IC 1 PIN 4 TO G2 LINK
- (f) INSTAL G2 LINK

3 MODIFICATION TO MPE MEMORY CARDS 'MOD 4' TO PAGE IN AT PAGE 0

- (a) CUT TRACK BETWEEN E BUS CONNECTOR 6C AND G2 LINK
- (b) CONNECT E BUS CONNECTOR 9C TO G2 LINK
- (c) INSTAL G2 LINK

4 MODIFICATION TO TM716 MEMORY CARD(CORTEX USERS CLUB CARD) 'MOD 1' TO
PAGE IN AT PAGE 1

- (a) CUT TRACK BETWEEN E BUS CONNECTOR 11A AND PIN 9 OF IC 2 (74LS04)
- (b) CONNECT PIN 9 OF IC 2(74LS04) TO E BUS CONNECTOR 9C
- (c) CUT TRACK BETWEEN IC 2(74LS04) PIN 2 AND PIN 13 OF IC 1(74LS00)
- (d) CONNECT PIN 13 OF IC 1(74LS00) WITH 1K PULL UP RESISTOR TO
+ SUPPLY
- (e) CONNECT PIN 13 OF IC 1(74LS00)WITH DIODE TO PIN 2 OF IC 2
(74LS04) THE POSATIVE END OF THE DIODE TO IC 2
- (f) CONNECT PIN 13 OF IC 1(74LS00)WITH A SECOND DIODE TO PIN 8
OF IC 2(74LS04)THE POSATIVE END OF THE DIODE TO IC 2

5 MODIFICATION TO TM716 MEMORY CARD 'MOD 2' TO PAGE IN AT PAGE 0

USE STEPS 3c , 3d , 3e AS ABOVE

- (g) CONNECT WITH DIODE E BUS CONNECTOR 9C TO PIN 13 OF IC 1(74LS00)
THE POSATIVE END OF THE DIODE TO E BUS CONNECTOR 9C

HOW TO USE EXTENDED ADDRESS ON E BUS

WHEN THE CORTEX IS FIRST POWERED UP PAGE 0 IS THE ENABLED . TO
SWITCH TO PAGE 1 USE

```
LI R12,>0000
SBO 7
B @>0080(RETURN)
```

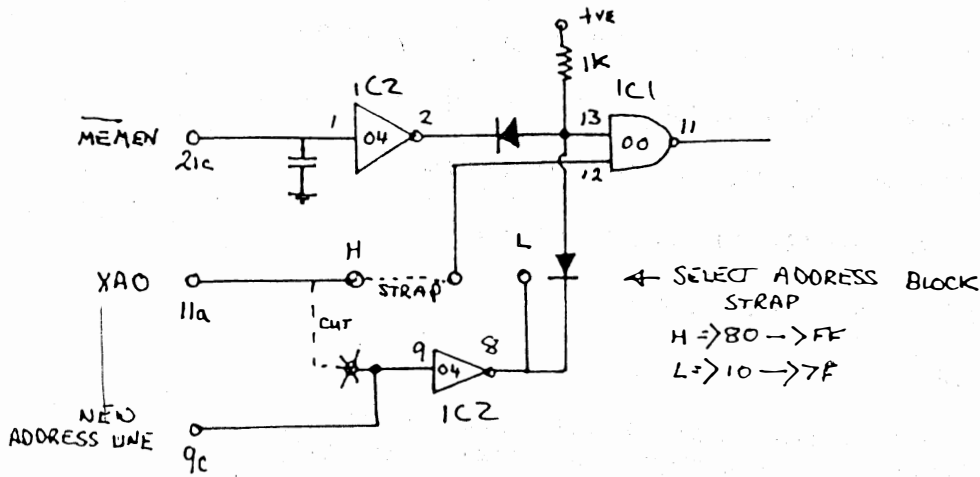
```
IN BASIC      100 BASE 0
              110 CRB(7)=1
```

TO SWITCH BACK TO PAGE 0 USE

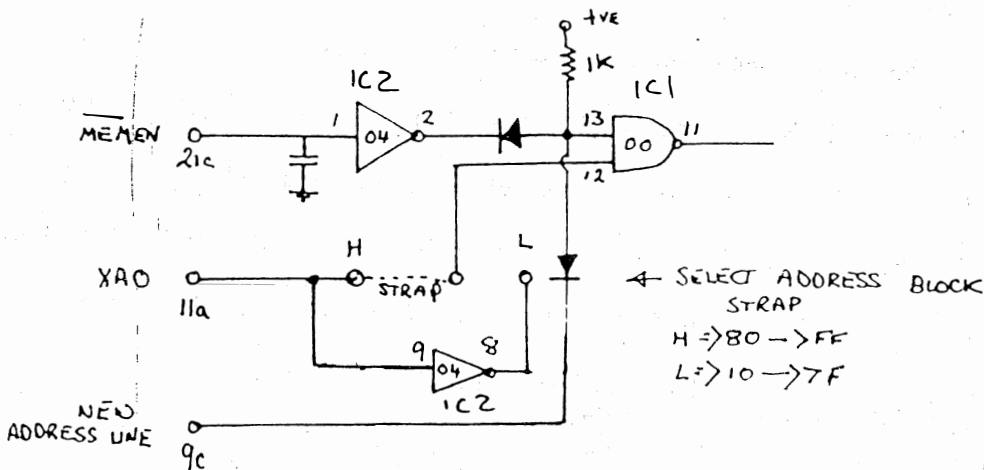
```
LI R12,>0000
SBZ 7
B @>0080(RETURN)
```

```
IN BASIC      100 BASE 0
              110 CRB(7)=0
```

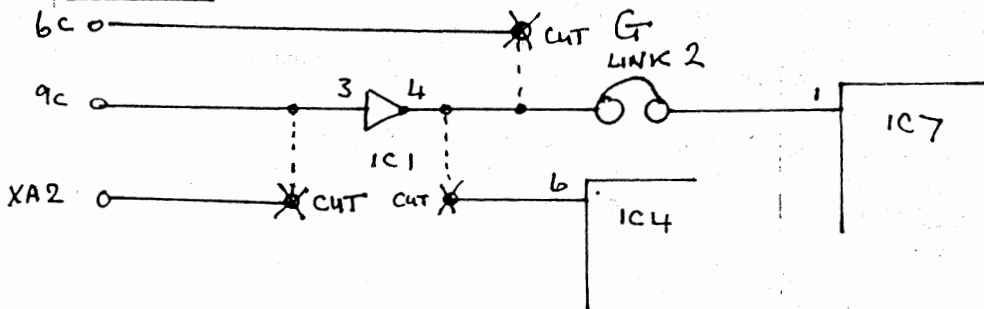
MOD 1 (TM716)



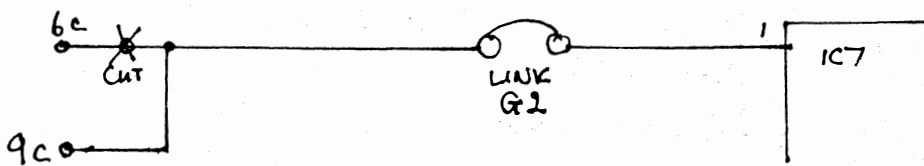
MOD 2 TM716



MOD 3 (MPE)



MOD 4 (MPE)



UNIVERSAL MOUSE AND KEYBOARD INTERFACE

THE CIRCUIT DIAGRAM FOLLOWING CAN BE BUILT UP ON A EUROCARD SIZE BREAD BOARD . IF A 8 BIT MOUSE INTERFACE IS REQUIRED USE IC'S 1,2,3,5,7,8,11,12,15,16,17,18,19,20

IF A KEYBOARD INTERFACE IS REQUIRED USE IC'S 1,2,20 . THE INTERFACE HAS THE FACILITY TO RESET THE COUNTERS TO A KNOWN VALUE , EITHER BY THE COMPUTER RESET LINE OR BUTTON THREE ON THE MOUSE . THIS FACILITY IS SELECTED BY LINKS 1 AND 2

IF A KEYBOARD IS USED THIS CAN BE A 7 BIT OR 8 BIT PARALLEL OUTPUT DEVICE . IF A 7 BIT KEYBOARD IS USED THE 8 BIT CAN BE SET LOW WITH LINK 3 THE FOLLOWING IS A M/C EXAMPLE ON HOW TO ACCESS THE CIRCUIT

```

LI R12,>1000
STCR R1,0      16 BIT VALUE
OR STCR R1,8   8 BIT VALUE
LI R12,>1020
STCR R2,0      16 BIT VALUE
OR STCR R2,8   8 BIT VALUE
LI R12,>1040    BASE ADDRESS OF KEYBOARD
STCR R3,8      8 BIT VALUE
OR STCR R3,7   7 BIT VALUE
OR STCR R3,3   3 BIT MOUSE KEYS
RT             RETURN
    
```

OTHER USES FOR THE CIRCUIT CAN BE FOR ANY POSITIONAL READING AS IN CONTROL OF SATELITE DISH , DRILL BED CONTROL , TURTLE CONTROL
 SEVERAL OF THESE COUNTERS CAN BE BUILT AND USED IN A ROBOT TO DO ALL YOUR HOUSEWORK OVER CHRISTMAS OR EVEN CONTROL YOUR HI-FI FROM YOUR CORTEX
 IF YOU DO NOT WISH TO BUILD THIS CIRCUIT TEXAS INSTRUMENTS HAVE PRODUCED A NEW IC WHICH WILL DO THE SAME JOB , SEE BELOW ---

'Smart Parts'™ from TI ASIC

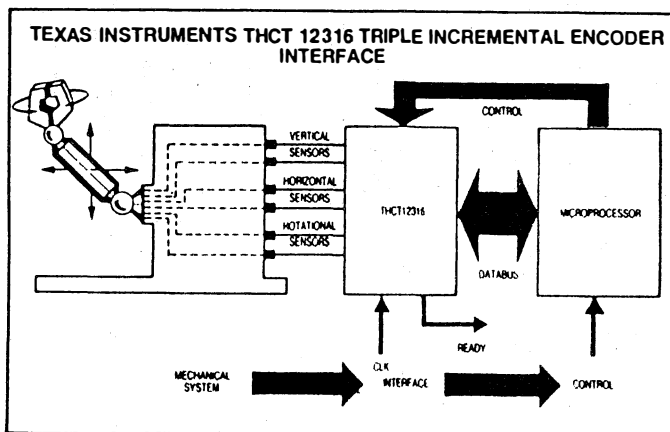
Smart Parts™ are designed to provide multi-application solutions where no standard part exists. If a smart part™ is in heavy demand it will become a TI standard product with full characterisation and qualification data.

These solutions are facilitated by TI's ASIC technology where a wide variety of SN 74/54 functions are already coded as cells in the 3, 2 and 1 micron libraries. These provide the ideal building blocks for the design of single chip solutions to widely used groups of logic.

The best illustration of a Smart Part™ is the THCT 2000, an incremental encoder interface already widely used across Europe in a number of industrial control applications. This single device replaces 14 TTL

devices and is configurable in several modes so as to allow:

- Up/down counter.
 - Direction discrimination.
 - Pulse width measurement.
 - And frequency measurement functions.
- Implemented in one of the single level metal HCMOS gate arrays, the THCT 2000 has been in production for the last three years.
- Future Smart Parts™ include:
- "3dimensional" incremental encode interface
 - CMOS line drivers.
 - Micro processor peripherals, e.g. timer.



TEXAS INSTRUMENTS

Texas Instruments Ltd
 Manton Lane
 Bedford
 MK41 7PA

Tel: (0234) 270111
 Telex: 82178
 Technical Enquiry
 Service
 Tel: (0234) 223000

APOLOGY

WE MUST APOLOGISE TO MR R M LEE FOR NOT PRINTING THE CORRECT ADDRESS OR CORRECT PRICE FOR HIS SOFTWARE THIS HAS NOW BEEN CORRECTED . IF ANY OTHER READERS HAVE A CHANGE OF ADDRESS PLEASE SEND THE ALTERATION TO ME (E. SERWA)

UNIVERSAL MOUSE/EXTERNAL KEYBOARD INTERFACE

