CORTEX USERS' GROUP

NEWS LETTER 1 AUGUST 1984

POWERTRAN NEWS

HARDWARE-

As you might have seen from the Electronics press, there have been one or two developments for the Cortex in recent weeks.

Most striking is a redesign of the computer's cabinet. The Cortex II, as we've called it, is a slimline, stone-coloured machine without the bulky built-in disc housing. A matching cabinet for the half-height disc drives can sit on top of the computer or be sited separately.

For those of you who want to update your machines to the new look, we will be supplying the new cabinet and modified PSU board at a special price of £45.00+ VAT

We have finally managed to organise a working E-Bus! £17.50+VAT. The modified version will accept memory expansion and I/O cards. A centronics printer interface has already been designed and published (in Electronics Today International). £35.00+VAT.

RGB interface kits for linear monitors are already installed in quite a number of your Cortices - I hope they all work as well as ours does! £28.50+VAT.

SOFTWARE-

We have commissioned an independent consultant to write a new DOS for the Cortex. This project is now well advanced and we expect to be able to announce a (reasonable!) price in the near future. The system will be available on either single-sided or double-sided 5¼" disc.

Microprocessor Engineering are still working on Cortex software and have an ever increasing list of titles available. These include programmers utilities, business software and quite a few games. Phone Southampton 780084 for details.

Lombard Systems is another software house which has started producing material for the Cortex. Their first offering is a complete FORTH in two EPROMS which plug straight into the main board (replacing two of the BASIC EPROMs supplied with the computer. Write to Lombard Systems, 18 Lombard Street, Lidlington, Bedford MK43 0RP.

Your Own Pages

(The following letters and programs have been sent in by Cortex users - Powertran Cybernetics Limited cannot accept liability for their contents).

ROBERT LEE
9-11 Seaside
Eastbourne
East Sussex

TERMINAL EMULATOR PROGRAM BY R.M.LEE

This terminal emulation program enables your CORTEX computer to become a DUMB terminal. Connections are made via the RS232 serial interface. The program also scans the cassette port for INPUT, so that tape ASCII files can be down loaded. The cassette BAUD rate must be slower than the transmission speed. The listing below was sent 26 miles to the VAX computers at Brighton Polytechnic using this program. The BAUD rates have to be defined before the program is run e.g.

```
10 BAUD 2,1200
20 BAUD 3,600
```

The program resides at locations 6000H to 6056H, but is totally relocatable.

```
6000 0300 LIMI >0003       ;Disable LEVEL 4 interupt
6004 020C LI   R12,>0010   ;Set CRU base for keyboard
6008 1DF9 SBO  -7          ;Reset keyboard flag
600A 1FFD TB   -3          ;Test keyboard interupt flag
600C 130E JEQ  >602A       ;and skip next section if not set
600E 04C0 CLR  R0          ;
6010 3600 STCR R0,8        ;Store keyboard character in R0
6012 9800 CB   R0,@>5541   ;Test for ESCAPE (1BH)
6016 1601 JNE  >601A       ;
6018 0380 RTWP             ;Return if ESCAPE detected
601A 1EF9 SBZ  -7          ;Keyboard reset strobe
601C 1DF9 SBO  -7          ;
601E 020C LI   R12,>0080   ;Set CRU base for RS232 port
6022 1D10 SBO  16          ;Enable output
6024 1F16 TB   22          ;Test RS232 for end of transmission
6026 1601 JNE  >602A       ;Skip if not ready to send
6028 3200 LDCR R0,8        ;Load port with new data
602A 020C LI   R12,>0080   ;
602E 1F15 TB   21          ;Test RS232 for input buffer full
6030 1603 JNE  >6038       ;Skip if not full
6032 3601 STCR R1,8        ;Store input character from port
6034 1E12 SBZ  18          ;Reset input buffer
6036 0F01 WRIT R1          ;Write character to screen
6038 020C LI   R12,>0180   ;Set CRU base for cassette
603C 1F15 TB   21          ;Test cassette for input buffer full
603E 16E0 JNE  >6000       ;restart if not full
6040 3601 STCR R1,8        ;Store input character from cassette
```

```
6042 1E12 SBZ  18          ;Reset input buffer
6044 9801 CB   R1,@>0082   ;If character is Line Feed then do
6048 13DB JEQ  >6000       ;not send
604A 020C LI   R12,>0080   ;Set CRU base for RS232
604E 1D10 SBO  16          ;Request to send to RS232 port
6050 1F16 TB   22          ;Restart if not ready to send
6052 16D6 JNE  >6000       ;
6054 3201 LDCR R1,8        ;Send character to RS232 port
6056 10D4 JMP  >6000       ;Continue
```

Robert has also written an interesting game programme. It's a 3D maze and he wants money for it - how much, you'll have to ask him yourself.

Brendan Hanrahan
20 Elm Mount Park
Beaumont
Dublin 9
EIRE

Brendan's a bit of an all-rounder - he tinkers with the main board as well as the keyboard - judge for yourselves.

3¾K OF BULLETPROOF RAM, FOR FREE!

The Memory Enable logic of the Cortex defines locations above F000 to be memory-mapped area (except for the on-chip RAM between F000 and F0FB). Only the first 200H bytes are decoded on-board, the rest being left for the constructor to play with. But honestly, who is going to need 4K of memory-mapped I/O? The logic checks if the first nibble (most significant four bits) of the address is F (all high) via IC23b and hence produces the MEMI/O (active low) signal. If we further check that the next three bits are low then we limit the MEMI/O to addresses between F000 and F1FF leaving everything above F1FF free for use as RAM. This extra check is already done by IC29b and we can use this, in conjunction with IC23b, to produce the new MEMI/O signal. A few more gates are needed but thankfully these can be obtained from the many spare gates available on the main board. A few cuts and a few wire links are needed. The entire job takes about ten minutes. Here is a detailed account on how to proceed.

1) Remove the main board and place it face down.
2) Join IC21 pins 9 and 10 with a solder bridge and join this bridge to IC23 pin 8 with a wire link.
3) Join IC21 pin 8 to IC31 pin 1 with a 17cm link.
4) Join IC23 pin 5 to IC31 pin 3 with a 15cm link. Continue this joint to the track in the middle of IC30.
5) Join IC31 pin 2 to IC31 pin 10.
6) Secure the long links with tape and turn the board over (face up).
7) A track leads from IC23 pin 8 towards IC10. Cut this track 1cm from IC23.
8) Remove IC23. A track leads from pin 5 to pin 8. Cut this track.
9) Replace IC23 and reconnect the main board.
10) Turn the power on. If you have made no mistakes the Cortex will enter BASIC as usual.

You will now find that you can put information into memory between F200 and FFF8. This new memory area has one major advantage: it is unaffected by a Reset, hence the name "bulletproof RAM". The new RAM is useful for testing machine-code programs because if they crash the Cortex you can Reset and try again without having to reload.

50% FASTER MACHINE-CODE PROGRAMS

The TMS9995 has 250 bytes of on-chip RAM between locations F000 and F0FB inclusive. Almost all of this fast access RAM is unused by the Cortex and is the perfect place to put your registers. The normal default WP for the Cortex is ED46 which puts the registers of your CALLed program in ordinary memory. To change this, one only has to make a single alteration to the system software. The alteration can be made from BASIC with the instruction: MWD(1F20H)=0F020H. Location F020 is chosen because it avoids any areas used by system software.

A FEW EXTRAS

Cortex users may find the following pieces of information helpful. Locations EE36 and EE37 hold the current horizontal and vertical positions of the cursor. GRAPH-RUB causes all units to be disabled except for unit 1. CRB(6)=1 causes a blip to be heard. Finally, there are several undocumented MIDs, some useful ones are:

opcode		function
0001		WRIT R0
0006 xxxx		MSG @>xxxx
0009		WRIT > (write immediate)
000B	┐	Retrieve character from keyboard buffer.
xxxx	├──	If null then execute xxxx
yyyy	├	If not null then execute yyyy
zzzz	┘	If ESC then execute zzzz
		xxxx, yyyy and zzzz should be JMPs.
000C		READ R0 (used to get characters from the tape cassette when the keyboard is disabled)
000F		TEXT (as in BASIC)

There are a few other MIDs yet to be discovered.

KEYBOARD CONTROL

Ever wanted to completely disable the keyboard and still be able to access the screen? The UNIT command will not help. However, all is not lost. There are two ways of controlling the keyboard:

1) CRB(1)=0 switches the keyboard off and
   CRB(1)=1 (or a reset) switches it on again.
2) if you make location ED72 non-zero then only the ESC key will be accepted from the keyboard. This is used by the SAVE and LOAD routines.

Another thing many users may want to do is check if a key is being continuously depressed. According to data sheets the 2376 encoder produces a random value if no key is depressed and the ASCII code of the depressed key in all other circumstances. With this in mind we can produce a routine to test if the 2376 output is static:

```
K=CRF(0):Q=KEY(0):IF K=CRF(0):K=INT(K/256)
ELSE K=0
```

The KEY function is used in the above routine to empty the buffer. You can test if the GRAPH key is pressed by using CRB(15). Users will find this useful for games.

BUGS AND THINGS

The COL function does not work. If the pixel is set the COL function returns the colour last plotted not the colour of the pixel.
Did you know you could renumber programs backwards!
When you type * as a command you get a 'required hardware not found' error. What hardware is the Cortex looking for?
Type in the following line: 10 FOR X=1 TO -32768 STEP -1
and then list the line. Interesting, eh?
Problem: My screen is shifted to the left so that I loose the leftmost character in GRAPH mode. The fault is on the Cortex main board and my monitor has no horizontal control. Any solutions?
Finally, how many Cortex users are there?

Mr M D Rudnicki
32 Kings Drive
Bognor Regis
West Sussex

MD has a sound generator board and an EPROM selector board.
He's also worked out a solution to the COL problem:

True Colour of a Pixel in Graph Mode

Quite a few people may have noticed that COL (X,Y) doesn't return the true colour of a pixel. Instead, it returns the current foreground colour if a pixel is present, else it returns the current background colour.

Eg. Type in Colour 7, 1 : GRAPH : PLOT(100, 100): COLOUR15,7: PRINT COL (100,100)
Prompts the reply 15 instead of 7.

In graph mode the 16K VRAM contains, amongst others, 2 tables 6K Bytes long. The first resides from >ØØØØ to >17FF and the second from >2ØØØ to >3FFF. It was found that the first table contains the pixel pattern and the second block the colour information.

Data in both of these are stored in blocks of 8, each representing 1 screen character. If X and Y are the co-ordinates of the pixel, then the VRAM address is worked out from the following:

Address = INT (Y18) x 256 + 8 x INT (X18) + MOD (Y,8)

There are 32 screen columns each take up 8 Bytes 32 x 8 = 256, so for every screen line down a group of 256 locations has to be jumped over. INT (Y/8) gives the actual screen line and multiplying this by 256 selects the correct group. The term 8 x INT(X/8) selects the correct group of 8 Bytes, whilst MOD (Y,8) selects the correct Byte. Using this idea a temporary "COL" routine can be 'Called' until the ROM is changed!

True Colour of Pixel: A=Ø: CALL 6218H, X, Y, ADR, (A)

X, Y = Screen co-ordinates.

```
100 DATA      1728, -1Ø24Ø, -38Ø7, 1728, -1Ø24Ø, -38Ø7, ØC41ØH, -12192
110 DATA      -38Ø8, 0C41ØH, 1115, Ø, -16192, -16127, 2353, 2689
120 DATA      580, 7, -245Ø8, 576, 248, -24512, -16127, -16383
130 DATA      1696, 25Ø88, -16Ø63, - 1638Ø, 6Ø8, 8192, 1696, 25Ø88
140 DATA      579, 7, 643, Ø, 4867, 2581, 1539, 4346
150 DATA      581, Ø8ØØØH, 645, Ø8ØØØH, 5633, 2369, 577, 384Ø
160 DATA      1474, 141Ø, -11135, 896, Ø, Ø, Ø, Ø
170 FOR I =   62ØØH to 626EH STEP 2
180 READ A :  MWD (I) = A: NEXT 1
```

The above is the data for a machine code routine which gives the true colour of a pixel. In the call statement A, X, and Y can be any variables or expressions. The disassembly is as follows:

```
GET BYTE    >ØØ SWPB RØ            ;Select correct byte (VRAM address)
                MOVB RØ, @>F121    ;Store in VDP
                SWPB RØ            ;Select other byte
                MOVB RØ, @>F121    ;Store in VDP
                MOV * RØ, *RØ      ;Delay
                MOVB @>F12Ø, R1    ;Get contents of location
                MOV *RØ, *RØ       ;Delay
                RT                 ;Return to main program
                DATA >ØØØØ
MAIN ROUTINE>6218 MOV RØ, N3       ;RØ = X, R3 = X
                MOV R1, R4         ;R1 = Y, R4 = Y
                SRL R1, 3          ;R1 = INT (Y/8)
                SLA R1, 8          ;R1 = 256 x INT (Y/8)
                ANDI R4, > ØØØ7    ;R4 = MOD (Y,8)
                A R4, R1           ;R1 = R1 + R4 (=256xINT(Y/8) +MOD(Y,8)
                ANDI RØ, >ØØF8     ;RØ = MOD (X,8)
                A RØ, R1           ;R1 = R1 + RØ = VRAM Address
                MOV R1, R4         ;R4 = R1
                MOV R1, RØ         ;RØ = R1
                BL Q>62ØØ (GET BYTE);Get VRAM Byte
                MOV R1, R5         ;Store it in R5 (Pixel data)
                MOV R4, RØ         ;Restore VRAM address
                ORI RØ, > 2ØØØ     ;Move onto colour table
                BL @ >62ØØ(Get Byte);Get colour byte
                ANDI R3, >ØØØ7     ;R3 = MOD (X,8) = Pixel No
LOOP >6244      CI R3, >Ø          ;R3 = Ø?
                JEQ >625Ø          ;Jump if yes
                SLA R5, 1          ;Else move onto next pixel
                DEC R3             ;Decrease pixel No&
                JMP >6244 (loop)   ;Find out whether it is zero yet
 6250           ANDI R5,> 8ØØØ     ;() the pixel set
                CI R5, >8ØØØ       ;_____" "_____
                JNE >625C (No Shift); Yes but we need to get
                SRL R1, 4          ;the right colour data
NO SHIFT >625C  ANDI R1, >ØF ØØ    ;Get colour data
                INCT R2            ;Get correct store location
                INC R2             ;R2 = ADR (A) + 3
                MOVB R1, * R2      ;Store colour in variable
                RTWP               ;And return to program
```

Absolute understanding of the routine is not expected, however it does work. Note that whilst the Cortex works out its colours in 8 x 8 pixel block, the VDP actually uses 8 8 x 1 pixel blocks - with correspondingly higher resolution.

Paul Edwards
11 St Julian Gardens
Cocherane Park
Newcastle-Upon-Tyne
NE7 7LL

Paul's sent us this M/C routine for colour graphics. He's got another one for flashing text that we didn't have room to print.

General Description

This short routine allows the BASIC programmer to use the full colour capabilities of the machine. The Cortex in GRAPH mode is initialised to what TEXAS call graphics mode 2, where it is possible to define two colours for each line of a defined shape. The colours are defined 'foreground' and 'background' for each byte of the **shape** and 8 bytes are required for each shape. I have found **strings** the easiest way to define the colour table, reading bytes or words from DATA statements.

The M/C routine requires that you preset register RØ to the screen positon used to SPUT the shape, and R1 to be the address of the colour definition string. It is essential to CALL the routine AFTER using the SPUT command to place the shape.

Example sections of BASIC programe:-

```
10   DIM SCD (1)                    ;8BYTES REQUIRED
20   GRAPH
30   SHAPE Ø,..................     ;ANY SHAPE
40   A=ADR ($ CD (Ø))
50   FOR 1=Ø to 7
60   READ C
70   MEM (A+I) = C
80   NEXT I
90   PS = 24Ø                       ;SCREEN POSITION
100  SPUT PS, Ø
110  CALL 61ØØH, PS, A              ;Assumes m/c at 61ØØH
1000 DATA ØB1H, Ø41H,.....etc...    ;8 BYTES OF COLOUR
```

Program Listing

```
1                        name    graphics-colour

3                        section graph
4      6ØØØ    co)tb)    equ     6ØØØH   ;colour table (+4ØØØH)
5      F12Ø    vdp       equ     Øf12ØH  ;video processor address
6       5f2    vramset   equ     Ø5f2H   ;video vram set-up routine
```

```
()
() U 61ØØ
61ØØ Ø2Ø8 LI R8, >6ØØØ          ;prepare R8 for subroutine.
61Ø4 ØA3Ø SLA RØ, 3
61Ø6 A2ØØ A RØ, R8              ;SPUT no, *8 for vram address.
61Ø8 Ø2Ø2 LI R2, >ØØØ8          ;8 bytes to transfer.
61ØC Ø6AØ BL @>Ø5F2             ;set vram address.
611Ø D831 MOVB *R1 +, @>F12Ø    ;send colour data.
6114 Ø6Ø2 DEC R2                ;8 done?
6116 15FC JGT >611Ø             ;No, do next.
6118 Ø38Ø RTWP                  ;Finished.
()
()
```

Mark Colclough
3 Moat Road
Oldbury
Warley
West Midlands

Mark can't get his Cortex going! Is there anybody nearby who can get together with him to compare notes?

John Mackenzie
4 Werstan Close
Worcs
WR14 3NH

John has written "The Labyrinth of Trag" - first adventure program for the Cortex. It's on cassette and takes up 25k of memory. He'll let you have copy for a fiver - probably less if you can help him sort out the problem he's having with the colour on his Cortex.

Mr R Swingwood
Plough Road
Angers Green
Great Bentley
Essex
CO7 8NA

Mr Swingwood has sorted out the SWAP statement that several people have found:

There is a BASIC statement - SWAP - which is not documented, probably because it does not work! The'bug' can be corrected with the following:-

MWD (339ØH) = 13Ø8H

SWAP is used in GRAPH mode to globally change colours on the screen. It makes use of a 16 byte table, beginning at location ØA4H, that must be modified prior to executing SWAP so as to define the colour changes. In general, if we want to change colour 'm' to colour 'n'then:-

```
mem (ØA4H + m) = N
```

will set up the table entry. For example:-

```
mem (ØA4H + 7) = 13

mem (ØA4H +15) = 1

SWAP
```

would change CYAN to MAGENTA and WHITE to BLACK. It is advisable to re-set the table after executing SWAP to avoid unwanted colour changes during subsequent SWAP's. For example:-

```
mem (ØA4H +  7) = 7

mem (ØA4H + 15) = 15
```

As it stands, SWAP will change both foreground and background colours over the entire screen but it's a simple matter to change that!

The following will suppress any background colour changes by SWAP:-

```
MWD (337CH) = ØDØØ3H

MWD (337EH) = 1ØØ2H
```

to re-enable do:-

```
MWD (337CH) = Ø243H

MWD (337EH) = ØØØFH
```

To suppress foreground colour changes do:-

```
MWD (3386H) = ØA44H
MWD (3388H) = ØDØØ4H
MWD (338AH) = 1ØØØH
```

and to re-enable:-

```
MWD (3386H) = Ø944H
MWD (3388H) = ØDØ24H
MWD (338AH) = ØFØØØH
```

The vertical distance covered by SWAP can be limited to a range of character rows by:-

```
MWD (3366H) = 2ØØØH + 256 * SR

MWD (336AH) = 256 * NR
```

Where SR = Start row (Ø - 23)
NR = Number of rows (1 - 24)
Care must be taken that SR+NR< 25

N D Osmond
The Birches
Synwell Green
Wotton-under-Edge
Gloucestershire

Mr/Ms (?) Osmond has a Cortex with two DSDD 5¼" and has kept a file of the various problems that have been encountered together with their solutions. Write to N.D for a copy and with details of your own experiences with the MDEX operating system from MPE

Keep the letters coming in - depending on the mail, we'll be putting another Newsletter together for November/December.

GAMES COMPETITION

Even programmers serious enough to use a Cortex need to relax from time to time. Since the speed and graphics capabilities of our machine make it ideally suited for games, we feel that it is worth encouraging this area.

The ten best games sent to us before the publication of the next newsletter (late November) will win for their writers a free Cortex II update kit (worth £45.00).

Please send in your entries on tape or disc together with instructions for play, a brief description and the price you will be asking for copies.

All entries will be listed in the next issue and the winners will receive their prizes around Christmas time.

The games will be assessed by a panel of judges, whose decision will be final. No correspondence will be entered into.

Tapes and discs can only be returned if accompanied by SAE.

Copyright of all material entered in the competition will remain vested in the original writer(s).