CORTEX USER GROUP

NEWS LETTER II FEBRUARY 1985

Firstly, an apology!

We had every intention of releasing this, our second News Letter, just before Christmas. Alright! so we don't have any excuses but, what with the demand for CORTEXES and the flood of reader's letters, it has taken us this long to reach publication. Hope you think it's worth waiting for!

We intend to become more active over our contacts with our users and are aiming to produce a regular, **quarterly** News Letter from now on. Probably the most important part of this publication is the contributions we receive from you, the user. Please keep your programmes, queries, new ideas and criticisms coming in. We need them - for you.

You will note this News Letter is considerably larger than our first and, because of the extra work involved in producing four issues per year, we will have to charge a nominal annual sum for you to become registered with the User Group. This will be £5 p.a. which will just about cover our admin. costs.

We learn - finally! - that Texas have now discontinued the 9909 (the floppy disc controller chip). As an alternative, they are offering a new plug-in module and this is due now. Naturally, we shall charge no extra for this module to those of you who have been waiting for delivery of your CORTEX. We look forward to clearing all outstanding orders as soon as Texas deliver to us.

Secondly the advertisements:

CDOS is available, on single or double-density disc. You'll find it well worth the £43 + VAT (no postage). Please state whether CDOS is required on 40 track or 80 track; single-sided or double-sided disc. You may have seen the review of CORTEX II in the December copy of PCN. For those that missed it, we attach a copy of the article. Reviews being what they are, it's fairly complementary about us!

For a while, the price of CORTEX II is being held at £299 + VAT. We cannot keep this forever, so get your friends to send their orders pretty quick!

GAMES COMPETITION

We had a good response to our Competition and they were generally of a very high quality. This made judging them very difficult! The names of the hard-working contributors is listed below, together with the prizes they will receive.

FIRST PRIZE goes to Robert Lee, of Eastbourne

who wrote MAZE. He will be receiving £20.

SECOND PRIZE goes to Gary Alexander, of BRISTOL

who wrote G Design. He will be receiving £10.

If we had a third prize, I think it would have gone to Mr R Green, of Rotherham who sent us WINE and FORM 1. Their use by our readers might be limited but the programmes deserve a recommendation for both interest and originality.

We are thinking of having another competition later in the year but the subject is being kept secret for now.

Partly as a result of the Competition and also from your many letters, we have a good variety of GAMES for you to buy. These are on sale at £6 + VAT each, with a £1.50 royalty sent to the originator on each one. They are described, briefly in the paragraphs below.

We can reach a wide readership with this News Letter and will manage all the overheads, eg. paperwork, tape copying, etc. Keep your games rolling in and lets see how they sell!

** Some of the tapes you send us will not load - or only with difficulty. Do try to check them before sending ** Thanks.

Wine Ideal for the non-technical winemaker. This prog. asks what you want to make - table/desert, colour, quantity, etc. and when you've filled in the answers, including the type of fruit, it tells you just what correcting additions you need, ie Pectozyme, malic acid and so on.

Form 1 This goes with WINE and prints out a standard form for your records - you'll be surprised how many notes you accumulate over the years and this will help.

Burglar The object of the game is to manoeuvre a little man around the screen - avoiding obstacles, jumping off ledges, missing poisonous bushes, etc., etc. This is in order to collect five keys, which will give you access to the next screen. There are plenty of screens (I lost count!) - each getting more difficult. The graphics are very good and the keys are fast and easy to use.

Muncher Another variation of PACMAN! The graphics are good but I found the keys a little awkward to find quickly enough and never managed a full screen. It is a little annoying to keep pressing the keys in order to move continuously.

G Design This enables characters and shapes to be designed on the screen. The computer then generates the correct numbers to form the character or shape. It also works in reverse, giving you the character for the numbers entered. A very useful tool for graphics generation.

3D Graph This will draw 3 D graphs of functions already entered and should be able to draw a user function as well. Unfortunately, I was unable to get it to accept any of my functions!

Cortello A screen version of the board game OTHELLO and it is excellent if you have no one else to play with. I found it easy to beat the lower two levels but ran out of time to press my luck further.

Night Attack The screen presents the New York skyline, with the Empire State building in the centre and the statue of Liberty in the centre foreground. The city is under attack by a spacecraft, which drops bombs onto the skyscrapers - causing visible damage. Liberty's torch can be pointed towards the spacecraft and/or the bombs and each success is rewarded by marks - most marks, naturally, for destroying the spacecraft. You're in trouble if a bomb hits the statue!!

Wall This was a little tame compared with some of today's arcade games. Even so, it is always a challenge to beat the computer - which I didn't! The object is to bomb a wall down, where your aircraft flies back and forth across the wall - losing both height and time. You must demolish the wall before you hit it.

Maze This game appealed to me! The computer generates a maze at one of five levels of difficulty. You are placed in the middle of the maze and are given a 3 D view from where you stand. The aim is to find a food chest but to try to visualise the maze, by looking forward, right & left and behind you is mind blowing and a great challenge. Very enjoyable.

NOTE. A minor hiccup with this issue.

You will find a lot of '%'s in some of the listings. These should, of course, be '@'!

Your Own Pages .

The following letters and programs have been sent in by CORTEX Users. Powertran Cybernetics Limited cannot accept liability for their contents.

Graham Thirsk
Pertenhall Beds

Remember Robert Lee's Terminal Emulator programme, published in News Letter 1? Well, Graham's been doing some work on it and we offer his comments.

TERMINAL EMULATOR PROGRAM EXTENSION

The program below allows full control of the RS 232 port or Cassette port for the Terminal Emulator presented by R M Lee.

```
XXX       LIMI  0                        ! disable all interupts
          L1 R12, > 080                  ! set CRU address RS232
                                           (0180 for cassette)
          SBO 31                         ! Reset UART
          SWPB RO                          Select correct byte
          LDCR RO,8                        Load control register
          SBZ  13                          Ignore Interval register
         *LDCR R1,11                       Load receive data rate
        **LDCR R2,12                       Load Xmit data rate
YYY                                       EMULATOR PROGRAM CONTINUES
```

The process should be repeated using CRU address 0180H and R3, R4, R5 for the parameters if cassette port requires modification.

Omitting line * and making line ** LDCR R1, 12 makes the baud rate for Xmit and receive the same i.e. R1.

The routine for RS 232 is called by.

Call XXX, control data, RCV rate, Xmit rate.

Obviously these parameters may be included in the MC program so it may be Auto Run after loading.

CONTROL VALUES

a) BIT COUNT

7 bits	= 02H
8 bits	= 03H

PARITY. Add Values to those selected for Bit Count

b)

No PARITY	010H
Even "	020H
Odd "	030H

STOP BITS Add to values selected

c)

11/2	STOP	0H
2	STOP	040H
1	STOP	080H

eg. 8 bits No parity 2 stop bits = 063H

BAUD RATE - parameters given by

DR (division ratio) = 500,000/Baud rate required
if DR > 03FFH then DR = DR/8 + 400H

e.g. 1200 BAUD = 1A1H
300 BAUD = 4DOH
75 BAUD = 741H

The parameters preset by CORTEX BASIC for the cassette Port (SAVE + LOAD) may be modified in the same way.

MEM (OIC8H) contains the control data for the cassette (1BYTE) (043H)

MWD (18B2H) contains the Division Ratio for the cassette

MEM (5546H) contains the control data for the RS 232
(1BYTE) (062H)

BAUD RATES for Devices set by UNIT are best set by the BAUD command !

--- xxx ---

Helge A Larsen
NORWAY

Helge has sent us a short assembly routine for use in arcade type games. She apologises for her English (which is a lot better than my Norwegian!)

"I have tried to write some games for my children and found that using the KEY-function to manoeuver an object across the screen is not a good idea. You have to push the Repeat button to get a continuous movement whilst you are pushing the buttons and when you then release the button, the movements continues. To overcome this I have written a short assembly programme but the problem is not fully solved. Because by reading the values from the keyboard you will have a lot of random numbers while the keys are not operated. I don't think this is a great problem in games and could even be useful.

"Maybe some people are annoyed at bringing up games in this group but I find that the computer is very valuable as joy also and more serious users may also find useful hints in this programme. As can be seen from the printout, the programme is thought to be used with a CALL and two parameters , the adr. of x and y. First R0 and R1 are incremented and saved. They are used in the routine at 15FAH. This routine also uses R2. The values from the keyboard are then read into R0 and 14 FAH then makes branches depending on the value in R0.

"On the addresses from 6018H we have a table with the displacement to jump in high order byte and the value of the arrow keys in low order byte. Please note that the table has to be finished with 0. To calculate the address to be jumped to use the formula 'next adr+2*disp.'

An example:
The left arrow is pushed. This key has the value 8. You find it at 6018H.
The displacement is 6 and the jump is to 6024H.

"In this program I have used the four cursor keys but a fire button may easy be included in the table. Of course new displacements have to be calculated and a suitable routine written."

```
UNIT-2
MON
Monitor Rev. 1.1 1982
[]U 6000 6032

6000 05C0 INCT R0
6002 05C1 INCT R1
6004 C140 MOV R0, R5
```

```
6006 C101 MOV R1,R4
6008 020C L1 R12,>0010
600C 3400 STCR R0, 0
600E 0240 ANDI R0, >00FF
6012 0A80 SLA R0, 8
6014 06A0 BL ‰>15FA
6018 0608 DEC R8
601A 0809 SRA R9,0
601C 0A0B SLA R11,0
601E 0C0A DATA >0C0A
6020 0000 DATA >0000
6022 0380 RTWP
6024 0615 DEC *R5
6026 1001 JMP >602A
6028 0595 INC *R5
602A 0380 RTWP
602C 0614 DEC *R4
602E 1001 JMP >6032
6030 0594 INC *R4
6032 0380 RTWP
[]G

CORTEX BASIC Rev. 1.1 1982
*Ready
UNIT -2
LIST

10  X=128: Y=96
20  CALL 0600H, ADR(X), ADR(Y)
30  IF X>255 THEN X = 255
40  IF X<0 THEN X = 0
50  IF Y<0 THEN Y = 0
60  IF Y>191 THEN Y = 191
70  PLOT X,Y
80  GOTO 20
```

--- xxx ---

J H Taylor
Co. Down

J H has sent us a listing for a BASIC sprite designing program. Try it.

To use a sprite at line 50 ...

```
50 SPRITE No, X,Y, shape no, colour
     where No is 0 to 31 , using lower numbers first.
     X is 0 to 255, Y is 0 to 191, origin top LH corner.
     colour is 1 to 15.
```

A shape must be defined first and it is

```
20 SHAPE No, 1st,2nd,3rd,4th
     where 1st to 4th are 16 bit binary numbers showing whether a
     pixel (picture element) is "on" (1), or "off" (0).
```

Luckily these binary numbers are expressed in Hexadecimal or, better still in Decimal (as used here).

The program presents a square frame for 8x8 pixels with an X in one corner. Draw the shape in X's by moving them with the up, down, left, right, arrow keys. If you want a blank, type the space bar and move to the next space. You can go over it again and again until it is right, then type Q for Quit and it takes a moment to work out the four numbers. Copy them down, with a sketch of the shape and, if you want more, Escape and RUN.

Now you press the button at the back to wipe out that lovely program and write one to use the shapes. If you get something interesting please let us all know.

```
10 ?%"C5D8R";"THIS PROGRAM PREPARES SHAPE NUMBERS"
20 ?%"2D10R";"DO YOU NEED INSTRUCTIONS? (Y/N)"
25 INPUT, $A
30 IF $A="Y" THEN GOTO 60
40 IF $A<>"N" THEN PRINT "Y OR N PLEASE" :GOTO 20
50 GOTO 120
60 REM INSTRUCTIONS
70 ?"The 8x8 pixel shape is put in a frame"
75 ?"an X is in the top corner"
80 ?"Move the X using the arrow keys"
85 ?"For a blank hit the space bar and move"
90 ?"When the shape is complete"
95 ?"type Q for Quit and the"
100 ?"four shape numbers will be printed"
110 ?"ANY KEY TO CONTINUE"
115 K=KEY (0)
116 IF K=0 THEN GOTO 115
120 REM DRAW FRAME
125 ?%"C"
130 FOR H = 336 TO 345:SPUT H,45:NEXT H
140 FOR J = 696 TO 705:SPUT J,45:NEXT J
150 FOR V = 376 TO 656 STEP 40: SPUT V,73:NEXT V
160 FOR W = 385 TO 665 STEP 40: SPUT W, 73:NEXT W
190 REM USE KEYS TO DRAW SHAPE IN FRAME
200 X = 377 :SPUT X,88
210 K = KEY (0)
220 IF K = 0 THEN GOTO 210
225 IF K = 81 THEN GOTO 300 ! QUIT
230 IF K = 32 THEN GOTO 290 ! space
240 IF K = 11 THEN X = X-40:GOTO 280
250 IF K = 10 THEN X = X + 40 ":GOTO 280
260 IF K = 8 THEN X = X -1 :GOTO 280
270 IF K = 9 THEN X = X +1 :GOTO 280
280 SPUT X,88 :GOTO 210 !"X"
290 SPUT X,32 "GOTO 210 ! space
300 REM CALCULATE THE NUMBERS NEXT
400 DIM P(15), Q(15), R(15), S(15)
410 DIM A(7),B(7), N(15)
420 P=0:Q=0:R=0:S=0
430 X1 = 424 : X2 = 384
440 GOSUB 1000
450 P = N2
460 X1 = 504 : X2 = 464
470 GOSUB 1000
```

```
480 Q = N2
490 X1 = 584 : X2 = 544
500 GOSUB 1000
510 R = N2
520 X1 = 664 : X2 = 624
530 GOSUB 1000
540 S = N2
550 ?TAB (3);P,Q,R,S :?
560 ?"Remember > than 32767 is negative!"
```

If I want to suppress the annoying printing of STOP AT 570
I add a line ...
570 GOTO 570! requires Escape to get out.
(Credit to M. Stevenson)

```
900 REM SUBROUTINE TO CALCULATE
910 REM DECIMAL FROM 16 BITS
1000 FOR Y = 0 TO 7
1010 SGET (X1-Y), A(Y)
1020 IF A (Y) = 88 THEN N(Y) = 1
1030 ELSE N(Y) = 0
1040 NEXT Y
1050 FOR Z = 0 TO 7
1060 SGET (X2-Z),B(Z)
1070 IF B(Z) = 88 THEN N(8+Z) = 1
1080 ELSE N(8+Z) = 0
1090 NEXT Z
1100 Y = 0: N1 = N(0)
1110 Y = Y + 1
1120 N1 = N1 + 2 ∧ Y * N(Y)
1130 IF Y<7 THEN GOTO 1110
1140 N1 = INT (N1)
1150 Z = 8
1160 N2 = N1 + 2 ∧ Z * N(Z)
1170 Z = Z + 1
1180 IF Z<16 THEN N2 = N2 + 2 ∧ Z * N(Z) : GOTO 1170
1190 N2 = INT (N2)
1200 IF N2>32767 THEN N2 = N2-65536
1210 RETURN
     END
```

--- xxx ---

Gary Alexander
Bristol

The hard/soft section this time comes from one of the prizewinners in our Games Competition.

BELL MODIFICATION

The following simple (and very cheap!) modification with a little software allows programs to produce sounds from the CORTEX speaker other than a simple 'beep'.

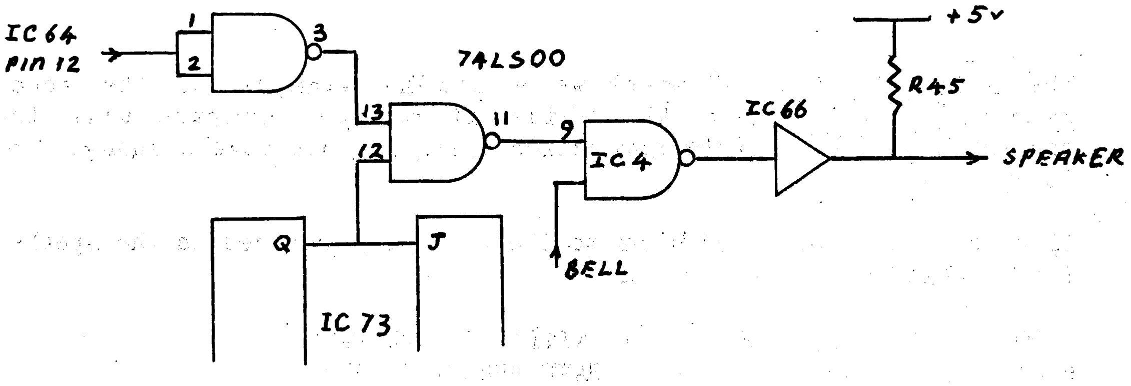


Figure 1

The circuit diagram, (fig.1), shows how the addition of two NAND gates on the spare CRU bit 7 (Q7, pin 12, IC64) can be used to switch the Bell line, (CRU bit 6), from 'normal' to 'musical' (!) operation.

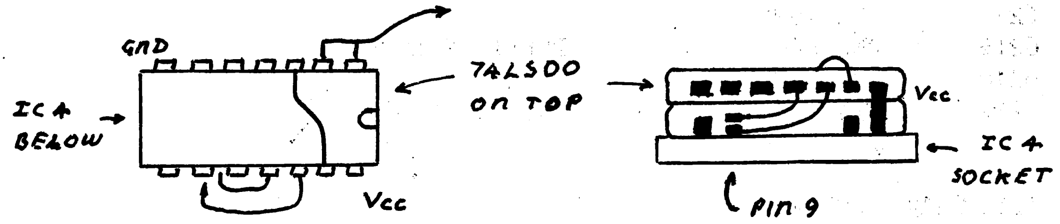


Figure 2

The easiest way to implement the mod is to mount the 74LS00 on top of IC4 as shown in Fig.2.

The following connections are required:

IC64, pin 12 to 74LS00, pins 1 & 2
74LS00, pin 3 to 74LS00, pin 13

Remove pin 9 of IC4 from the socket hole to effectively break the connection between it and the J-Q line of IC73. Then connect:

74LS00, pin 11 to IC4, pin 9
74LS00, pin 12 to empty socket hole (pin 9, IC4 socket)

The two power supply legs can be soldered to IC4. i.e:

74LS00, pin 7 to IC4, pin 7 (Ground)
74LS00, pin 14 to IC4, pin 14 (Vcc)

THE SOFTWARE

To drive the bell, CRU bit 6 is switched by a short Assembler routine which can be called from BASIC. This allows parameters to be passed which vary the frequency and duration of the sound e.g.

```
CALL 06200H,p1,p2,p3
```

where p1 & p2 control the frequency, and p3 controls the duration.

The BASIC routine below shows a simple example of the sound generation possibilities. While it may not compare with the sophistication of a sound generation chip, it does make a change from 'beep'!!

If anyone knows of a solution to the clicking produced in the speaker I'd be glad to hear about it.

```
6200      L1 R12,>0000        INITIALISE CRU BASE
6204      MOV R2,R4           SAVE DURATION (P3)
6206      MOV R0,R3           SAVE FREQUENCY (P1)
6208      SBO 6               BELL ON
620A      DEC R3              LOOP
620C      JNE >6208
620E      MOV R1,R3           SAVE FREQUENCY (P2)
6210      SBZ 6               BELL OFF
6212      DEC R3              LOOP
6214      JNE >6210
6216      DEC R4              DURATION LOOP
6218      JNE >6206
621A      RTWP                BACK TO BASIC
```

ASSEMBLER ROUTINE

```
100  CRB (7) = 1
110  FOR J = 1 TO 25
120  FOR I = 25 TO 40
130  CALL 06200H, 2*1,2*1,I/J
140  NEXT I
150  NEXT J
160  WAIT 20
170  FOR I = 1 TO 5
180  CALL 06200H, 80, 80, 330
190  CALL 06200H, 220, 220, 110
200  NEXT I
210  CRB (7) = 0  ! SET BELL BACK TO NORMAL
```

--- xxx ---

Mark Rudnicki
Bognor Regis

There must be something in the sea air. Mark not only writes programs - he's one of our most prolific correspondents as well. Here's a graphics technique that he uses in his games programs.

BETTER USE OF GRAPHICS MODE

The enclosed machine code sets up the graphics mode like the text mode so that the screen becomes addressable as 32x24 characters. Using this method, the full resolution of the 9929 can be employed. The program supports 4 calls:

1) Initialise. Sets up the screen for use.
   CALL 6200H

2) Initialise a character.

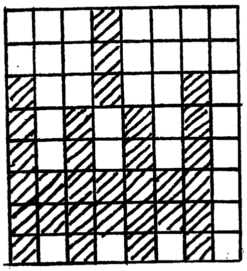
```
CALL 6248H, Chr No.S1,S2,S3,S4,C1,C2,C3,C4
```

Chr No. = 0 to 255 a character number

S1,S2,S3,S4 16 bit values to define the shape (see shape command)

C1,C2,C3,C4 16 bit values to define the colours

e.g.



White on black	S1=>1010_{16}	C1=>$F1F1_{8}$
White on black		
White on black	S2=>$92AA_{16}$	C2=>$F141_{16}$
Blue on black		
Blue on black	S3=>$AAFE_{16}$	C3=>4141_{16}
Blue on black		
Blue on black	S4=>$FEAA_{16}$	C4=>4171_{16}
Cyan on black		

3) Put a character
Puts the character onto the screen:

```
CALL 62A8H, Posn, chrl.    (like sput)
Posn = position 0-767 = 32xrow + column
Chr = character    as defined
```

4) Get a character
Gets the character in a cell and stores the value

```
Var = 0: CALL 62C0H, Posn, ADR (Var)
Posn = position, Var = variable      Like SGET in text mode.
```

Using these commands, very good screen displays can be built up. Note that plot will not work, but the sprite commands will. This was how my program Burglar was done.

Note that on initialistion, the screen locations are filled with the value 32 = a space in ascii. Redefining this character immediately changes all those characters on the screen.

e.g. To flash the new screen red:

```
5  GRAPH: CALL 6200H
10  FOR I = 0 TO 20
20  CALL 6248H,32,0,0,0,0,1616H,1616H,1616H,1616H
30  WAIT 15
40  CALL 6248H,32,0,0,0,0,1111H,1111H,1111H,1111H
50  WAIT 15
60  NEXT I
```

Note that all characters to be used within the game must be userdefined - this is a little tedious but worthwhile - see my programs "Burglar", "Invaders" and "Asteroids".

I will be pleased to reply to any points arising.

```
INIT 6200  LI RO,>0058
     6204  MOVB RO,Q>F121
     6208  SWPB RO
     620A  MOVB RO%>F121
     620E  MOV *RO,*RO
     6210  LI RO,>0300
     6214  LI R1,>2020
     6218  MOVB R1,%>F120
     621C  MOV *R1,*R1
     621E  DEC RO
     6220  JH>6218
     6222  RTWP
     6228  MOVB R10,%>F120
     622C  SWPB R10
     622E  MOV *RO,*RO
     6230  MOVB R10,%>F120
     6234  MOV *RO,*RO
     6236  RT
     6238  SWPB RO
     623A  MOVB RO,%>F121
     623E  SWPB RO
     6240  MOVB RO,%>F121
     6244  MOV*RO,*RO
     6246  RT

CHAR 6248  ANDI RO,255
     624C  SLA RO,3
     624E  ORI RO,>4000
     6252  LI R9,3
     6256  BL %>6238
     625A  MOV R1,R10
     625C  BL %>6228
     6260  MOV R2,R10
     6262  BL %>6228
     6266  MOV R3,R10
     6268  BL %>6228
     626C  MOV R4,R10
     626E  BL %>6228
     6272  AI RO,>0800
     6276  DEC R9
     6278  JH >6256
     627A  AI RO,>0800
     627E  LI R9,3
     6282  BL %>6238
     6286  MOV R5,R10
     6288  BL %>6228
     628C  MOV R6,R10
     628E  BL %>6228
     6292  MOV R7,R10
     6294  BL %>6228
     6298  MOV R8,R10
     629A  BL %>6228
     629E  AI RO,>0800
     62A2  DEC R9
     62A4  JH >6282
     62A6  RTWP
```

```
PUT  62A8  AI RO,>1800
     62AC  ORI RO,>4000
     62BO  BL ‰>6238
     62B4  SWPB R1
     62B6  NOV *RO,*RO
     62B8  MOVB LR1,‰>F120
     62BC  SWPB R1
     62BE  RTWP

GET  62CO  AI RO,>1800
     62C4  BL ‰>6238
     62C8  INCT R1
     62CA  INC R1
     62CC  MOV R1,R1
     62CE  MOVB ‰>F120,*R1
     62D2  RTWP
```

To initialise: GRAPH: CALL 6200H
To initialise a character: CALL 6248H,N,S1,S2,S3,S4,C1,C2,C3,C4
To put a character: CALL 62A8H, Posn, Chr
To get a character: CALL 62COH, Posn, ADR (Var).

--- xxx ---

Robert Lee
Eastbourne

More sea air - more useful tips. Thanks Robert.

Brendan's bulletproof ram is great, providing you like taking the computer apart. For those of you not so keen, do not despair. Memory locations 06000H to 060EFH are also protected from system resets, although only 240 bytes, it is very useful.

The following is what I have found, so far:

Operating system calls. Rev 1.1
CALL 021CH Stops program, prints"*Ready".
CALL 01F4H NEW (Clears User program).
CALL 03EF2H RUN BASIC program
CALL 01744H LOAD program from cassette
CALL 01752H LOAD program, R8 points to 8 byte name.

Useful memory locations. Rev 1.1
001EH 16 Bit UNIT enable flags.
551AH Contents are loaded into 9995 interval timer.
5534H Colour attribute byte, default 47H.
ED6AH Cursor ON/OFF flag, 0000=ON.
18B2H Cassette 9902 interval register contents. 04D0H=300
      baud, 0468H=600 baud, 043H=1200 baud.
0084H UNIT 1 BLWP pointer
0086H UNIT 2 CRU base address (80), 1 signifies CRU.
0088H UNIT 3 CRU base address (180)
008AH UNIT 4 BLWP pointer
008CH Next 12 words, undefined UNIT jumps.

Useful routines. Rev 1.1
0114H LEVEL 1 Interrupt
0278H LEVEL 2 Interrupt
00B6H LEVEL 3 Interrupt, timer and clock
0602H LEVEL 4 Interrupt, I/O, keyboard, CRU.
0785H Centronics printer routine.

FOUR TIMES FASTER, SAVING AND LOADING WITH TWO RESISTORS

To change the cassette SAVE and LOAD baud rates is an extremely simple task. But in order that the system works, two modifications have to be made to the cassette port.

1) To double the frequency of the oscillator IC72c. This can be done by soldering a 100k resistor in parallel with R54 (100K).

2) To speed up the monostable IC70, so that the timing is correct for the new frequencies. This can be done by soldering a 51K resistor in parallel with R44 (39K).

At first, you may find it necessary to operate a dual system with a double pole switch to switch the two resistors in and out of circuit. This enables old programs to be loaded, and then SAVED at 1200 baud after flicking the switch.

To make the computer SAVE and LOAD at 1200 baud, memory location 018B2H must be changed from 04D0H (300 baud) to 0434H (1200 baud).

The program shown below will do this automatically, and will print the message 'Loading' when the computer has found the correct program. The program should be SAVED at 300 baud, but with the new system switched IN. The entry point when SAVING is 0620AH. SAVING at 300 baud enables the computer to LOAD the program and if the auto boot is used, to load the following program at 1200 baud. 4 times faster.

All references to component numbering, refer to the original overlay, as published by Electronics Today International.

```
6200 06A0 BL ‰>622C                      ;DATA
6204 06A0 BL ‰>6240                      ;DATA
6208 0434 DATA >0434                     ;DATA Baud rate.
620A C820 MOV ‰>6200,‰>17E0              ;Set up pointers.
6210 C820 MOV ‰>6202,‰>17E2              ;
6216 C820 MOV ‰>6204,‰>16EA              ;
621C C820 MOV ‰>6206,‰>16EC              ;
6222 C820 MOV ‰>6208,‰>18B2              ;
6228 0460 B   ‰>624C                     ;Jump to end
622C C3E0 MOV ‰>EFB0,R15                 ;Continue with program
6230 0FA0 MSG ‰>6236                     ;Print message 'loading'
6234 045B RT                             ;Return from subroutine
6236 0A0D DATA > 0A0D                    ;DATA 'Loading'
6238 4C6F DATA >4C6F                     ;
623A 6164 DATA >6164                     ;
623C 696E DATA >696E                     ;
623E 6700 DATA >6700                     ;
6240 04CI CLR R1                         ;Wait loop
6242 0601 DEC R1                         ;
```

```
6244 16FE JNE >6242                     ;
6246 C060 MOV %>EFB6,R1                 ;
624A 045B RT                            ;Return from subroutine
624C 0460 B   %>021C                    ;Return to BASIC
```

For auto booting of the next program on the tape , the following should be entered.

```
624C 0208 LI  R8,>6254                  ;Set up pointer to name
6250 0460 B   %>1752                    ;Branch to LOAD routine
6254    $Name of next program, up to 8 characters.
```

CENTRONICS PRINTER SPOOLER

WARNING! BEFORE RUNNING THIS PROGRAM, AN AREA OF MEMORY MUST BE SAVED FOR THE SPOOLER BUFFER. THIS CAN ONLY BE DONE BY ENTERING NEW 8000H.

This program enables the computer to use part of it's memory as a printer spooler, enabling high speed dumping of programs or text to the spooler buffer. The printer would then print out the contents of the buffer at it's relatively slow speed, freeing the computer for other tasks. The spooler is transparent to the user and does not noticeably slow the computer down.

The program is written in 3 sections,

1) 06070H-060A8H Sets up pointers so that the spooler program becomes active. The entry point for the program is 0607CH and this should be used when the program is saved, with auto run set.

2) 06000H-06040H Passes characters from the CORTEX's printer routine to the spooler routine, and checks for full buffer conditions i.e. printer pointer at bottom of buffer and computer pointer at top of buffer, or computer pointer= printer pointer - 1, where wrap-around occurs in the buffer.

3) 06044H-0606AH Accessed by the level 3 interrupt of the CPU. This routine is called every 1/100th of a second by the 9995 internal decrementer and is used to check for the buffer empty condition, printer pointer=computer pointer. The routine then checks the printer BUSY line and sends a character if this line is at logic '0'. Care must be taken not to over-write this section of code while the spooler program is active, otherwise the computer will crash.

The buffer length is 7823 characters (06071H-07FFFH) and is set by the contents of memory locations 0606CH and 0606EH. Once set up, memory locations 06071H-060A8H are over written by the buffer.

In use, the printer is accessed as normal with either UNIT 4 in BASIC or P9 in the Monitor. Now listings will not be delayed by the printer speed (up to 7K of characters). Don't forget to disable the printer after you have finished (UNIT-4 or P 1).

```
                                   ;PRINTER SPOOLER
[]U 6000 60A8                      ;BY R.M. LEE
6000 0289  CI  R9,>0000            ;Test for printer pointer=0
6004  1604  JNE  >600E             ;If not zero then skip next section
6006 880A C R10,‰>606E             ;Is computer at top of buffer
600A 1601 JNE >600E                ;If it isn't then goto next section
600C 10F9 JMP >6000                ;Buffer is full, wait for room
600E 880A C R10, ‰>606C            ;If buffer not full,test computer
6012 1601 JNE >6016                ;at top of buffer,if not then
                                    continue
6014  04CA  CLR  R10               ;else reset computer pointer to
                                    bottom
6016 DAA0 MOVB ‰>6070,‰>
6071 (R10)                         ;Move character from temp store to
                                    buffer
601C C20A MOV R10,R8               ;Test for computer pointer=printer
601E 0588 INC R8                   ;pointer-1
6020 8248 C R8,R9                  ;If true then wait for room
6022 13FE JEQ >6020                ;
6024 058A INC R10                  ;Increment computer pointer
6026 880A C R10,‰>606C             ;Is computer pointer at top of buffer
602A 1601 JNE >602E                ;If not then return
602C 04CA CLR R10                  ;If it is then reset pointer
602E 045B RT                       ;Return from subroutine
6030 D839 MOVB *R9+,‰>6070         ;Move character to temp store
6034 02E0 LWPI >EEA0               ;Set up workspace pointer
6038 06A0 BL ‰>6000                ;Call spooler sub-routine
603C 02E0 LWPI >EE74               ;Return workspace pointer
6040 0460 B ‰>0758                 ;Branch back to printer routine
6044 02E0 LWPI >EEA0               ;Set up workspace pointer
6048 8289 C R9,R10                 ;Test for buffer empty
604A 130D JEQ >6066                ;Do not print if true
604C 020C LI R12,>0800             ;Set up CRU base for printer
6050 IF09 TB 9                     ;Is printer busy
6052 1309 JEQ >6066                ;Do not print if true
6054 3229 LDCR ‰>6071(R9),9        ;Move character from buffer to
6058 ID08 SBO 8                    ;printer and pulse strobe line
605A 1E08 SBZ 8                    ;
605C 0589 INC R9                   ;increment printer pointer
605E 8809 C R9,‰>606C              ;Is printer pointer at top of buffer-1
6062 1601 JNE >6066                ;If not then return
6064 04C9 CLR R9                   ;Reset printer pointer
6066 02E0 LWPI >ED14               ;Return workspace pointer
606A 0380 RTWP                     ;Return from interrupt
606C IF8F TB -113                  ;DATA,buffer length in bytes
606E IF8E TB -114                  ;DATA,buffer length-1in bytes
6070 8289 C R9,R10                 ;DATA,for set up of program
6072 1B08 JH >6084                 ;
6074 0460 B ‰>6030                 ;
6078 0460 B ‰>6044                 ;
607C 0300 LIMI >0002               ;Entry point for program setup
6080 C820 MOV ‰>6070,‰>0758        ;
6086 C820 MOV ‰>6072,‰>075A        ;
608C C820 MOV ‰>6074,‰>075C        ;
6092 C820 MOV ‰>6076,‰>075E        ;
6098 C820 MOV ‰>6078,‰>010E        ;
609E C820 MOV ‰>607A,‰>0110        ;
60A4 0300 LIMI >000F               ;
```

```
60A8 0460 B %>021C          ;Return to BASIC
[]P 1                       ;R9=PRINTER POINTER
                            ;R10=COMPUTER POINTER
```

--- xxx ---

Richard Swingwood
Colchester

Here's some useful information that Richard has dug out of his CORTEX.
The COL function can be corrected with the following:

MWD (1D12H) = OF120H

The real-time clock executes at OOB6H with a workspace pointer of ED14H. Registers are used as follows:

R0 - 10ms count
R1 - hours
R2 - minutes
R3 - seconds
R4 - }
R5 - } TIC count

There is enough room at the end of this routine to patch in a branch to a user routine which will then be executed every 10 milliseconds -

Put BL %> user-routine at address 010EH and terminate with a RTWP.

Registers 6 through 10 are free for you to use.

BASIC STATEMENTS - EXECUTION ADDRESSES

STOP	1E20	SPUT	19FC
GOTO	24FC	SGET	1992
GOSUB	2500	BOOT	321A
ELSE	3FC2	SWAP	3348
REM	3F36	MOTOR	186E
FOR	2146	MAG	1BFA
LET	2772	TOF	3IDE
DATA	3F36	TON	3ID6
NEXT	2240	POP	259C
ERROR	1EA4	DIM	2032
PRINT	2AB8	ON	29FA
CALL	IFIE	IF	25AE
LOAD	173A	DEF	200E
INPUT	262E	NEW	OICE
READ	2CFA	END	1E3A
RESTOR	2D36	*	549E
RETURN	256C	BIT	4E6A
UNIT	31E4	CRB	IFA6
TIME	313C	CRF	IF8C
SAVE	1642	MEM	2908
BASE	1F86	MWD	299E
ESCAPE	2138		
NOESC	213E		

RANDOM	2C9E
BAUD	0164
ENTER	20F6
PLOT	5142
UNPLOT	513E
COLOUR	1AD8
PURGE	2BFC
GRAPH	1A84
TEXT	1A7E
WAIT	1B58
CHAR	1A88
NUMBER	29CC
LIST	3C0A
RENUM	2D78
SPRITE	1B9E
SHAPE	1B6A

BASIC FUNCTIONS - EXECUTION ADDRESSES

ABS	2466
ADR	29C8
ASC	2936
ATN	18B4
COS	5346
EXP	4A28
FRA	249A
INT	2440
LOG	5056
KEY	2482
SIN	535A
SQR	53E6
SYS	241C
TIC	3128
SGN	3038
BIT	4E9E
CRB	1FEE
CRF	1FC4
MEM	2924
MWD	29B8
LEN	1F7C
MCH	1F6E
POS	1F4A
COL	1B14
MOD	24C8

An error in the graphics mode clear-screen routine causes the vertical position of the SPRITE 0 to be changed to 0. This can be corrected with MWD (594H)=2FFH.

--- xxx ---

Andy Kendall
Bristol

Andy's a keen enthusiast who seems to have set up a little sub-group down in the West Country. This lower-case generation program can be blown into your EPROMS if you get fed up with loading it from tape or disc. He and John Mackenzie have worked out a wrinkle that improves string-handling when using CDOS and Andy claims to be hot on the trail of a 74LS2001 subsititute!

```
LIST

10 REM Lower case generation program
20 REM Written by A.J.Kendall
30 REM Tel: 0272 744444
40 DATA3,-28087,9088,16647,4681,9984,3,-28607,896,2083,-28087,9088
50 DATA3,4721,896,4258,7712,-32256,3,4680,-8036,16647,4681,9344
60 DATA 8194,2080,-32256,2048,-32248,9356,16677,6241,17536,16644,
   4161,8960
70 DATA 7,-27307,21824,7,4681,9344,3,4681,8960,7,4681,-15344
80 DATA 3,-28088,8061,7,4673,1024,3,-28624,9984,16655,4161,8960
90 DATA 4,-28087,9088,8,-23983,16896,10,-21846,-23296,8,-27615,
   18560
100 DATA 4,-28088,-8036,7,-32207,1920
110 FOR CH=97 TO 122
120 READ  A,B,C
130 CHAR CH,A,B,C
140 NEXT CH
150 CHAR 79,29224,-23926,9984
160 TEXT
```

--- xxx ---

Tim Gray
West Midlands

Tim's been doing some interesting things with external video. He's also sent in about a pound and a half of listings. We've included a sprite movement routine and his 3D graphics program - keep 'em coming Tim.

Here is a program that allows automatic movement of sprites without the need for complex for-next loops.

The main program resides in an area of memory from 5EEA to 604E not normally used by the CORTEX software when in GRAPH mode, it is where text characters above 128 would be if they had been defined using the CHAR statement. The variables are stored in an area of memory that will be overwritten if an error message is printed or if the text screen is scrolled but this should not be a problem as long as speeds are defined within a program after the GRAPH statement is used.

```
CALL "SPEED", 05FF6H,<sprite no.>,<X speed>,<Y speed>,<auto stop>
```

This section builds a table of X and Y speeds and auto stop for each sprite.

XXXX one word is used for each parameter.
YYYY and the table covers 5800 to 58C0 (32 sprites)
AS if auto stop is not zero the sprite will stop moving when it gets to an off screen location if it is zero the sprite will re-appear at the opposite end of the screen from where it left.
X and Y speeds should be in the range -255 to +255 and zero for no movement.

CALL "ANIMATE",06026H,<N>

This section puts a patch into interrupt 4 vector to call the main routine and enable VDP interrupts if N is 1 or disable interrupts if N is 0.

To use the program first put all sprites to their starting positions, then set their speeds using CALL "SPEED" then enable the main program with CALL "ANIMATE".

The main program uses the VDP interrupt to automatically update sprite positions at the end of every TV field and sets the early clock bit as required to enable sprites to bleed off the screen in any direction.

The current positions of sprites can be found in a table from 5700 to 5780 two word entries are used for each sprite the MSB of the first word contains the Y position and the MSB of the second word contains the X position.

The sprite speeds can be changed during an animated sequence but no attempt should be made to write anything to the VDP without first switching off the animation using CALL "ANIMATE" as the VDP may cause an interrupt during the operation and corrupt the VDP address setup.

SPRITE SPEEDS

```
5EEA 5900 WP 5EEE PC
5EEE 04CC CLR R12
5EF0 1F06 TB 6
5EF2 137A JEQ >5FE8
5EF4 C020 MOV ‰>0026,R0
5EF8 1375 JEQ >5FE4
5EFA 020A LI R10,>5700
5EFE 0208 LI R8,>1B00
5F02 0209 LI R9,>5800
5F06 0289 C1 R9,>58C0
5F0A 1B6C JH >5FE4
5F0C C039 MOV *R9+,R0
5F0E C079 MOV *R9+,R1
5F10 C1B9 MOV *R9+,R6
5F12 C000 MOV R0,R0
5F14 1607 JNE >5F24
5F16 C041 MOV R1,R1
5F18 1605 JNE >5F24
5F1A 022A AI R10,>0004
5FIE 0228 AI R8,>0004
5F22 10FI JMP >5F06
5F24 06A0 BL ‰>05F2
5F28 D0E0 MOVB ‰>F120,R3
5F2C C0C3 MOV R3,R3
```

```
5F2E D120 MOVB ‰>F120,R4
5F32 C104 MOV R4,R4
5F34 D0A0 MOVB ‰>F120,R2
5F38 C082 MOV R2,R2
5F3A D160 MOVB ‰>F120,R5
5F3E 9803 CB  R3,‰>5FEA
5F42 1350 JEQ >5FE4
5F44 D683 MOVB R3,*R10
5F46 A681 A R1,*R10
5F48 981A CB *R10,‰>5FEA
5F4C 13FC JEQ >5F46
5F4E C186 MOV R6,R6
5F50 1308 JEQ >5F62
5F52 981A CB *R10,‰>5FEC
5F56 1A05 JL >5F62
5F58 981A CB *R10,‰>5FEE
5F5C 1B02 JH >5F62
5F5E 04E9 CLR ‰>FFFC(R9)
5F62 05CA INCT R10
5F64 D684 MOVB R4,*R10
5F66 C000 MOV R0,R0
5F68 1115 JLT >5F94
5F6A A680 A R0,*R10
5F6C 1708 JNC >5F7E
5F6E E160 SOC ‰>5FF0,R5
5F72 04DA CLR *R10
5F74 C186 MOV R6,R6
5F76 1323 JEQ >5FBE
5F78 04E9 CLR ‰>FFFA(R9)
5F7C 1020 JMP >5FBE
5F7E 2160 COC ‰>5FF0,R5
5F82 161D JNE >5FBE
5F84 981A CB *R10, ‰>5FF2
5F88 121A JLE >5FBE
5F8A 4160 SZC ‰>5FF0,R5
5F8E 66A0 S ‰>5FF4,*R10
5F92 1015 JMP >5FBE
5F94 A680 A R0,*R10
5F96 911A CB *R10,R4
5F98 1208 JLE >5FAA
5F9A 4160 SZC ‰>5FF0,R5
5F9E 071A SETO *R10
5FA0 C186 MOV R6,R6
5FA2 130D JEQ >5FBE
5FA4 04E9 CLR ‰>FFFA (R9)
5FA8 100A JMP >5FBE
5FAA 2160 COC ‰>5FF0,R5
5FAE 1307 JEQ >5FBE
5FB0 981A CB *R10,‰>5FF2
5FB4 1B04 JH >5FBE
5FB6 E160 SOC ‰>5FF0,R5
5FBA A6A0 A ‰>5FFR,*R10
5FBE 0268 ORI R8,>4000
5FC2 06A0 BL ‰>05F2
5FC6 0248 ANDI R8,>3FFF
5FCA 064A DECT R10
5FCC D83A MOVB *R10+,‰>F120
5FD0 058A INC R10
5FD2 D83A MOVB *R10+,‰>F120
```

```
5FD6 058A INC R10
5FD8 D802 MOVB R2,‰>F120
5FDC C082 MOV R2,R2
5FDE D805 MOVB R5,‰>F120
5FE2 109D JMP >5FIE
5FE4 D2E0 MOVB ‰>F121,R11
5FE8 0380 RTWP
5FEA D000 MOVB R0,R0
5FEC C800 MOV R0,‰>D800
5FF0 8000 C R0,R0
5FF2 1F00 TB 0
5FF4 2000 COC R0,R0
5FF6 C140 MOV R0,R5
5FF8 0245 ANDI R5,>00IF
5FFC 0204 LI R4,>0006
6000 3944 MPY R4,R5
6002 0226 AI R6,>5800
6006 0A41 SLA R1,4
6008 CD81 MOV R1,*R6+
600A 0A42 SLA R2,4
600C CD82 MOV R2,*R6+
600E CD83 MOV R3,*R6+
6010 C140 MOV R0,R5
6012 0245 ANDI R5,>001F
6016 0204 LI R4,>0004
601A 3944 MPY R4,R5
601C 0226 AI R6,>5700
6020 04F6 CLR *R6+
6022 04D6 CLR *R6
6024 0380 RTWP
6026 0207 LI R7,>5EEA
602A C807 MOV R7,‰>EDAE
602E C820 MOV ‰>1C1A,‰>604A
6034 0206 LI R6,>2000
6038 C000 MOV R0,R0
603A 1301 JEQ >6042
603C E806 SOC R6,‰>604A
6040 1002 JMP >6046
6042 4806 SZC R6,‰>604A
6046 06A0 BL ‰>05E0
604A E281 SOC R1,R10   mov R1,R2
604C 0000 DATA >0000
604E 0380 RTWP
```

```
1 REM ******************
2 REM ****3D GRAPH******
3 REM ******************
4 REM ** AN EXAMPLE OF 3D COMPUTING***
5 REM **    BY TIM GRAY            ***
6 REM
7 GOTO 3000
8 RESTOR
10 TEXT : READ PN !number of points
20 READ MAX !maximum value used"
30 MAX=ABS(MAX) !auto scaling of size
40 DIM PT(PN,3), PTE(PN,2)
50 FOR I=1 TO PN
55 REM *** X,Y,Z coordinates for all points
```

```
70 READ XO: PT(I,1)=XO/MAX
80 READ YO: PT(I,2)=YO/MAX
90 READ ZO: PT(I,3)=ZO/MAX
110 NEXT I: PRINT "<OC>";
120 READ LN !No of lines to connect points
130 DIM LIN(LN,2)
135 REM which points connected be each line
140 FOR I=1 TO LN
160 READ LIN(I,1)
170 READ LIN(I,2)
190 NEXT I
200 RH=15: D=750: TH=4.7: PH=3.5: CX=126: CY=96
210 COLOUR 15,1: GRAPH
250 S1=SIN(TH): C1=COS(TH): S2=SIN(PH): C2=COS(PH)
299 REM *** TRANSFORMATION ***
300 FOR I =1 TO PN
310 X=PT(I,1): Y=PT(I,2): Z=PT(I,3)
320 XE=X*S1+Y*C1: YE=-X*C1*C2-Y*S1*C2+Z*S2: ZE=X*S2*C1-Y*S2
    *S1-Z*C2+RH
330 SX=(D*XE/ZE)*0.7+CX: SY=CY-D*YE/ZE
340 IF SX <0 OR SX>255 THEN SX=0
350 IF SY <0 OR SY>192 THEN SY=0
360 PTE(I,1)=SX: PTE(I,2)=SY
370 NEXT I
399 REM **** PLOT LINES****
400 PRINT "<OC>"
405 FOR I=1 TO LN
410 X1=PTE(LIN(I,1),1)
420 Y1=PTE(LIN(I,1),2)
430 X2=PTE(LIN(I,2),1)
440 Y2=PTE(LIN(I,2),2)
450 IF X1=0 OR Y1=0 OR X2=0 OR Y2=0 THEN GOTO 470
460 PLOT X1,Y1, TO X2,Y2
470 NEXT I
499 REM *** READ KEYBOARD ***
500 K=KEY(0)
510 IF K=04EH THEN RH=RH-1: GOTO 250
520 IF K=046H THEN RH=RH+1: GOTO 250
530 IF K=045H THEN D=D+50: GOTO 250
540 IF K=052H THEN D=D-50: GOTO 250
550 IF K=8 THEN TH=TH+0.1: GOTO 250
560 IF K=9 THEN TH=TH-0.1: GOTO 250
570 IF K=11 THEN PH=PH+0.1: GOTO 250
580 IF K=10 THEN PH=PH-0.1: GOTO 250
590 ELSE GOTO 500
700 GOTO 250

1999 REM ***PN, MAX ***
2000 DATA 40,2
2009 REM *** TOP CIRCLE ***
2010 DATA 0,2,-2,1.4,2,-1.4,2,2,0,1.4,2,1.4,0,2,2,-1.4,2,1.4,-2,2,0,-
     1.4,2,-1.4
2019 REM *** 2nd CIRCLE ***
2020 DATA 0,0.5,-1.2,0.85,0.5,-0.85,1.2,0.5,0,0.85,0.5,0.85,0,0.5,1.2,
     -0.85,0.5,0.85
2030 DATA -1.2,0.5,0,-0.85,0.5,-0.85
2039 REM *** 3rd CIRCLE ***
2040 DATA 0,0,-0.7,0.5,0,-0.5,0.7,0,0,0.5,0,0.5,0,0,0.7,-0.5,0,0.5,-
     0.7,0,0,-0.5,0,-0.5
```

```
2049 REM *** 4th CIRCLE ***
2050DATA0,-1.7,-0.4,0.3,-1.7,-0.3,0.4,-1.7,0,0.3,-1.7,0.3,0,-1.7,
     0.4,-0.3,-1.7,0.3
2060 DATA -0.4,-1.7,0,-0.3,-1.7,-0.3
2069 REM *** 5th CIRCLE ***
2070DATA0,-2,-1.2,0.85,-2,-0.85,1.2,-2,0,0.85,0,-2,1.2,-0.85,-2,
     0.85,-1.2,-2,0,-2,0.85,-2,0.85,0,-2,1.2,0.85,-2,0.85,-1.2,-2,0
2080 DATA -0.85,-2,-0.85
2089 REM *** LN ***
2090 DATA 72
2099 REM *** TOP SECTION ***
2100 DATA 1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,1,1,9,2,10,3,11,4,12,5,
     13,6,14,7,15,8,16
2109 REM *** 2nd SECTION ***
2110 DATA 9,10,10,11,11,12,12,13,13,14,14,15,15,16,16,9,9,17,10
     18,11,19,12,20,13,21,14,22,15,23
2120 DATA 16,24
2129 REM *** 3rd SECTION ***
2130 DATA 17,18,18,19,19,20,20,21,21,22,22,23,23,24,24,17
2140 DATA 17,25,18,26,19,27,20,28,21,29,22,30,23,31,24,32
2149 REM *** BOT SECTION ***
2150 DATA 25,26,26,27,27,28,28,29,29,30,30,31,31,32,32,25
2160 DATA 25,33,26,34,27,35,28,36,29,37,30,38,31,39,32,40
2170 DATA 33,34,34,35,35,36,36,37,37,38,38,39,39,40,40,33
3000 COLOUR 15,12: TEXT
3010 PRINT
3020 PRINT "The object displayed can"
3030 PRINT "be moved in three"
3040 PRINT "dimensions by using the following"
3050 PRINT
3060 PRINT "CURSOR LEFT revolves object anticlock"
3070 PRINT "CURSOR RIGHT revolves object clockwise"
3080 PRINT "CURSOR UP revolves object away"
3090 PRINT "CURSOR DOWN revolves object towards"
3100 PRINT "E  enlarges object"
3110 PRINT "R  reduces object"
3120 PRINT "N  brings object nearer"
3130 PRINT "F  moves object away"
3140 PRINT
3150 PRINT "the difference between the last two"
3160 PRINT "is the perspective"
3170 PRINT
3180 INPUT "PRESS RETURN " ;$INP
3190 IF $INP<>" " THEN GOTO 8
3200 GOTO 3180
```

--- xxx ---

M D Rudnicki
Bognor Regis

Enclosed is an article for the CORTEX Users Magazine. The program enables the use of the "MULTICOLOUR" mode on the video display processor which divides the screen into 64x24 block, each of which can be any of the 16 colours.

For a test program, the user could try the following:

```
10 GRAPH: CALL 6200H,0C8H,I:CALL 62C8H
20 FOR Y=0 TO 47
30 FOR X=0 TO 63
40 CALL 62A8H,X,Y, INT (RNDX 15.99)
50 NEXT X
60 NEXT Y
70 GOTO 20
```

Press "ESCAPE" to stop the program then either type TEXT <return> or GRAPH <return> to return to a standard mode.

P.S. ETI are currently scrutinising my parallel I/O board which has been running faultlessly for some 2 months now. I've connected 2 sound boards to it (1 with 2 x SN76489. 1 with M112 (see Maplins catalogue), a 5 octave keyboard and my PE master rhythm. If nothing is forthcoming from ETI then I will be willing to make available some of this hardware for the User's Group in some form or another.

MACHINE CODE LOADER PROGRAM FOR MULTICOLOUR MODE

```
10000 FOR I = 6200H TO 62DEH STEP 2
10010 READ A
10020 MWD (I) = A
10030 NEXT I : RETURN
10040 DATA 1728,-10240,-3807,-16368,609,128,1729,-10239
10050 DATA -3807,-15279,896,4096,608,16384,1728,-10240
10060 DATA -3807,1728,-10240,-3807,-15344,1729,-10239,-3808
10070 DATA 1729,576,16383,1115,1728,-10240,-3807,1728
10080 DATA -10240,-3807,-15344,-12192,-3808,1729,1115,4096
10090 DATA -16064,581,1,-16192,2323,-16127,2356,2644
10100 DATA -15932,2611,2615,-24125,-15999,582,7,-24122
10110 DATA -16377,1696,25144,645,0,5386,2626,577
10120 DATA 15,-24510,1696,25112,896,0,0,0
10130 DATA 577,240,-24510,1969,25112,896,0,0
10140 DATA 1696,-25112,896,0,641,15,5378,1120,
10150 DATA -25168,641,31,5380,545,64,1120,25168
10160 DATA 545,128,1120,-25168,1218,-16382,544,6144
10170 DATA -16318,577,31,-16190,2419,2643,-24509,1696
10180  DATA -25122,1410,642,768,4592,896,0,0
```

TO SET UP THE SCREEN, THE FOLLOWING NEEDS TO BE INCLUDED INTO THE PROGRAM:

```
LINE NO XXX:   GRAPH: CALL 6200H,0C8H,1: CALL 62C8H
```

Thereafter, to set a pixel:

```
CALL 62A8H,X,Y,C
Where X=X co-ordinate (0-63)
      Y=Y co-ordinate (0-47)
      C=Colour (0-15)
```

--- xxx ---

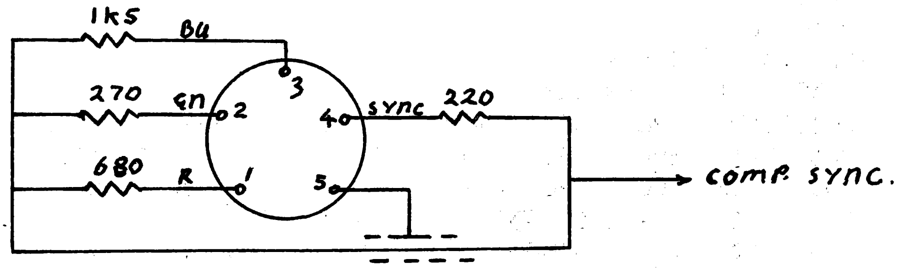
C C Kuan
London

I am so glad that I am not alone. Long Live the CORTEX!
Looking at all the other micros that have gone under despite huge finance and mass production facilities, it is wonderful to see that the CORTEX is now thriving into CORTEX II.
Could be useful:

1.
```
10 GRAPH
20 T=TIC [0]
30 RANDOM T-32767*INT [T/32767]
40 PLOT 256*RND,192*RND
50 GOTO 40
```

The above generates a regular pattern, not "completely random".

2. I have a green monitor, but no composite signal from CORTEX (amplifier stage to boost modulator input doesn't work, as in Jupiter Ace). Luckily I found this in Phillips P7001 green monitor's manual; it takes RGB or composite sync.



Question: Who has the Data Sheet for the Canon MD-110 SSSD disc drive, even Canon doesn't have it anymore?

Andy Kendall
Bristol

Here's a couple of programs for the User Group. One is an adaption of a P C W listing and the other is an original, which I've found useful on several occasions.

Here's something to ponder!,
Type in:

```
10  N=62.4
20  PRINT N
```

Then list it. Interesting eh?

```
PRINT: UNIT 2

LIST

 10  REM CHAR.DEF.PROG.
 20  REM A.J.KENDALL 22.2.84
 30  PRINT % "C": PRINT "    CHARACTER DEFINITION PROGRAM."
 40  PRINT :  PRINT "Use the cursor control keys to move the
     '□' to the desired point".
 50  PRINT "Press 'HOME' to fill or erase '□' point".
 55  ? : ? : ? "Press the SPACE bar to end definition".
 60  ? : ? : ? "Press the SPACE bar to continue".
 61  K=KEY[0]: IF K=32 THEN GOTO 70
 62  ELSE GOTO 61
 70  F=0
 72  CHAR 133,-488,24966,6271
 73  CHAR 134,487,-24967,-6272
 80  CO=0: C1=0: C2=0: C3=0
 85  REM SWITCH OFF CURSOR
 90  PRINT "<1D>"
100  PRINT % "C": INPUT "Enter thechar. no. you wish to define-;CO
110  REM
120  CHAR 128,-1,-1,-1: TEXT
130  DIM A[8,10]: DIM B[8,10]
140  B[0,0] = 128
150  FOR Y=1 TO 10
160  FOR X=1 TO 8
170  B=((X+15)+40*Y)
180  A[X,Y] = B
190  NEXT X
200  NEXT Y
210  FOR Y=1 TO 10
220  FOR X=1 TO 8
230  B[X,Y] = 32
240  NEXT X
250  NEXT Y
260  FOR X=1 TO 8: B[X,1] =128: NEXT X
270  FOR X=1 TO 8: B[X,10] =128: NEXT X
280  FOR Y=1 TO 10: B[1,Y] =128: NEXT Y
290  FOR Y=1 TO 10: B[8,Y] =128: NEXT Y
300  B[4,5] =133
310  FOR Y=1 TO 10
320  FOR X=1 TO 8
330  SPUT A[X,Y],B[X,Y]
```

```
340  NEXT X
350  NEXT Y
360  IF F=0 THEN GOSUB 870
370  F=1
380  K=KEY [0]
390  W=V: Z=T
400  IF K=8 THEN V=V-1: GOTO 470
410  IF K=9 THEN V=V+1: GOTO 470
420  IF K=10 THEN T=T+1: GOTO 480
430  IF K=11 THEN T=T-1: GOTO 480
440  IF K=30 THEN GOTO 540
450  IF K=32 THEN GOTO 560
460  GOTO 380
470  IF V <2 OR V>7 THEN V=W: PRINT "<07>": PRINT % "H": GOTO 380
480  IF T <2 OR T>9 THEN T=Z: PRINT "<07>": PRINT % "H": GOTO 380
490  IF B [W,Z] =134 THEN B [1,0] =128
495  IF B [W,Z] =133 THEN B [1,0] =32
500  IF B [V,T] =128 THEN B [0,0] =134
505  IF B [V,T] =32 THEN B [0,0] =133
510  B [W,Z] = B [1,0]
511  B [V,T] = B [0,0]
515  W=V: Z=Y: GOTO 310
540  IF B [W,Z] =133 THEN B [W,Z] =134: GOTO 310
550  IF B [W,Z] =134 THEN B [W,Z] =133: GOTO 310
560  FOR Y=9 TO 2 STEP -1
570  FOR X=7 TO 2 STEP -1
580  IF A [X,Y] =180 THEN CO =0.5
590  IF A [X,Y] =298 THEN CO =0.5
600  IF A [X,Y] =382 THEN CO =0.5
610  CO =CO + CO
620  IF B [X,Y] =134 THEN GOTO 640
630  IF B [X,Y] <>  128 THEN GOTO 670
640  IF A [X,Y] < 181 THEN C1 =C1 + CO
650  IF A [X,Y] < 299 AND A [X,Y] > 180 THEN C2 =C2 + CO
660  IF A [X,Y] > 298 THEN C3 =C3 + CO
670  NEXT X
680  NEXT Y
690  IF C1 > 32768 THEN C1 =C1 - 65536
700  IF C2 > 32768 THEN C2 =C2 - 65536
710  IF C3 > 32768 THEN C3 =C3 - 65536
720  PRINT % (0,13)" Char. no."; CO
730  PRINT "Exp 1- "; C1
740  PRINT ."Exp 2- "; C2
750  PRINT "Exp 3- "; C3
760  PRINT :PRINT "Immediate effect?"
770  K=KEY [0]
780  IF K = 89 THEN GOTO 810
790  IF K = 78 THEN GOTO 820
800  GOTO 770
810  CHAR CO,C1,C2,C3: TEXT: SPUT 460,CO
820  PRINT: PRINT "Again?"
830  K=KEY [0]
840  IF K = 89 THEN GOTO 70
845  REM SWITCH ONCURSOR IF 78
850  IF K = 78 THEN PRINT "<1C>": END
860  GOTO 830
870  V = 4: T = 5
880  RETURN
900  GOTO 670
```

```
JUNE                 60 A.D.
SUN  MON  TUE  WED  THU  FRI  SAT
 1    2    3    4    5    6    7
 8    9   10   11   12   13   14
15   16   17   18   19   20   21
22   23   24   25   26   27   28
29   30
```

```
LIST

10  REM CALENDERS PCW OCT 83 PG 285
20  PRINT % "C"
30  PRINT "CALENDER"
40  PRINT
50  DIM $ M [12,2]
60  FOR 1=1 TO 12
70  READ $ M [1,0]
80  NEXT I
90  DATA "JANUARY", "FEBRUARY", "MARCH", "APRIL", "MAY", "JUNE",
    "JULY", "AUGUST", "SEPTEMBER"
100 DATA "OCTOBER", "NOVEMBER", "DECEMBER".
110 PRINT "TO PRINT A CALENDER OF ANY MONTH, ENTER THE MONTH AND
    YEAR REQUIRED"
120 PRINT
130 PRINT "IF THE WHOLEYEAR IS REQUIRED THEN ENTER MONTH 0."
140 PRINT
150 PRINT "TO END, ENTER MONTH 0, YEAR 0."
160 UNIT -2: PRINT
170 INPUT "MONTH NUMBER?"  ;M
180 PRINT
190 INPUT "YEAR NUMBER?" ;Y
200 PRINT
210 P=0
220 Z=0
230 PRINT % "C"
240 PRINT: PRINT: PRINT: PRINT
250 IF M < 0 OR M > 12 OR Y <-25000 OR Y >20000 THEN GOTO 170
260 IF M = 0 AND Y = 0 THEN END
270 PRINT "DO YOU REQUIRE A HARDCOPY"
280 K=KEY [0]: IF K = 89 THEN P = 1: GOTO 310
290 IF K - 78 THEN P = 0: UNIT -2: GOTO 320
300 GOTO 280
310 UNIT 2: PRINT
320 REM P IS THE PRINT FLAG
330 IF M = 0 THEN Z = 1
340 REM Z = REPEAT FLAG TO PRINT WHOLE YEAR
350 IF M = 0 THEN M = 1
360 M1 = M
370 Y1 = Y
380 REM M1 AND Y1 SAVE M AND Y
390 I = Y
400 $A = "A.D."
410 IF Y > = 0 THEN GOTO 450
420 $A = "B.C."
430 1 = -1
440 Y = Y + 1
450 REM
460 REM NEW PAGE AFTER JUNE IF PRINTING YEAR
```

```
470 PRINT TAB (10); $M [M,0]; TAB (25); 1; $A
480 GOSUB 850
490 I = J
500 REM
510 PRINT TAB (10);"SUN MON TUES WED THU FRI SAT"
520 M = M + 1
530 IF M > 12 THEN Y = Y + 1
540 IF M > 12 THEN M =1
550 GOSUB 850
560 N = J - 1
570 J = 1 - INT [1/7] * 7 + 1
580 IF J = 7 THEN J = 0
590 J = J * 4 + 10
600 K = 1
610 IF Y <> 1752 OR M <> 10 THEN GOTO 670
620 PRINT TAB (J); "1 2";
630 K = 14
640 J = 26
650 N = 30
660 REM 1752 WAS A CHANGE FROM JULIAN TO GREGORIAN CALENDER
670 FOR I = K TO N
680 PRINT TAB (J-1);
690 IF 1 <10 THEN PRINT " ";
700 PRINT 1;
710 J = J + 4
720 IF J > 36 THEN PRINT
730 IF J > 36 THEN J = 10
740 NEXT I
750 PRINT  : PRINT
760 Y = Y1
770 M = M1
780 IF P =1 THEN GOTO 800
790 WAIT 100
800 IF Z = 1 THEN M = M + 1
810 IF M < 13 AND Z = 1 THEN GOTO 340
820 GOTO 160
830 REM
840 REM CALC ROUTINE
850 K = Y + 4712
860 J = INT [K/4] + 365 * K
870 N = 30.6 * M -32.3
880 IF M > 2 THEN GOTO 910
890 N = N + 2.3
900 IF K - INT [K/4] * 4 = 0 THEN J = J - 1
910 J = J + INT [N + 1]
920 IF J <=2361220 THEN RETURN
930 K = Y - 300
940 IF M < 3 THEN K = K - 1
950 N = INT [K/100]
960 J = J - INT [0.75 * N] - 1
970 RETURN
```

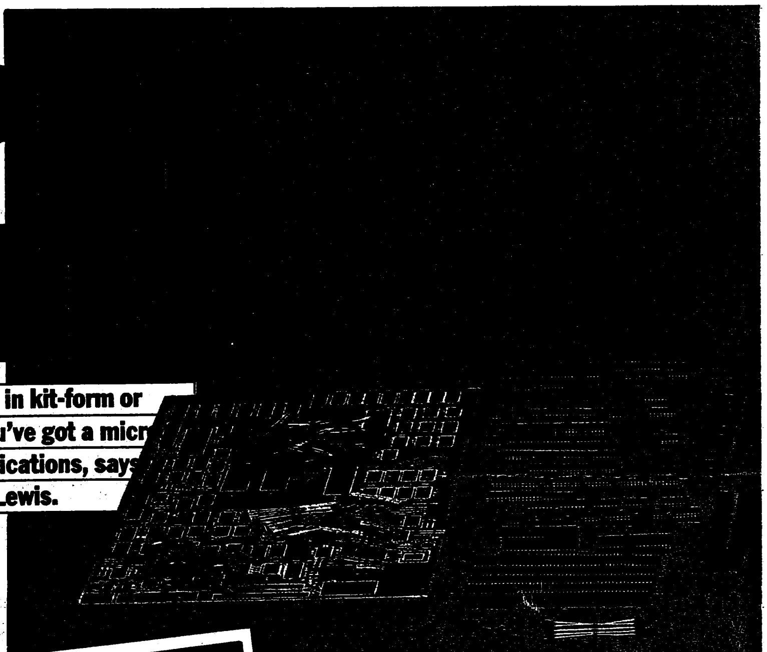
Also find enclosed the same wrinkle applied to relative files. It does not allow you to put the $ sign back on but does allow you to save strings of unlimited length, whereas before you could only save a maximum string length of 22 characters (as discovered by John Mackenzie).

```
;; UNIT -2

LIST 10 TO 80

10  DIM $S[10,2], $ F [10,4], $A1 [10,4], $A2 [10,3], $A3 [10,3],
    $A4 [10,2], $T [10,2], $D1 [10], $D2 [10], $D3 [10]
20  V = 0: G = 0: R = 1: W = 1: J = 0:
30  ERROR 60
40  OPEN 1, "CADATA2" R, CRE
50  GET R, V, S [J,0], F [J,0], A1 [J,0], A2 [J,0], A3 [J,0], A4
    [J,0], T [J,0], D1 [J], D2 [J], D3 [J]
60  POP
70  CLOSE R
80  PRINT % "C"

LIST 530 TO 620

530 OPEN 1, "CADATA2", R
540 ERROR 580
550 J = 0
560  GET R, V, $S [J,0], $F [J,0], $A1 [J,0], $A2 [J,0], $A3
    [J,0], $A4 [J,0], $T [J,0], $D1 [J], $D2 [J], $D3 [J]
570 GOTO 560
580 POP
590 FOR J = 1 TO B
600 PUT R, V, $S[J,0], $F [J,0], $A1 [J,0], $A2 [J,0], $A3 [J,0]
    $A4 [J,0], $T [J,0], $D1 [J], $D2 [J], $D3 [J]
610 NEXT J
620 CLOSE R

PRINT: UNIT -2

LIST               Relative File Example

10  DIM $A [9], $B [9]
15  ERROR 35
20  OPEN 0, "TTTT", R, CRE, 70
30  GET R, 1, B [0]
35  POP
40  CLOSE R
110 INPUT "ENTER $A"; $A [0]
210 OPEN 0, "TTTT", R
310 PUT R, 1, A [0]
410 CLOSE R
510 OPEN 0, "TTTT", R
610 GET R, 1, B [0]
710 PRINT $B [0]
810 CLOSE R
910 END
```

HARDWARE: CORTEX 2

ZIP K

Buy the Cortex 2 in kit-form or ready-built and you've got a micr with great specifications, says Brendin Lewis.

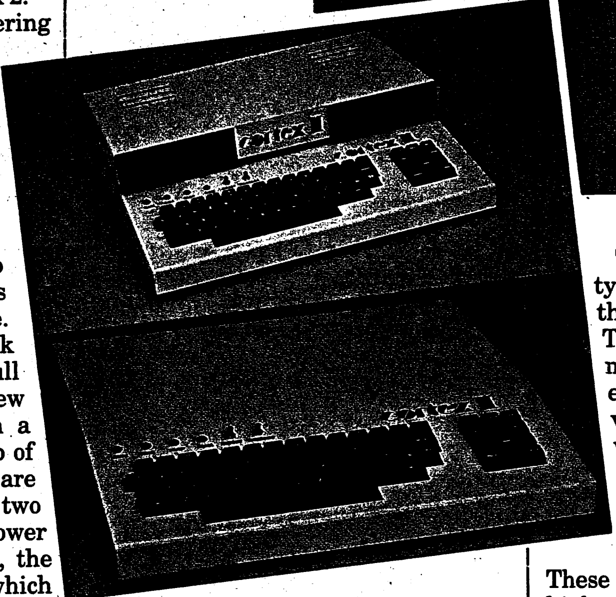


Most, if not all electronics enthusiasts will have heard of Powertran Cybernetics (previously Powertran Electronics). Over the past few years they have produced many high-quality kits.

In 1982 Powertran released the Cortex, a low cost 16-bit micro with a real sting in the tail. On specifications alone, the system is up there with the best. Based on the Texas 9995 microprocessor, running at an astonishing 12MHz, the Cortex out-benchmarks almost all the popular microcomputers. Other features include 64K of RAM, high-resolution colour graphics with sprites, a floppy disk interface, serial and parallel ports, and an expansion bus known as the Ebus.

A few months back the Cortex was withdrawn for a refit, now the system is sold with a new smarter, slimline look and has been christened the Cortex 2.

For those of you adept with a soldering iron, good eyesight and a great deal of patience, the system comes as a kit. For those more prone to melt the table top rather than the solder, a ready-built and tested version is also available.



First impressions

The changes made to Cortex 1 to produce Cortex 2 are apparent as soon as you set eyes on the machine. The new model has a slimline look solely due to the removal of the full height floppy disk drives. The new model has half height drives in a separate cabinet which sits on top of the main unit. These drives are connected to the computer via two leads. The first is the low voltage power lead which plugs in at the back, the second is the 34-way ribbon cable which plugs in the side.

The cabinet is made of sheet metal, sprayed in light grey. The metal adds to the weight but still leaves it lighter than some so-called portables. The one thing to be said for metal cases is the added strength it provides.

Hardware

Towards the front of the machine are situated the keyboard, a number of LED indicators and two push button switches. The LEDs are marked; RUN, IDLE, MAP, and TIME and are dealt with in the 'In use' section as are the two switches marked RESET and RESTART.

The keyboard itself is of a simple construction with the minimum of frills. The main section contains all the standard alphanumeric keys, 55 in total. There is also a nine-key keypad containing cursor keys and various editing keys which are used by the Basic line editor.

Various connectors are apparent around the outside of the case but, unfortunately, there are no markings on the case itself to describe their function.

Working from the circuit diagrams and the PCB overlay diagrams, the following should be correct descriptions. On the right-hand side front are located the two connectors which make up the serial system. A 5-pin DIN socket is used to communicate with the cassette recorder (Powertran recommend a WH Smith model).

A 25-way 'D' type connector is used to interface the unit with any standard RS232 serial device. A 34-pin IDC conncetor is used to connect the new disk unit. This connector should really be on the rear of the system with the power supply connector and not in its current position on the right-hand side. Powertran supplies two types of disk unit for the Cortex with 125K or 1Mb unformatted capacities offered on single sided single density or double sided double density drives respectively.

At the rear of the unit are located the video outputs for both the UHF (a TV set) and RGB (optional extra) displays, in addition to the disk power supply connector.

The only other connector is the 'D' type for the Ebus and is situated this time on the left of the machine. This allows the machine to communicate with standard Eurobus expansion boards. On previous versions of the Cortex, the Ebus was not available due to design problems but I am assured that this one works properly.

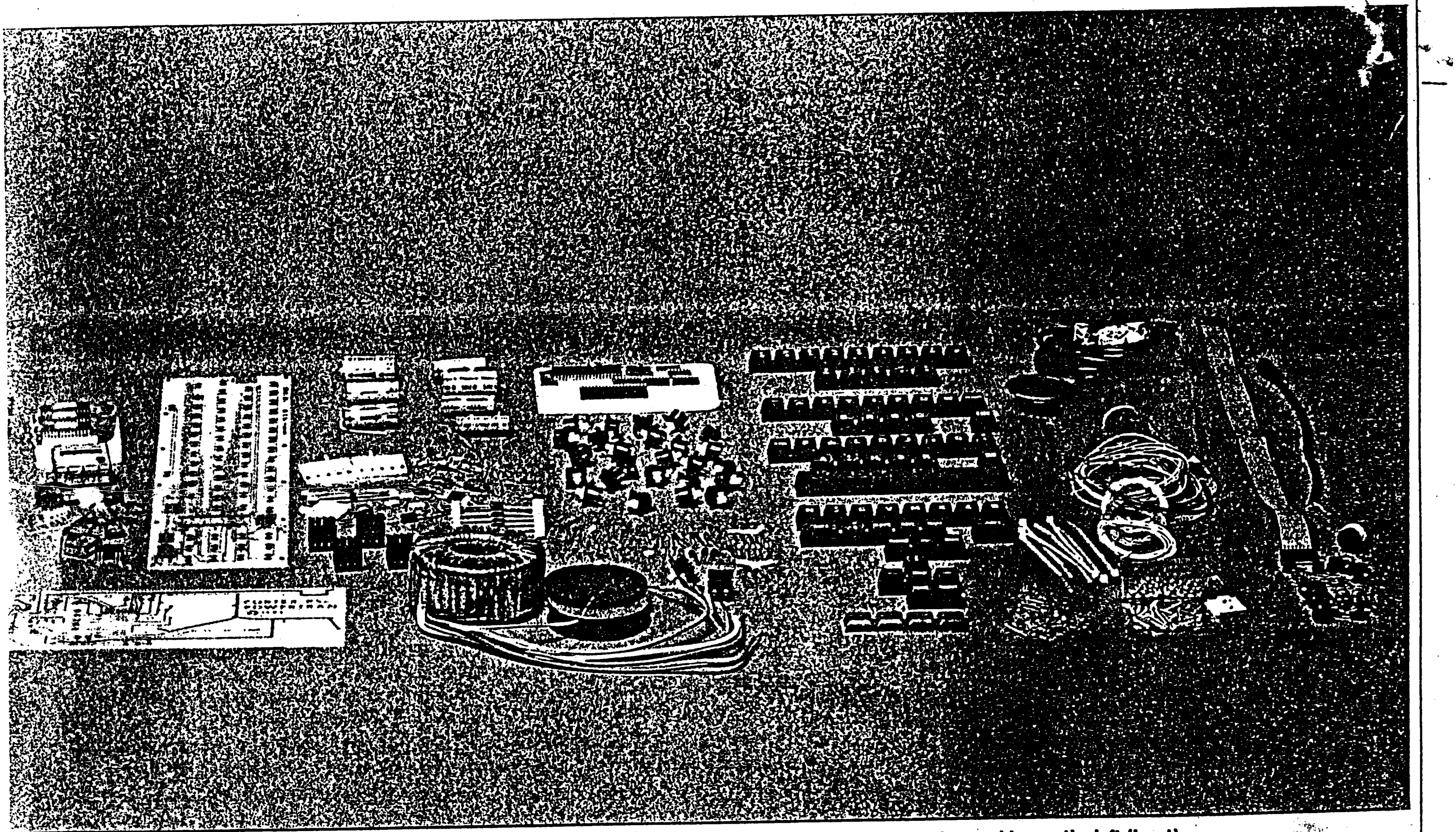
Documentation

These manuals are certainly not of the highest standard but they are readable and that's what counts. Three manuals make up the system documentation: a user manual, the disk operating system manual, and a construction manual if you're buying in kit-form. The user manual is written on the assumption that you actually built the system. No mention is made of the connectors or indicators which is a shame because some people will buy the ready-built version and will not be familiar with the system. What it does have is an in-depth description of Cortex Basic.

The construction manual is a photocopy of the original article which appeared in *Electronics Today International* as appears to be the norm with Powertran kits. This is quite adequate as it not only deals with descriptions and construction but also has an item on 'how it works' for each section of the project.

Construction

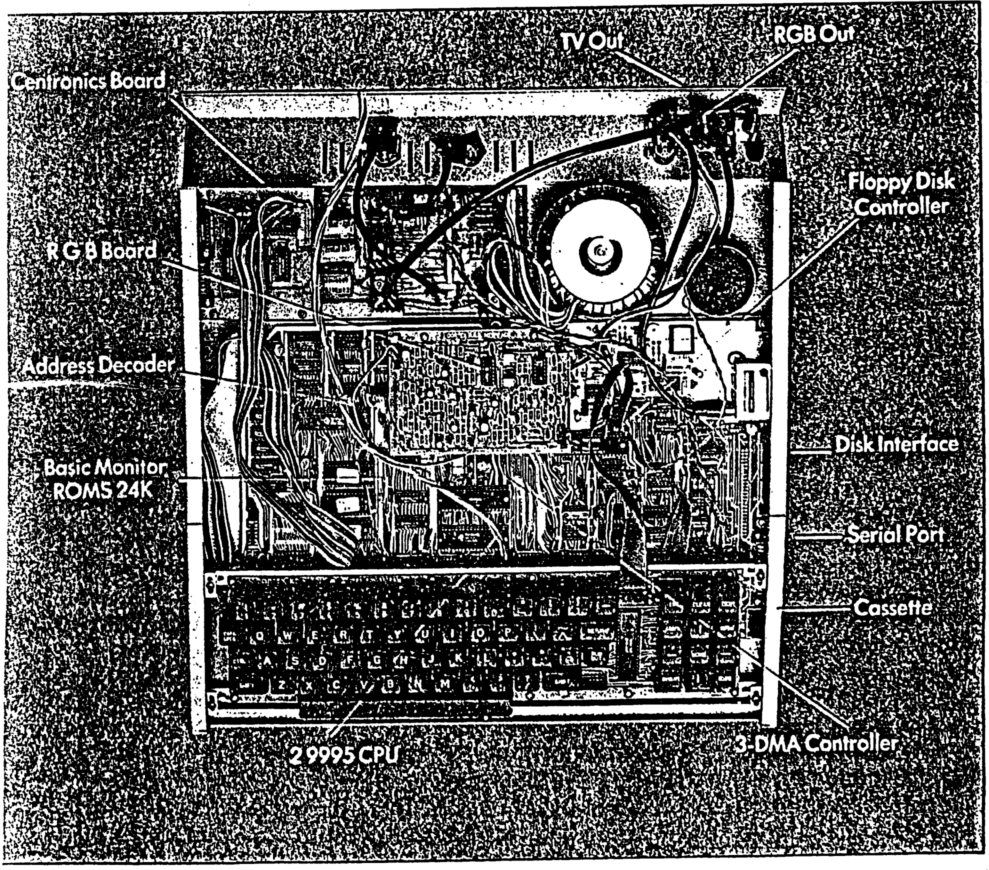
There is no reason why a system built from a kit should look any different from a professionally built machine. All components supplied to make up the



If you buy the Cortex 2 in kit-form, you will have to turn the resistors, capacitors and chips (above), into the machine on the left (inset).

Cortex are of a high standard and if constructed properly give the desired high-quality appearance.

But taking on the task of actually building a kit that's as complex as the Cortex involves certain skills not normally associated with the average computer end user. The most important of these is the ability to use a soldering iron; though identifying and handling of components is also important. Another factor, almost as important as the soldering skills, is patience. Don't expect to sit down and build the system in one session; it is possible but not recommended. A simple rule of thumb to follow is that the more time you spend on building the system, the more likely you are to have a finished product that works.



One of the first things you notice when looking at the Cortex PCB is that all the integrated circuits are socketed. There are two schools of thought on this subject (ignoring the obvious increase in cost) for it can be argued that sockets don't provide a good electrical contact for long periods. This is true, but in this case, matters such as ease of construction, repair and modification of the board far outweigh the disadvantage of having to re-seat a chip occasionally.

While building a project from a kit, it's surprising how much one learns about that project. This knowledge then builds confidence to take on the task of taking on a hardware modification. A socket system gives such an option and is a facility rarely available to the average user.

The construction of the review system was excellent except for one small detail. Whoever built this system broke one basic rule of assembly, *ie* to make sure all cables which connect between the main unit and the lid of the system are long enough for the lid to be removed and placed to one side. Of course the system must still be able to operate with the lid removed. In this case, only the LEDs, reset buttons and two power supply regulators are mounted on the lid and it was the cables running to the LEDs that was cut too short.

Processors

The system itself is based on the Texas

29)

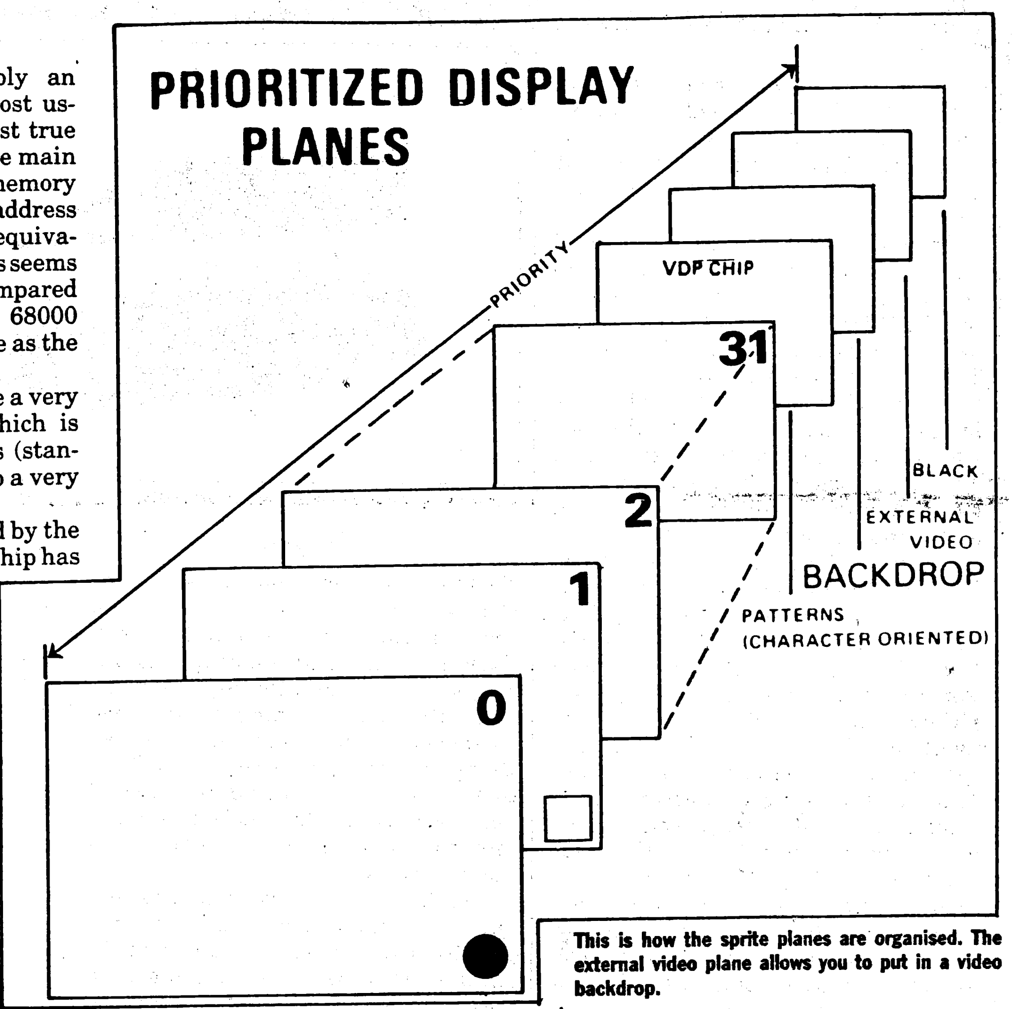
HARDWARE PRO-TEST: CORTEX 2

◀ 27

Instruments TMS9995, probably an unfamiliar microprocessor to most users. It was in fact one of the first true 16-bit microprocessors. It has one main drawback which is a limited memory addressing range. It can only address 32Kwords (16-bit) which is the equivalent of 64K bytes. These days, this seems a very small amount when compared with the 8086 (1Mb) and the 68000 (16Mb) though it is still the same as the Z80 and the 6502.

It does on the other hand, have a very high clock speed of 12MHz which is faster than all these processors (standard models only) which leads to a very fast machine.

Video on the Cortex is handled by the TMS9928 video processor. This chip has its own 16K of memory leading to a graphics resolution of 256×192 with 16 colours on the screen at one time.



This is how the sprite planes are organised. The external video plane allows you to put in a video backdrop.

Also implemented on the 9928 is a sprite capability with up to 256 being allowed. These sprites can be defined as an 8×8 or 16×16 (64 sprites only) pixel grid. An option to magnify these is also available. The command MAGØ maps each sprite pixel to one pixel on the screen, while the command MAG1 maps each sprite pixel on to a 2×2 grid on the screen thus allowing a single sprite to fill a 32×32 grid on the screen. As is usual with sprites, software is available to check for collision between sprites while hardware within the video processor takes care of which sprite is 'in front' of any other when displayed.

One very interesting feature of the graphics processor is that one colour is defined, not as a colour, but as transparent. This allows the background colour to be seen 'through' the sprite.

The idea of one image being in front of another is possible because of the way in which the 9928 implements its graphics. The screen can be thought of as 36 planes on top of each other—rather like placing 36 photographic slides on top of each other and looking through them. Thus a sprite placed on plane 8 would cover an image of a sprite on plane 9.

One of the 36 planes is set aside for an external video expansion board. This board mixes a video signal from, for example, a video recorder with that of the computer. Thus it is possible, albeit difficult, to do things such as make a car in a video game hit a real wall.

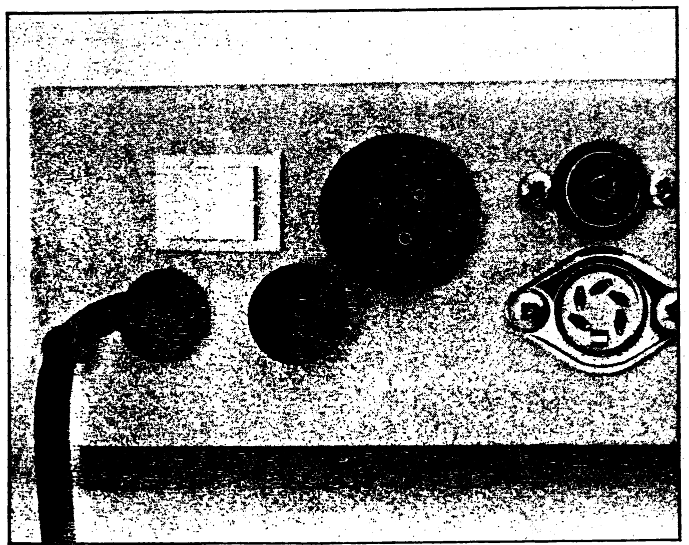
Because the graphics memory is separate from the user memory the system still provides the full 64K bytes for use. Well, nearly all, as a small area of memory is set aside for memory mapped I/O allocation.

Firmware

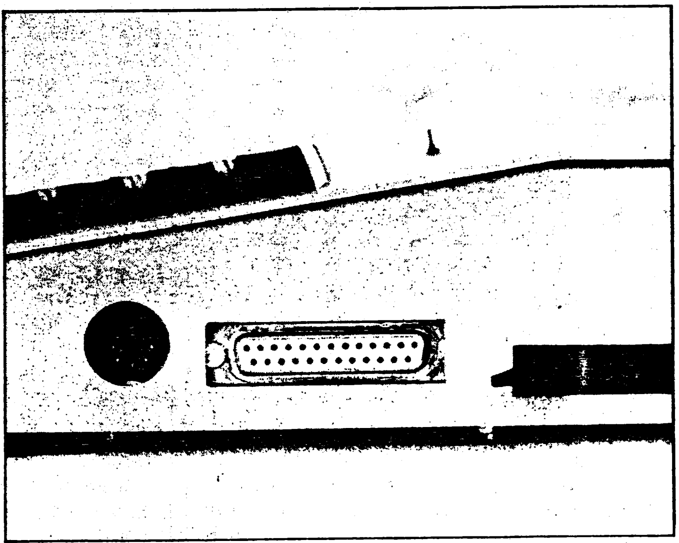
Basic comes as standard on the system and is stored in ROM along with the system monitor. All firmware is held in 3×8K EPROMs which, again, are not part of the main memory map. Instead, these EPROMs are known as phantom memory.

When the system is first powered up, the dynamic RAMs are checked for

30 ▶



None of the sockets is labelled.



The cassette and RS232 interfaces and the expansion bus are on the side.

◀ 29

correct operation and then the contents of the EPROMs are copied into RAM. This allows far greater system flexibility because Basic can be overwritten if the system is running 9995 machine code only.

Though Basic is the language fitted with all new systems, it is possible to change the language simply by replacing the EPROMs. Two other languages are presently available for the Cortex. The first is fig-Forth (available from Lombard Systems of Bedford). The language comes on two 8K EPROMs, which replace the first two Basic ROMs. To enhance the Forth package, a utilities disk is also available containing an editor and various I/O utilities.

For those who don't wish to replace the Basic ROMs permanently, Forth is also available on an auto run disk with all the utilities included. With this version, the Basic 'boot' command is used to load Forth on top of the Basic interpreter. UCSD Pascal P-code system is also available. This includes the operating system, full screen editor, assembler, compiler and debugger. Further extensions allow multi-tasking and support for extended addressing via the Ebus which will allow a full megabyte of memory to be accessed. All very nice, I hear you P-code followers thinking. True, except that it weighs in at £535. With the standard Cortex kit at £300, it does seem a bit excessive.

Software

A number of companies are now offering software which will run on the Cortex, and Powertran itself has commissioned an independent software house to write a new disk operating system.

Microprocessor Engineering (0703-775482) offers a number of software packages which run on the Cortex. MDEX is a disk operating system with a disk-based version of Basic which is similar to Microsoft Basic. EDIT is an editor based upon a number of mainframe-based editors including SOS. RESCUE is a set of three programs for processor, memory and disk diagnostics. It also includes a disk editor/ recovery program. SPL is a systems programming language similar to 'C' for which a ROM-based nucleus is being developed.

This is by no means a full list and needless to say, a number of games are also available. No doubt, more will be made available after the competition that Powertran ran in its August newsletter. Though this was the first Cortex user group newsletter it seems that they will run about every three months, as the next was due in December.

Topics include all the latest updates in both hardware and software, plus hints and tips from present Cortex owners. Well done, Powertran—I'm all in favour of companies that keep close ties with their user groups.

The dual disk drive is designed to sit neatly on top of the machine.

In use

There should be few problems with using Cortex Basic For the first time user. But those familiar with standard Microsoft Basic may take a little time to get used to it. Contained within the Basic is a line orientated editor which is initiated whenever a syntax error is encountered. Being brought up on a version of Basic totally devoid of editing facilities, I normally retype the whole line automatically so using this type of editor is quite a luxury.

The Basic itself is excellent allowing auto line numbering, renumbering, definition of sprites and allowing 16 parameters to be passed to a machine code routine, to name but a few of its many facilities.

On the system disk supplied by Powertran were three simple demo programs, all using sprites. The sprite moving sections of the demos were smooth and fast considering that they were in Basic, obviously another pointer to the overall speed of the system.

When coming out of a program I noticed that the text on the screen was different to that displayed when I loaded it. This is because the Cortex works in two distinct modes, text and graphics. When in text the normal 24 × 40 display is used, but when in graphics mode the characters are actually plotted onto the screen, thus allowing only 24 × 32 characters on the screen. Typing the command 'text' is all that's required to swap back to text mode.

Though the system itself is fast, the disk drives themselves are quite slow. Together with this, the disk operating system is cumbersome and long-winded to use. I won't delve any deeper into the workings of this particular DOS as it is in the process of being replaced.

As mentioned previously, there are a number of LED indicators on the top of the case. The first two show how much of the time the system is actually running code and how long it is sitting idle. The MAP LED shows when memory mapped I/O is taking place. The final one, marked TIME, flashes continually and I could find no details about it. None were given in the construction manual so I assume this was not available on the Cortex 1.

Verdict

Building from a kit, in this case, offers a cheap way to obtain a powerful and versatile micro, but it must be said that it is not just putting pieces together jig-saw fashion.

Buying the Cortex 2, in whatever form, must be looked on as something of a challenge. The reason for this is that a lot of software development has to be done by the user because it cannot be bought off the shelf. Incompatibity is the main cause. Though there are a small number of other 9900 series machines on the market, each is a small fish in a large ocean compared with the likes of Commodore, Acorn and Sinclair. So don't expect a new game or package to appear every week, because it won't. Software is being written, but not in vast quantities.

Though the Cortex is primarily a home computer, there is no reason why it should not become a small business micro if relevant software were to become available.

The disk capacity is large enough, the speed is more than ample, and printer ports and high-resolution graphics are also included. ◩

Specifications

Price	Basic kit £299, built £399, £1,195 with all the options
Processor	TMS9995 12MHz
ROM	24K for Basic and Monitor
RAM	64K plus 16K for graphics
Screen	24 × 40 text, 256 × 192 graphics
Keyboard	64 keys including nine key keypad
Interfaces	RGB, Centronics, Ebus, Floppy disks external video input
Distributor	Powertran Cybernetics 0264-64455