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

13

CORTEX USER GROUP NEWSLETTER (Sept 1987)

Issue Number 13

CONTENTS

Letters.

Dennis Johnson. Porthcall

Please find enclosed details of a sound generator circuit that I have fitted to my Cortex and have been using for some time. I have written a space invaders programme using the P.S.G. controller and a short othello game for one player against another using the keyboard I will forward them if of any use.

We have included Denises P.S.G. circuit in this edition of the newsletter and look forward to publishing his other articles as soon as he sends them in. Please dont bother to ask if we want articles or programmes, just send in anything you have. Even if people do not actually want the particular programme sent in it can usually be of interest to see the programming tequniques used.

Oliver Hulme. Hednesford staffs.

Congatulations on yet another successful user group meeting on september the 5th. I for one had a very enjoyable day. As an amateur I tend to feel a little out of place with all the experts, but thankfully you did not let my ignorance show.
I would therefore like to thank all you fellow Cortexians for all the help you have given me. It's surprising how much of your know how I have managed to pick up. See you all at the next meeting.

Oliver is retired and his Cortex is the first thing he has done with electronics since the days of the valve. He has recently got E.Bus up and running and has fitted one of the new Western Digital disk controller cards. This and his P.C.B. Plot programme in this issue shows that he is getting to grips with the latest technology.

R.M.Lee. Kent.

Mr Lee has recently married and moved house so his computing has slowed down for a while. He asked us to print his new address.-

R.M.Lee, 8 Rendown Road, Lordswood, Chatham, Kent, ME5 8SG.

Also on the move is John Makenzie his new address is.-

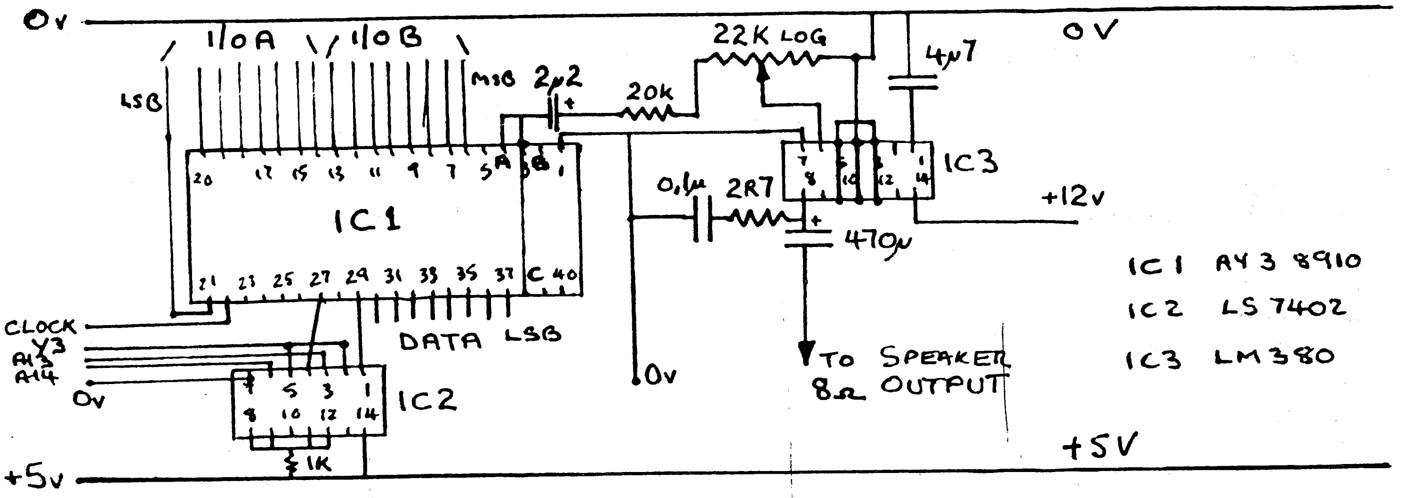
J.S.Makenzie, 20 West Road, Barton Stacey, Winchester, Hants.

MDEX.
The user group has now taken posession of about 150 disks full of MDEX software. As soon as we have sorted it out we will publish details of what is available. Rex Collins has offered to handle MDEX support for the group and Athony Rowell is in the process of generating a 4 drive version that will allow us to do disk format transfers from 40T to 80T etc. Also Nigel Osmond who uses Q Basic the Basic compiler for MDEX a lot, has offered to write some articles on how to use it. So we will be hearing more in future.

REMEMBER TO SEND IN YOUR ARTICLES FOR THE NEXT NEWSLETTER

PROGRAMMABLE SOUND GENERATOR.

By D.Johnson.



I built the above circuit on a Vero VQ board and fitted it inside the lid of my Cortex 1.
Y3 is the input from IC35 on the main board and maps the PSG to F160
A13 & A14 via the 7402 give me the necessary function codes for the PSG.
Clock must be less than 2 MHz and as I havent got a disk drive I used IC69b on the main board to divide CLK by two to give me 1.5MHz.If you have a disk drive then you could add an LS74 to the above circuit.
The LM380 gets quite warm but does not need a heat sink.If you prefer you could run the signal through your own amplifier from channels A,B & C by connecting them together and to 0v via 1K2 and through 100Mfd to the amp. input.

DETAIL.
The PSG has 15 registers R0 to R15.
R0 to R5 provide tones,R6 noise,R7 is the enable register,R8,9 & 10 control each channels volume,R11,12 & 13 control the envelope shape and R15 & 16 are the input output ports,A & B.
The addresses are:

F160	latch address
F162	Read data from PSG
F164	Write data to PSG
F166	Inactive

Register 7 is laid out as follows:
Bits 0,1 & 2 enable tones from channels A B & C when low.
Bits 3,4 & 5 enable noise from channels A B & C when low.
Bits 6 & 7 enable input from I/O A & B when low and output when high.
I have run I/O channels to 9 way D type connectors and use them for games controllers.The pins all have internal pull ups and read FFFF when read.
All you need to do is ground any output pin and read.
I have also successfully run my Epson MX80 in parallel through these ports.
I used the printer spooler from Newsletter 2 with my coding added in place of the CRU coding.

PROGRAMMING:

```
To read I/O:      10 MEM(0F160H)=0EH (port A)
                  20 A=MEM(0F162H)
                  30 Print A (or whatever you want to do with it)

In M/C            LI R1,>E00
                  MoVB R1,@>F160
                  MOVB @>F162,R2
                  MOVB R2,@>Save location

To write I/O      10 MEM(0F160H)=0FH (port B)
                  20 MEM(0F164H)=DATA

In M/C            LI R1,@>F00
                  LI R2,@>DATA
                  MOVB R1,@>F160
                  MOVB R2,@>F164
```

SORT DIRECTORY PROGRAMME   BY   C.J.YOUNG

Sorts the disk directory entries into alphabetical order

```
 10  REM
 20  REM ***************
 30  REM *             *
 40  REM *   SORTDIR   *
 50  REM *             *
 60  REM * Version 1.0 *
 70  REM *             *
 80  REM ***************
 90  REM
 100  DATA 0420H,06180H,0D000H,01601H
 110  DATA 0380H,0460H,06550H,0202H
 120  DATA 040H,0D0D0H,0DC11H,0DC43H
 130  DATA 0602H,016FBH,0380H,0C100H
 140  DATA 0C141H,05C4H,05C5H,0706H
 150  DATA 09D74H,015F8H,016F0H,0926H
 160  DATA 016FBH,010F4H
 170  DATA 0
 180  REM
 190  REM * Set up variables *
 200  REM
 210  TEXT
 220  ? " Sort Dir Program 1.0 1987"
 230  ? " Input Drive ?";
 240  IK=KEY[0]
 250  IF IK=0: GOTO 240
 260  IF IK<48: GOTO 240
 270  IF IK>51: GOTO 240
 280  DRV=IK-48
 290  ? DRV
 300  D2=DRV*2
 310  D=DRV*256
 320  NF=0  !Number of files
 330  REM
 340  REM * Get drive parameters *
 350  REM
 360  DP=MWD[06382H+D2]
 370  SPT=MWD[DP]  ! Sectors/track
 380  NS=MWD[DP+2]  !No of Sectors
 390  DS=MWD[DP+4]  !Directory start
 400  MF=MWD[DP+6]  !Max Files
 410  BPS=MWD[06362H+D2]  !Bytes/Sector
 420  DDA=DS*BPS  !  Disk Dir Addr
 430  DDL=MF*64  !  Disk Dir Length
 440  REM
 450  REM * Set up Arrays *
 460  REM
 470  DIM MC[99]
 480  DIM B[DDL/6+1]
 490  DIM $NM[1]
 500  DIM $SY[1,1]
 510  $SY[0,0]="SYSTEM$"
```

```
520  $SY[1,0]="AUTOEXEC"
530  ST=0  !Start Of O/P
540  AMC=ADR[MC[0]]
550  SWP=AMC+14
560  CHK=AMC+30
570  AB=ADR[B[0]]
580  FOR I=0 TO 777 STEP 2
590   READ Q
600   IF Q: MWD[AMC+I]=Q
610     ELSE I=999
620  NEXT I
630  REM
640  REM * Read directory *
650  REM
660  CALL AMC,0,D,DDA,AB,DDL
670  REM
680  REM * Sort Directory *
690  REM
700  FOR X=0 TO MF-1
710   IF MWD[AB+X*64]=0: GOTO 750
720   IF X=NF: GOTO 740
730   CALL SWP,AB+NF*64,AB+X*64
740   NF=NF+1
750  NEXT X
760  ? "Number of Files ="NF
770  IF NF<2: STOP
780  REM
790  REM * Check For System Files *
800  REM
810  FOR Q=0 TO 1
820   FOR Z=0 TO NF-1
830    FOR I=0 TO 7
840     $NM[0;I+1]=%MEM[AB+Z*64+I+2]%0
850    NEXT I
860    IF $NM[0]<>$SY[Q,0]: GOTO 900
870    IF Z=ST: GOTO 890
880    CALL SWP,AB+ST*64,AB+Z*64
890    ST=ST+1
900   NEXT Z
910  NEXT Q
920  IF NF-ST<2: STOP
930  REM
940  REM * Sort Rest Of Files *
950  REM
960  FOR Z=ST TO NF-2
970   FOR X=Z+1 TO NF-1
980    CALL CHK,AB+X*64,AB+Z*64
990   NEXT X
1000 NEXT Z
1010 REM
1020 REM * Write Directory *
1030 REM
1040 CALL AMC,0FFH,D,DDA,AB,DDL
1050 ? "Done"
1060 STOP
```

DOUBLE DENSITY DISK INSPECT.

BRIAN HARRIS. PLYMOUTH, DEVON.

In NEWSLETTER 10, it was mentioned that the DISK INSPECT UTILITY does not work on double density disks. What in fact happens is that only half a sector is displayed. I.E. Only 128 bytes instead of 256. Some time ago I modified the D.I. utility, (CDOS disk inspect utility 1.0 1984) to diplay the full 256 double density bytes. The following is a listing of the amended program. New lines have !** after them, altered lines !*. Do'nt forget the space corrections in lines 270,290 and 540.

```
LIST
 100  TEXT : ? @(0,17);"CDOS  double  density  disk  inspect "  !*
 110  ? @(0,23);"[Ascii,Decrement,Hex,Increment,Modify]";
 120  DIM X[4],B[50]: $M="H"
 130  AX=ADR[X[0]]: AB=ADR[B[0]]
 140  MWD[AX]=0420H: MWD[AX+2]=06260H
 145  MWD[AX+4]=0D8C6H: MWD[AX+6]=02H
 150  MWD[AX+8]=0380H
 160  ? @(0,19);" Drive    ": ? " Track      ": ? " Sector     "
 165  ? @(10,19);: INPUT %1;D
 167  IF D>3 THEN GOTO 155
 170  ? @(8,20);: INPUT %3;T
 180  IF T<0 OR T>159 THEN GOTO 170
 190  ? @(9,21);: INPUT %2;S
 200  IF S<0 OR S>15 THEN GOTO 190
 210  E=0
 220  CALL AX,D,T,S,ADR[E],AB,0,0
 230  IF E<>0 THEN ? @(16,19);"READ ERROR";£E/256 LAND 03FH: GOTO 350
 240  ? @(16,19);"                        "
 250  BB=AB: ? @"H";
 260  FOR R=0 TO 15
 270   ? £;R*16;"     ";  !*
 280   FOR C=0 TO 15  !*
 290    IF $M="H" THEN ? £;MEM[BB];   !*
 300    IF $M="A" THEN GOSUB 520
 310    BB=BB+1
 320   NEXT C
 330   ?
 340  NEXT R
 350  ? @(20,20);: INPUT "Command"£1,$K;
 360  IF $K="I" THEN S=S+1: GOTO 430
 370  IF $K="D" THEN S=S-1: GOTO 430
 380  IF $K="" THEN GOTO 160
 390  IF $K="A" THEN $M=$K: GOTO 250
 400  IF $K="H" THEN $M=$K: GOTO 250
 410  IF $K="M" THEN GOTO 720
 420  GOTO 160
 430  IF S<0 THEN T=T-1: S=15
 440  IF S>15 THEN T=T+1: S=0
 450  IF T<0 THEN T=0
 460  IF T>159 THEN T=159
 470  ? @(8,20)£"000"T: ? @(9,21)£"00"S
 480  GOTO 210
 490  CALL AX,D,T,S,ADR[E],AB,0,0FFH
```

```
500  IF E<>0 THEN ? @(20,19);"WRITE ERROR";£E/256 LAND 03FH
510  GOTO 350
520  IF MEM[BB]<020H THEN $Q="."
530    ELSE $Q=%MEM[BB]%0
540  ? $Q;" ";   !*
550  RETURN
560  BB=AB: R=0: C=6   !*
570  IF MEM[BB]>01FH THEN $SS=%MEM[BB]%0
580    ELSE $SS="."
590  ? @(C,R);$SS;: ? @"L";
600  K=KEY[0]: IF K=0 THEN WAIT 1: GOTO 600
610  IF K=08H THEN C=C-2: BB=BB-1   !*
620  IF K=09H THEN C=C+2: BB=BB+1   !*
630  IF K=0AH THEN R=R+1: BB=BB+16   !*
640  IF K=0BH THEN R=R-1: BB=BB-16   !*
650  IF K=0DH THEN GOTO 490
660  IF K>01FH THEN MEM[BB]=K: GOTO 570
670  IF C<6 AND R=0 THEN C=6: BB=BB+1   !*
675  IF C<6 THEN C=36: R=R-1   !**
680  IF C>36 AND R=15 THEN C=36: BB=BB-1   !*
685  IF C>36 THEN C=6: R=R+1   !**
690  IF R<0 THEN R=0: BB=BB+16   !*
700  IF R>15 THEN R=15: BB=BB-16   !*
710  GOTO 570
720  IF $M="A" THEN GOTO 560
730  BB=AB: R=0: C=6   !*
740  ? @(C,R);£;MEM[BB];: ? @"2L";
750  K=KEY[0]: IF K=0 THEN WAIT 1: GOTO 750
760  IF K=08H THEN C=C-2: BB=BB-1   !*
770  IF K=09H THEN C=C+2: BB=BB+1   !*
780  IF K=0AH THEN R=R+1: BB=BB+16   !*
790  IF K=0BH THEN R=R-1: BB=BB-16   !*
800  IF K=0DH THEN GOTO 490
810  IF K>02FH THEN IF K<03AH THEN GOSUB 880
820  IF K>040H THEN IF K<047H THEN K=K-7: GOSUB 880
825  IF C<6 AND R=0 THEN C=6: BB=BB+1   !**
830  IF C<6 THEN C=36: R=R-1   !*
835  IF C>36 AND R=15 THEN C=36: BB=BB-1   !**
840  IF C>36 THEN C=6: R=R+1   !*
850  IF R<0 THEN R=0: BB=BB+16   !*
860  IF R>15 THEN R=15: BB=BB-16   !*
870  GOTO 740
880  K=MOD[K,16]
890  MEM[BB]=MOD[MEM[BB],16]*16+K
900  RETURN
```

P.C.B. Plot

O.W.Hulme

The PCB-PLOT programe was devised as an easy way to overcome the difficulty of drawing the tracks of a PCB. Erase and redraw a few lines on paper and it soon becomes unreadable, on the other hand a VDU leaves no trace of an alteration.

Before loading type in ´NEW 78EAH´ to reserve enough space for the transfer of screen into main memory.

From main memory it can be saved using MON.D 60EA 78EA ,but remember, it only records what was on the screen the last time you pressed the D key, which may not be what you are looking at the time of saving.

The L key loads the screen from main memory thus enabling work to be continued where you left off

If while printing pads you use delete to reposition them, reset the ink by using the home key. This puts ink to the pads but not the lines, enabling you to move from i.c pad to i.c.pad without leaving unwanted lines.

Code is included to call the paint routine but please check lines 870 and 890 to ensure that baud rate and unit number are compatable with your printer. The listing for the paint routine can be found in the GROUP NEWSLETTER No4, page 7.

PCB—PLOT

```
10   TEXT
20   ; "<C>"
30   ; "DID YOU REMEMBER TO SET ´NEW 783AH´?"
40   ; : ; " CONTROL KEYS"
50   ; : ; "  8.........8 PIN I/C PAD"
60   ; " 14........14 PIN I/C PAD"
70   ; " 16........16 PIN I/C PAD"
80   ; " 18........18 PIN I/C PAD"
90   ; " 20........20 PIN I/C PAD"
100   ; " 22........22 PIN I/C PAD"
110   ; " 24........24 PIN I/C PAD"
120   ; " 28........28 PIN I/C PAD"
130   ; "  4........40 PIN I/C PAD"
140   ; "  P........SINGLE PAD"
150   ; " ARROWS....CURSOR MOVEMENTS"
160   ; " HOME......MOVEMENTS ARE NEUTRAL"
170   ; " INSERT....MOVEMENTS ARE PLOT"
180   ; " DELETE....MOVEMENTS RAE UNPLOT"
190   ; " D.........LOAD VDU TO MAIN MEMORY"
200   ; " L.........LOAD MAIN MEMORY TO VDU"
210   ; " C.........ACTIVATE PAINT ROUTINE"
220   ; : ; " PRESS ANY KEY TO CONTINUE"
230   K=KEY[0]: IF K=0: GOTO 230
240   DATA 513,6144,1218,-10238,-3807,-10238,-3807,-14629
250   DATA -9184,-3808,1537,5884,896,4096,513,6144
260   DATA 514,64,515,-3808,-10238,-3807,1730,-10238
270   DATA -3807,-11024,1537,5885,896
280   FOR I=06000H TO 06038H STEP 2
290    READ A: MWD[I]=A: NEXT I
300   SHAPE 1,-3904,-24432,2052,0
310   A=0: B=0: C=2: F=0
320   SPRITE 0,A,B,1,15
```

```
330  K=KEY[0]: IF K=0: GOTO 330
340  IF K=09H: A=A+1
350  IF K=08H: A=A-1
360  IF K=0BH: B=B-1
370  IF K=0AH: B=B+1
380  IF K=017H: C=0
390  IF K=016H: C=1
400  IF K=01EH: C=2
410  IF K=031H: X=10: GOTO 550
420  IF K=032H: X=20: GOTO 550
430  IF K=038H: Y=17: X=15: GOTO 680
440  IF K=050H: E=A: B=B: GOSUB 760
450  IF K=044H: CALL 06000H,0603AH
460  IF K=04CH: CALL 0601CH,0603AH
470  IF K=043H: GOSUB 840
480  X=0
490  IF K=034H: Y=31: X=95: GOTO 680
500  IF C=0: UNPLOT A,B
510  IF C=1: PLOT A,B
520  SPRITE 0,A,B
530  GOTO 330
540  STOP
550  L=KEY[0]: IF L=0: GOTO 550
560  IF L=036H: X=(X+6)/2*5-5: Y=17: GOTO 680
570  IF L=034H: GOTO 610
580  IF L=038H: GOTO 630
590  IF L=030H: GOTO 650
600  IF L=032H: GOTO 660
610  Y=17: IF X=20: Y=31
620  X=(X+4)/2*5-5: GOTO 680
630  Y=17: IF X=20: Y=31
640  X=(X+8)/2*5-5: GOTO 680
650  Y=17: IF X=20: X=X/2*5-5: GOTO 680
660  Y=22: IF X=20: X=(X+2)/2*5-5: GOTO 680
670  GOTO 330
680  FOR I=0 TO X STEP 5
690   E=A+I
700   GOSUB 730
710  NEXT I
720  GOTO 330
730  F=B+Y: IF C=0: GOTO 790
740  PLOT E,F TO E+1,F TO E+1,F+1 TO E,F+1 TO E,F+2
750  PLOT E,F+2 TO E+1,F+2 TO E+1,F+3 TO E,F+3
760  IF C=0: GOTO 810
770  PLOT E,B TO E+1,B TO E+1,B+1 TO E,B+1 TO E,B+2 TO E+1,B+2
780  PLOT E+1,B+2 TO E+1,B+3 TO E,B+3: GOTO 830
790  UNPLOT E,F TO E+1,F TO E+1,F+1 TO E,F+1 TO E,F+2
800  UNPLOT E,F+2 TO E+1,F+2 TO E+1,F+3 TO E,F+3
810  UNPLOT E,B TO E+1,B TO E+1,B+1 TO E,B+1 TO E,B+2 TO E+1,B+2
820  UNPLOT E+1,B+2 TO E+1,B+3 TO E,B+3
830  RETURN
840  REM
850  MEM[0A4H+7]=15: MEM[0A4H+4]=1
860  SWAP
870  BAUD 2,1200: UNIT 2
880  CALL 05E00H
890  UNIT -2: RETURN
```

INTO ONTO AND OUT OF E.Bus PART 3 Tim Gray

This the third article in the series follows a request from some members for more details of how to add the hardware necessery to get the E.Bus up and running.

Firstly we now have available two P.C.B.s for the LS2001 replacement circuit shown in part one. The reason for two P.C.B.s is because on the Cortex Mk 2 the main P.C.B. is mounted the opposite way round in the case to the Mk 1. This means that a header plug version of the LS2001 replacement circuit can't be fitted as it would foul the keyboard.

The header plug version P.C.B then is is for Mk 1 Cortex and is fitted by plugging into the socket for the LS2001. The conventional P.C.B. is for Mk 2 Cortex and is designed to fit in between IC11 the TMS9995 and IC8 the TMS9929 mounted on stick down P.C.B. stand off pillars. If the TMS9911 is fitted the P.C.B. straddles over it, if the new WD2797 floppy controller is used the TMS9911 is not required so the space is vacant. The P.C.B. is then wired back to the LS2001 position where a socket is not required.

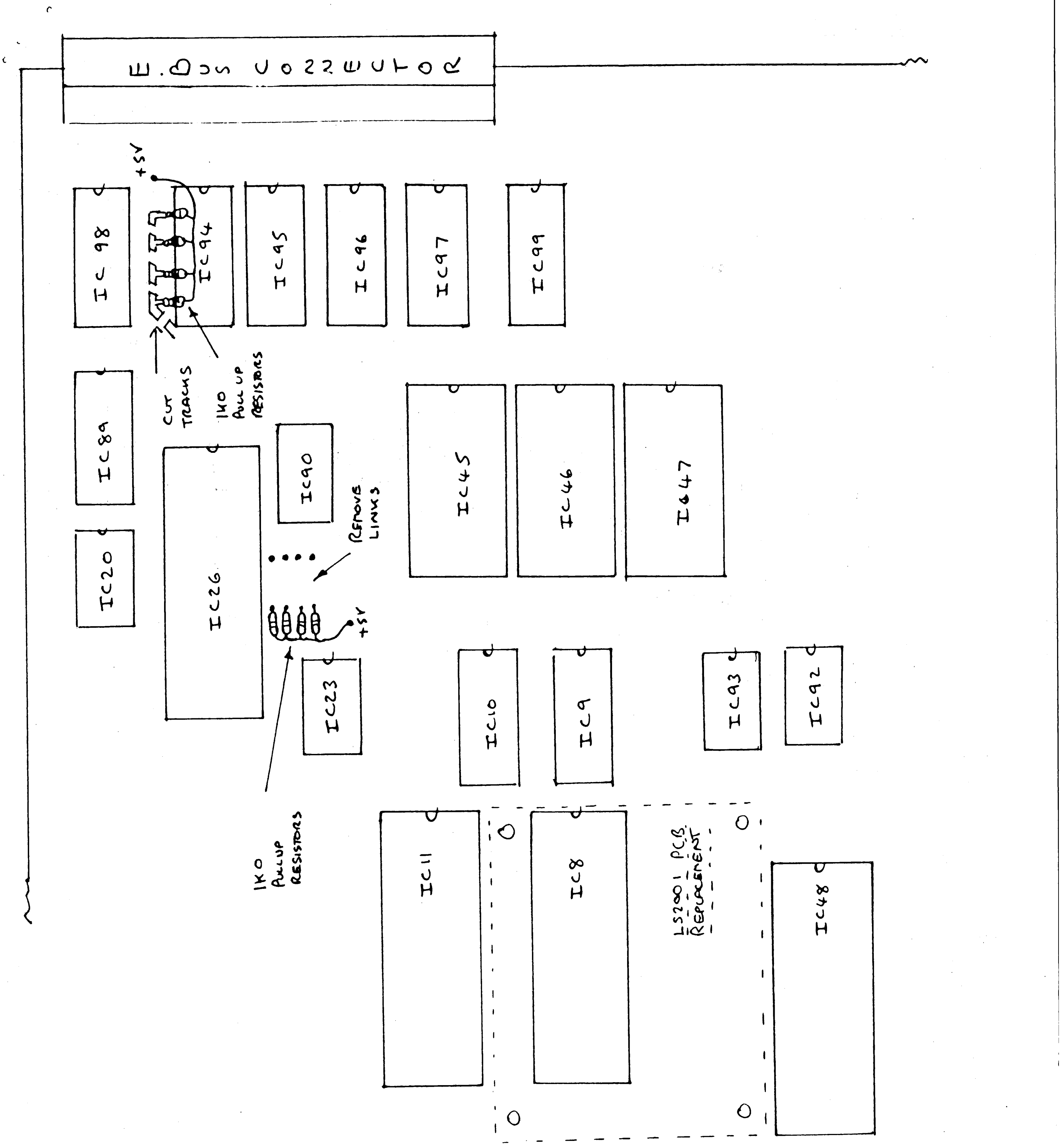
Fitting the LS2001 replacement

If you have done the mods on the main board as detailed in the Centronics interface kit start by removing them. Make up the LS2001 replacement P.C.B. as detailed in the drawings. If the header plug version is used fit wire wrap pins or socket and plug into IC89 position. If using the conventional P.C.B. connect fine wires to the terminals fit the P.C.B. in position using stand off pillars and wire back to IC89 position.

The conventional P.C.B. also has a LS04 fitted. This is to add two gates propogation delay between IC93 outputs and IC95,96 & 97 inputs. Cut the track from IC93 pin 3 and pin 6. Connect IC93 pin 3 to P.C.B. -ABEIN. Connect P.C.B. -ABEOUT to IC95,96 pins 1 and 19. Connect IC93 pin 3 to P.C.B. -DBEIN. Connect P.C.B. -DBEOUT to IC97 pin 19. This LS04 is not included on the header plug version but the same can be achieved by making a 14 pin header plug with IC93 and a LS04 saddle backed as per the drawing. This combined gate is then plugged into IC93 socket.
All the other E.Bus interface IC.s should now be fitted.

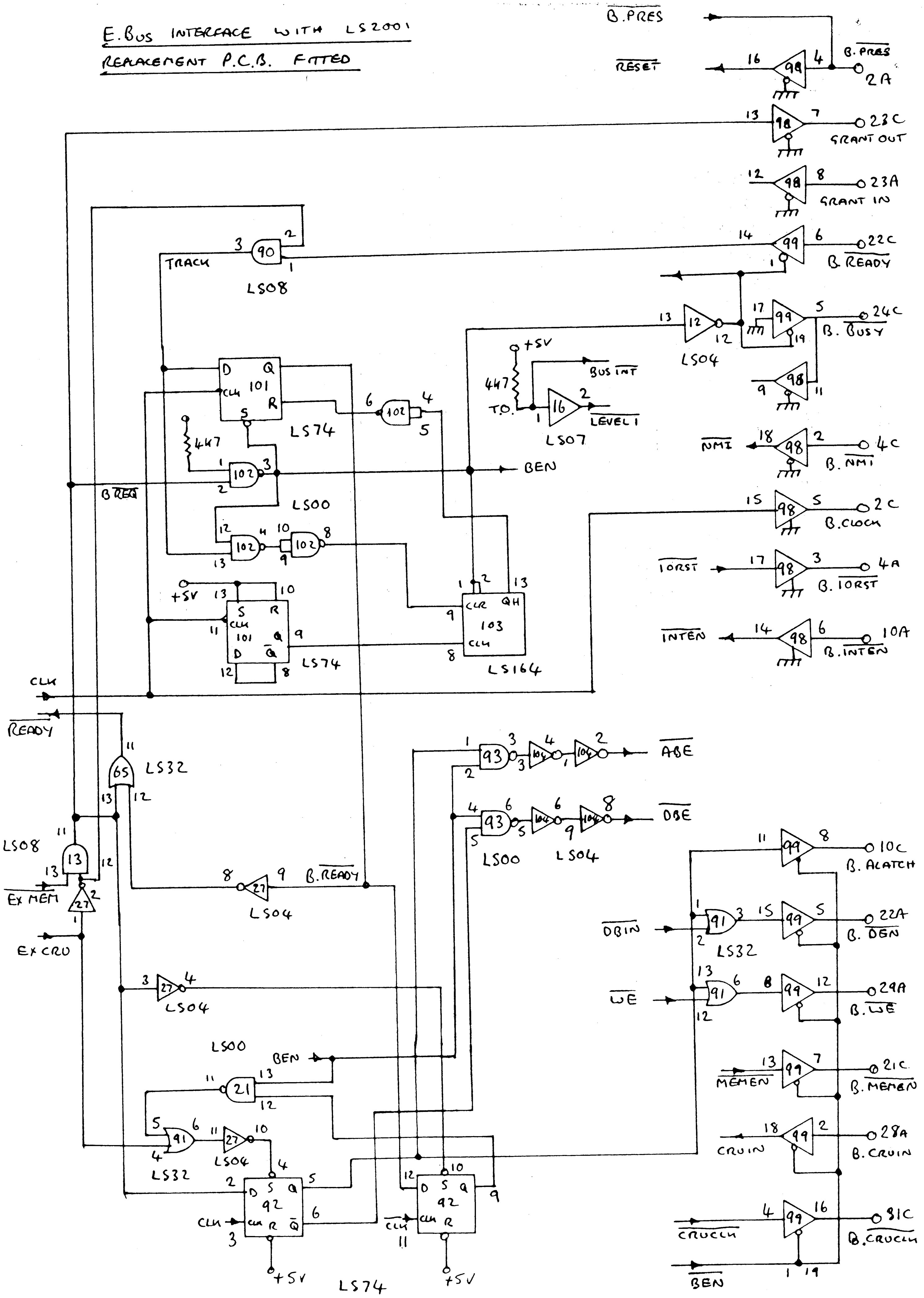
Fitting the memory mapper.

Any of the following IC.s can be used in this position:-
LS610, LS611, LS612, LS613 but LS611 and LS613 require 1K0 pull up resistors from +5V to pins 18,19,22,23,24,25,26 & 27. LS610 and LS611 require a pull up resistor on pin 28. These pull up resistors can be conveniantly mounted allong side IC26 where the wire links were and allong side IC94 where the track has to be cut. The 4 links allong side IC26 must of course be removed and the tracks that connect IC94 inputs to ground must be cut to allow the mapper to function correctly.


E.BUS CONNECTOR
+5V
IC98
IC94
IC95
IC96
IC97
IC99
CUT TRACKS
1K0 PULL UP RESISTORS
IC89
IC26
IC90
IC20
REMOVE LINKS
IC45
IC46
IC47
+5V
IC23
IC10
IC9
IC93
IC92
1K0 PULL UP RESISTORS
IC11
IC8
LS2001 PCB REPLACEMENT
IC48

E.Bus Interface with LS2001

Replacement P.C.B. Fitted



Testing

The main board can now be re-fitted to the Cortex and normal operation checked. After making sure the computer works normally then type *FRED. The Cortex should respond with the error message "expansion eprom not found" rather than "required hardware not found" as would be the case if the mapper was not fitted.

Backplane

The E.Bus backplane should be constructed and wired back to the E.Bus socket on the Cortex with a short length of ribbon cable. Connect every other wire in the ribbon as an earth lead between signal wires. It is advisable to use an extra power supply for the Backplane and if so do not connect the power lines down the ribbon cable.

Using the Bus...CRU

CRU input / output is quite easy as any access to CRU locations outside of the internal range automatically causes an E.Bus CRU access to occure. Connection of other TMS9902 serial ports however involves using interupts and will be dealt with in a future article.

Memory

Any access to external memory requires use of the memory mapper This device consists of 16 registers one for each 4K block of the 64K CPU memory map. The registers are located on word addresses from F100 to F11E. In the Cortex only the lower 8 bits of the device are used but as the address decoding is not complete each mapper word location appears to have both high and low byte set to the same value. The 8 bit value in each register forms the top 8 bits of a 20 bit address range. The mapper is normally set up for the conventional address range as shown:-

internal address,	mapper location,	mapper value,	extended addr
0000 - 0FFE	F100	0000	00000 - 00FFE
1000 - 1FFE	F102	0101	01000 - 01FFE
2000 - 2FFE	F104	0202	02000 - 02FFE
3000 - 3FFE	F106	0303	03000 - 03FFE
4000 - 4FFE	F108	0404	04000 - 04FFE
5000 - 5FFE	F10A	0505	05000 - 05FFE
6000 - 6FFE	F10C	0606	06000 - 06FFE
7000 - 7FFE	F10E	0707	07000 - 07FFE
8000 - 8FFE	F110	0808	08000 - 08FFE
9000 - 9FFE	F112	0909	09000 - 09FFE
A000 - AFFE	F114	0A0A	0A000 - 0AFFE
B000 - BFFE	F116	0B0B	0B000 - 0BFFE
C000 - CFFE	F118	0C0C	0C000 - 0CFFE
E000 - EFFE	F11A	0E0E	0E000 - 0EFFE
F000 - FFFE	F11E	0F0F	0F000 - 0FFFE

To access external memory one of the mapper locations must be programmed with a value greater than >0F. Lets assume we want to access extended memory starting at >14000. We can switch a 4K block of it into the normal 64K memory range starting at >2000 by programming the mapper register at >F104 to >14 instead of >02. This in itself is not enough we also need to switch the mapper on. The code for doing all this is as follows:-

```
     MOVB @>F104,R0        ;SAVE MAPPER CONTENTS
     LI   R1,>1400
     MOVB R1,@>F104        ;LOAD MAPPER WITH NEW VALUE
     CKON                  ;SWITCH MAPPER ON
     MOV  @>201A,R2        ;FETCH DATA WORD FROM EXT ADDR >1401A
     CKOF                  ;SWITCH MAPPER OFF
     MOVB R0,@>F104        ;RESTORE ORIGINAL MAPPER VALUE
```

Unfortunately it would be quite difficult to do this from Basic firstly because there is no command to switch the mapper on or off and secondly because it is difficult to find a 4K block of memory to switch out that Basic does not use in some way. The best way to access a large area of expansion memory from Basic is to use RAMDISC to configure a third drive as RAM. This allows use of the disk commands OPEN, CLOSE, PUT and GET to be used to store or recall strings or variables to or from expansion memory. As an alternative if you don't have disk drives or enough expansion memory to configure as RAMDISC use can be made of a routine in the Cortex ROM for tranfering data from external memory. This routine starts at 5456H and is not used by any of the Cortex system. There is a small bug in the routine but that can be easily fixed. The routine is only designed to transfer from external memory to internal memory but it can be modified to perform transfers in the opposite direction.

Here is an example of how to make use of the routine :-

```
 10  REM *** EXPANSION MEMORY ACCESS ***
 20  DIM VA[400],VB[400]
 30  COD=0
 40  AC=ADR[COD]
 50  REM *** SET UP CODE TO CALL ROUTINE AT 5456H ***
 70  MWD[AC]=0420H: MWD[AC+2]=05456H: MWD[AC+4]=0380H
 80  FOR A=1 TO 400
 90   VA[A]=A
100  NEXT A
110  REM *** STORE VA[] TO EXP MEM STARTING AT 14010H ***
130  MWD[05482H]=0C8B4H: MWD[05494H]=059CH  !** SET TO WRITE
140  CALL AC,0,014H,010H,2400,ADR[VA[0]]
150  REM *** READ BACK TO VB[0] **
160  MWD[05482H]=0CD22H: MWD[05494H]=059CH  !** SET TO READ
170  CALL AC,0,014H,010H,2400,ADR[VB[0]]
180  REM *** CHECK DATA ***
190  FOR A=1 TO 400
200   PRINT £"9999"VB[A];" ";
210  NEXT A
220  STOP
```

The parameters for the Calls at 140 and 170 are as follows :-

CALL AC,<zero>,<external page>,<external start addr>,<number of bytes to transfer>,<internal address for start of transfer>

The first parameter is not used but is required to put the other parameters in the correct registers of the call routine.

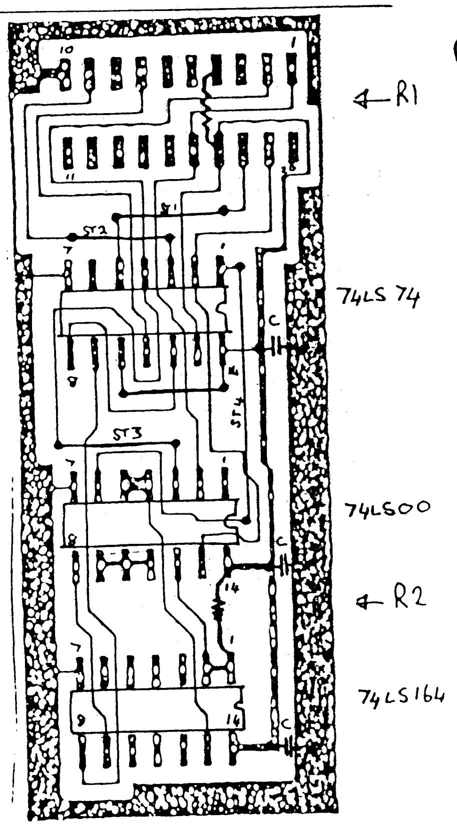
The external page number is the top byte of the extended address.

The external start address is the remaining three nibbles of the extended address.

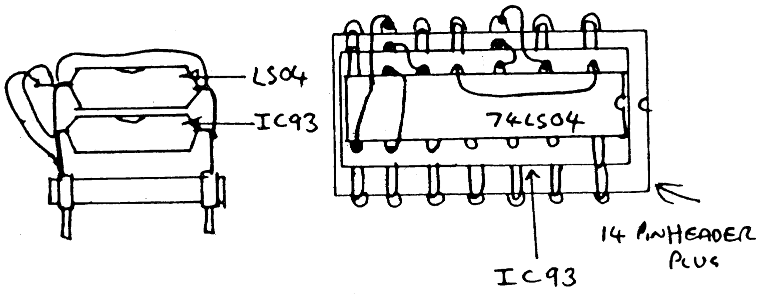
The transfer will start at the address given for external and internal memory and will be incremented up to the number of bytes to transfer. There is no limit to the size of eache transfer as the routine automatically increments the external page as it gets to the end of a 4K boundary.

It is of course necessary to keep track of memory usage but the routine can be used effectively to expand variable storage space considerably.

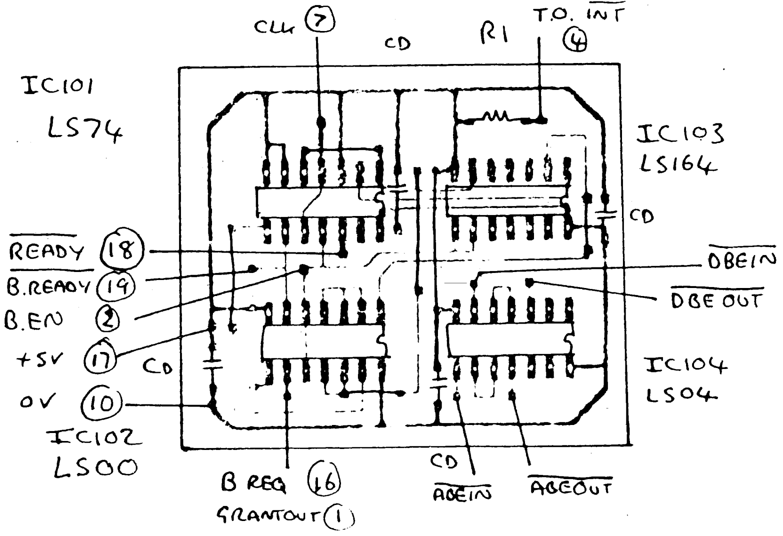
74LS2001 REPLACEMENT HEADER PLUG COMPONENT SIDE



ST1—4 STRAPS
C = DECOUPLING ·1μF
R1 & R2 are 47K MOUNTED ON COPPER SID



IC93 + LS04 HEADER PLUG

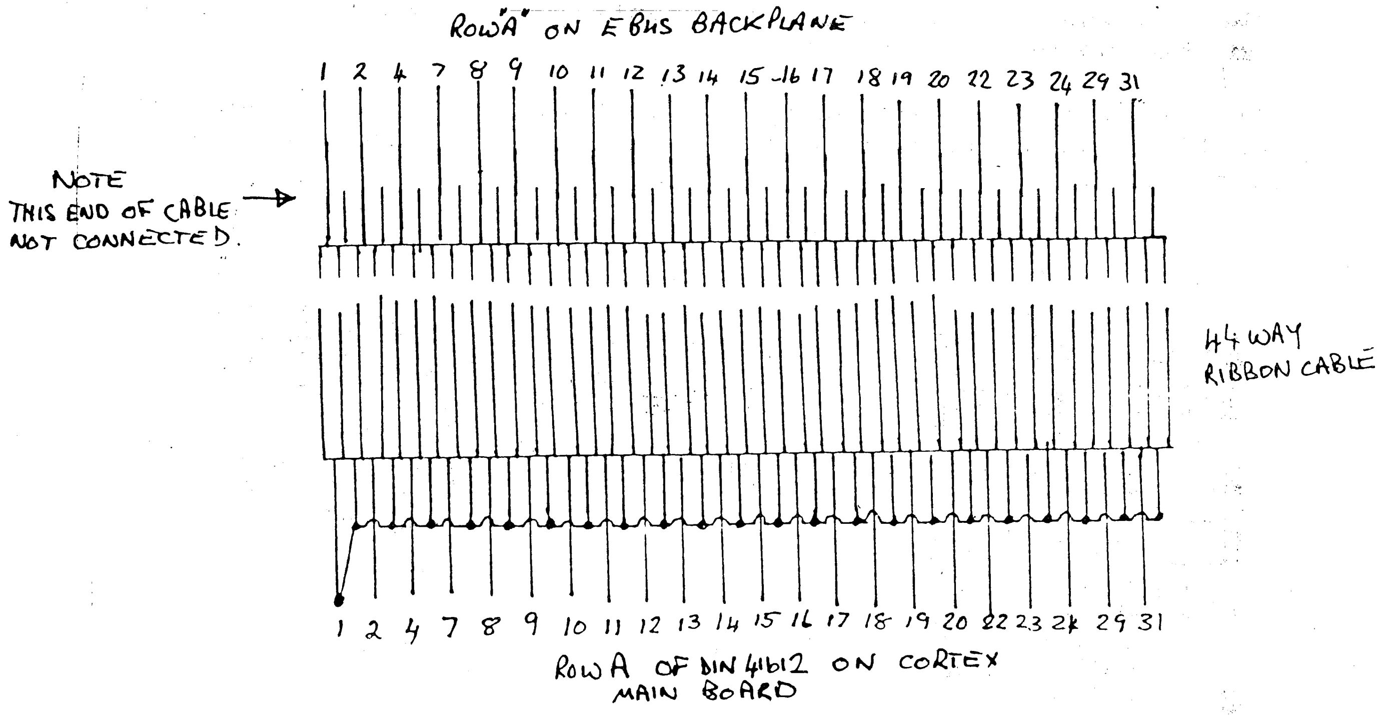


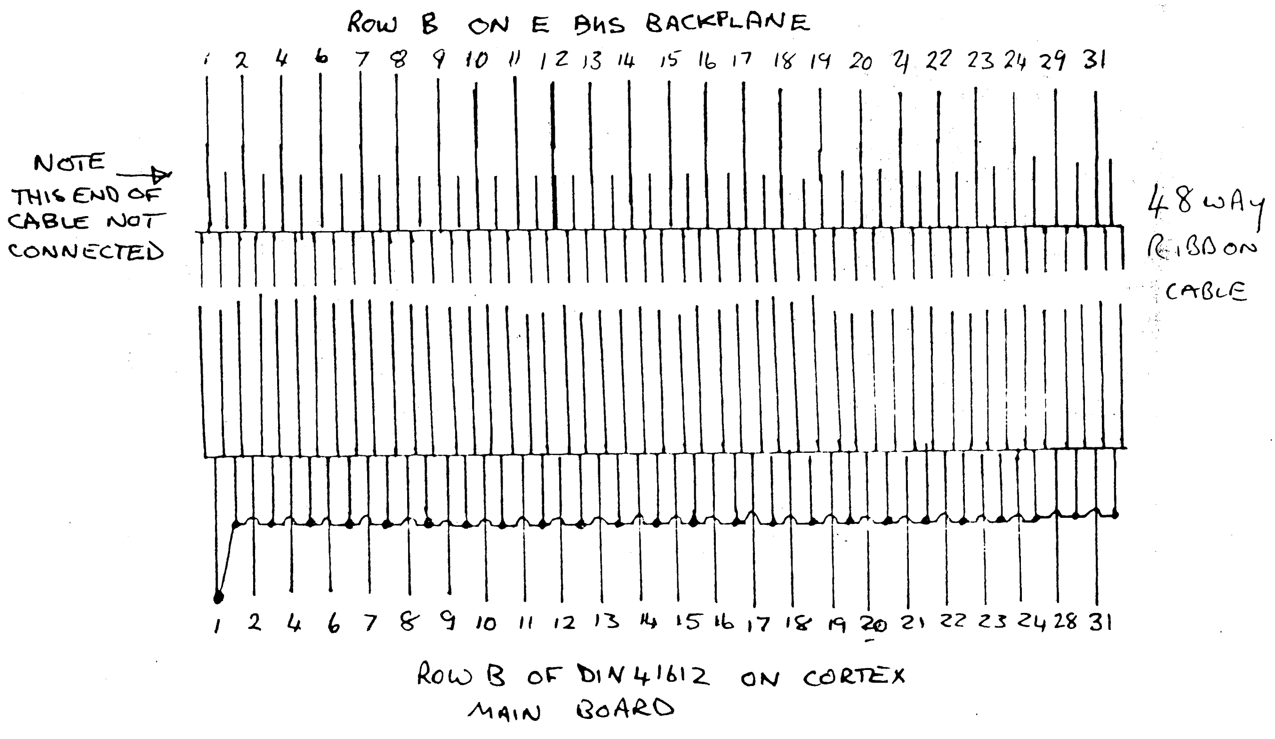
LS2001 REPLACEMENT P.C.B.

EXAMPLE OF E BUS CABLE

MAKE THE EBUS EXTENSION CABLE FROM TWO 50 WAY RIBBON CABLES

NOTE - ALTERNATE LINES ARE EARTH LINES AND ARE ONLY CONNECTED AT ONE END (CORTEX MAIN BOARD END)


ROW "A" ON EBUS BACKPLANE
1 2 4 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 23 24 29 31
NOTE
THIS END OF CABLE NOT CONNECTED.
44 WAY
RIBBON CABLE
1 2 4 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 23 24 29 31
ROW A OF DIN 41612 ON CORTEX
MAIN BOARD


ROW B ON E BUS BACKPLANE
1 2 4 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 29 31
NOTE
THIS END OF CABLE NOT CONNECTED
48 WAY
RIBBON
CABLE
1 2 4 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 28 31
ROW B OF DIN 41612 ON CORTEX
MAIN BOARD