

CORTEX USER GROUP

NEWSLETTER III

SEPTEMBER 1985

Welcome to the first of the new quarterly newsletters. We apologise for the delay in publication of this issue.

Once again the number of letters we have received has increased. Thank you for your continued interest and support. We are sure that almost any subject (relating to the Cortex) which you wish to write about will be of interest to other users, so please keep those letters coming.

You may notice a change in the format of the newsletter, as all of the articles are now arranged in relevant sections. We hope that this will make the newsletter easier to follow. It would also be helpful if people writing in were to keep different items roughly separated (e.g. keep programs separate from tips).

Further to the notice in the Newsletter II, referring to the discontinued Texas 9909 (Floppy disc control chip), we now bring you more news on this subject.

The chip is now replaceable by a module which uses a different chip.

Due to the prohibitive cost of producing this module, we have decided it was better to simply make the circuit diagram available (see enclosed sheet). The module takes its inputs from the 9909 socket, and outputs directly to the floppy disc control. The circuit has, of course, been built, and was found to work well.

The following letters, and programs have been submitted by Cortex Users. Powertran Cybernetics Limited cannot accept liability for their content.

CORTEX TIPS

This is your chance to pass on any amazing (or just plain useful) discoveries, that you have made about the workings of the CORTEX.

Mr S Pratt
Cleveland

Stephen has kindly sent in a couple of useful notes.

Some more information for the magazine, it's not much but it might be of some help.

1. The protected memory, mentioned by Robert Lee is fine as long as you do not attempt to define characters greater than 215 using "CHAR" since this is where they are stored.
2. The character set is stored in memory from 5BAEH to 60ECH inclusive, lower case is from 5D34H to 60DDH inclusive.

Hope this is useful.

- - - x x x - - -

Julian Terry
Rainham
Kent

Julian has been trying to confuse his Cortex with negative line numbers, and has also sent some information about XOP's.

In the first issue of the Users Group Magazine someone mentioned that one could renumber backwards, well you can also renumber over 32767. This is not adviseable since one ends up with negative line numbers and the program will not run.

```
XOP0      moves  *R11 → R0
           *R11+2 → R1
           *R11+4 → R2
```

```
XOP1      moves  R0 → *R11
           R1 → *R11+2
           R2 → *R11+4
```

XOP8 Sets registers 0 to 2 (inc) to zero.

XOP9 If R0 <> 0 then R0 = R0 + 8000H

XOP11 This is a very useful routine. R8 points to the
start of an arithmetic expression in internal
condensed form (not as in a listing).
When the command is executed the integer value
of the expression is returned in the argument of
XOP11. This will perform all error checking and
includes all system functions. R8 after execution
will point to the byte after the expression.

XOP 14 is used with the following format
XOP @>n, 14
Where n is an error number. The appropriate error
message will be displayed and correct action taken
(e.g. call to ERROR line).
n is in the range 0 - 63
At present any error >49 will return "SYSTEM ERROR".
There are two more error messages not mentioned in the
Cortex manual. These are 41-Expansion Eprom not
found, 0-FATAL SYSTEM ERROR (the computer locks up when
this is executed.) Personalized errors may be added
by putting "BL@> routine address" at location 1DFCH.
R1 will hold the error number.
R2 should be set to 5732H and RT executed if it is
not your error. If it is your error then the
routine should finish with B@> 1E14.
SYS[1] will be set as required.
Errors with numbers greater than 31H will be trapped.

XOP15 is spare.

- - - x x x - - -

Chris Young
Lossiemouth
Morayshire

Chris makes another appearance with his list of MID
instructions which are not otherwise documented.

0001 = WRIT R0
0002 = Prints CRLF
0003 = Reprints last message
0004 = WRIT R1
0005 = WRIT R1
0006 xxxx = MSG @>xxxx
0007 xxxx = MSG @>xxxx
0008 =
0009 = WRIT > (write immediate)
000A =

```

000B      = Retrieve Character from keyboard buffer.
xxxx      if null then execute xxxx
yyyy      if not null execute yyyy
zzzz      if ESC then execute zzzz
           xxxx,yyyy & zzzz should be JMP's
000C      = READ R0 (used to get characters from Cassette when
           keyboard is disabled)
000D      = READ R0
xxxx      if nul execute xxxx
yyyy      if not null execute yyyy
000E      = Wait until >EDAA is zero
000F      = TEXT (as in BASIC)
0010      = GRAPH (as in BASIC)
0011      = TEXT (if not in TEXT mode)
0012      = GRAPH (if not in GRAPH mode)
0013      = Clears keyboard buffer
0014      = cassette !
0015      = cassette !

```

- - - x x x - - -

Tim Gray
Wolverhampton
West Midlands

Here are some more tips from Tim.

CDOS Modifications

The program FILECOPY has no facility for choosing the files to be copied from one disc to another. If the following lines are added to the program it allows you to go through the directory, but only copy the selected files. The program prints the filenames found, and then asks if you want to copy it or not, answer Y or N, if yes the file is copied, if no the next directory entry is found.

```

361 PRINT
362 INPUT " do you want to copy it Y/N "$ANS
363 IF $ANS = "Y" THEN GOTO 370
364 IF $ANS = "N" THEN GOTO 790
365 GOTO 362

```

After adding the extra lines save the program with the new name COPYFILE so that you still have the choice of which program you use.

To allow the use of the * command to load program from drive 0 assemble the following code at 69A4.

```

69A4 MOV R8,R2
69A6 JMP >69AC
69A8 LI R2 >69B6
69AC CLR R1
69AE LI R11 >0080
69B2 B @>659C

```

and change the following memory locations

```

699A from 0D0A to 4344
699C " 434F " 4F53
699E " 5254 " 2031
69A0 " 4558 " 0D0A
69A2 " 2044 " 0000
69F0 " 0000 " 4056
69F2 " 0000 " 69A4

```

To make the routine auto load a program from the BOOT command change the following memory locations:-

```
6996 from 021C to 69A8
```

And from 69B6 onwards add the ascii codes for the filename required (maximum of 8 characters) terminating with 0000.

After making the required changes re-save the SYSTEM\$ file from the monitor using the D command:-

```
start addr 6900 end addr 6EFE entry point 6980, IDT - SYSTEM$,
auto run - Y.
```

The * command works by simply typing *<filename> with no delimiters or quotes.

To load program files from a machine code routine use the following:-

```

LI R1 ><drive number>
LI R2 ><start of filename>
LI R11 >0080
B @>659C

```

And finally a hardware modification for those of you who are brave enough to venture within the Cortex case.

If pin 4 of the keyboard socket is connected to pin 4 (D0) of IC63 the keyboard can be read directly in basic without the problem of random values being returned or having to press the repeat key, use the following line:-

```
BASE 16:K=KEY[0]:IF CRB [-8]=1 THEN K=CRF[8]
```

- - - x x x - - -

PROGRAMS

Written any good programs? Then send them in for us to print! All the following have been sent in by Cortex Users. Each listing has been thoroughly checked against the original.

Helge A Larsen
Stathelle
NORWAY

Our apologies to Mr Larsen for the assumption that Helge was a female name (Ref Newsletter II). He has written in to correct us on this point, and has sent two useful routines.

Obviously my name is unknown in England since I've been a female in the last newsletter. Helge is in Norway a male's name, Helga is as female name.

I've found a short routine to read the keyboard, and written a routine for scrolling one or more lines right pixel by pixel such that characters leaving the right side will enter at the left side.

Keyboard reading routine

```
6000 020C LI R12,0010
6004 3601 STCR R1,8
6006 3602 STCR R2,8
6008 3603 STCR R3,8
600A 3604 STCR R4,8
600C 9081 CB R1,R2
600E 1608 JNE 6020
6010 90C2 CB R2,R3
6012 1606 JNE 6020
6014 9103 CB R3,R4
6016 1604 JNE 6020
6018 05C0 INCT R0
601A 0580 INC R0
601C D401 MOV B R1,+R0
601E 0380 RTWP
6020 05C0 INCT R0
6022 04D0 CLR +R0
6024 0380 RTWP
```

Use of routine:

```
10 K=0
20 CALL 6000H,ADR(K)
30 PRINT K: GOTO 20
```

Scroll Routine

```
6000 06C0 SWPB R0
6002 C200 MOV R0,R8
6004 C181 MOV R1,R6
6006 0201 LI R1,>C000
600A 06A0 BL @>05F2
```

```

600E DC60  MOVB  @>F120,*R1+
6012 0281  CI    R1,>C100
6016 16FB  JNE   >600E
6018 0201  LI    R1,>C0F8
601C 0202  LI    R2,BFF0
6020 0203  LI    R3,0008
6024 D131  MOVB  +R1+,R4
6026 0994  SRL   R4,9
6028 1802  JOC   >602E
602A 04C5  CLR   R5
602C 1001  JMP   >6030
602E 0705  SETO  R5
6030 CC85  MOV   R5,*R2+
6032 0603  DEC   R3
6034 16F7  JNE   >6024
6036 0201  LI    R1,>C000
603A 0202  LI    R2,>BFF0
603E 0203  LI    R3,>0008
6042 C152  MOV   *R2,R5
6044 1606  JNE   >6052
6046 D111  MOVB  *R1,R4
6048 0994  SRL   R4,9
604A 181D  JOC   >6086
604C 04C5  CLR   R5
604E 06C4  SWPB  R4
6050 1007  JMP   >6060
6052 D111  MOVB  *R1,R4
6054 0994  SRL   R4,9
6056 1819  JOC   >608A
6058 04C5  CLR   R5
605A 06C4  SWPB  R4
605C 0264  ORI   R1,>8000
6060 DC44  MOVB  R4,*R1+
6062 CC85  MOV   R5,*R2+
6064 0603  DEC   R3
6066 13E9  JEQ   >603A
6068 0281  CI    R1,>C100
606C 11EA  JLT   >6042
606E 0201  LI    R1,>C000
6072 0268  ORI   R8,>4000
6076 06A0  BL    @>05F2
607A D831  MOVB  +R1+,@>F120
607E 0281  CI    R1,>C100
6082 16FB  JNE   >607A
6084 1004  JMP   >608E
6086 0705  SETO  R5
6088 10E2  JMP   >604E
608A 0705  SETO  R5
608C 10E6  JMP   >605A
608E 0248  ANDI  R8,>BFFF
6092 0228  AI    R8,>0100
6096 0606  DEC   R6
6098 16B6  JNE   >6006
609A 0380  RTWP

```

CALL 6000H,L,N

L = first line to be scrolled (upper 0)

N = number of lines to be scrolled.

Mr R M Lee
Eastbourne
Sussex

Robert not only solves other users problems (see bug byte section), but is also a keen programmer himself.

FLASHING CURSOR

This program simply makes the cursor flash on the screen, but only when the computer is not accessing the screen i.e. in IDLE state. The entry point when saving is 6064H. The routine is called every 10ms by the clock routine so do not overwrite it before removing the vector jump.

```
6000=EEO0
6002=6004
6004 0587 INC R7
6006 0287 CI R7,>0032
600A 1A26 JL >6058
600C C1AD MOV @>001C(R13),R6
6010 0286 CI R6,>046C
6014 1621 JNE >6058
6016 04C7 CLR R7
6018 0201 LI R1,>F120
601C 0200 LI R0,>F121
6020 0202 LI R2,>0008
6024 C0C2 MOV R2,R3
6026 0208 LI R8,>0BF8
602A 0205 LI R5,>EEC0
602E 06A0 BL @>05F2
6032 DD51 MOVB *R1,*R5+
6034 0602 DEC R2
6036 16FD JNE >6032
6038 0202 LI R2,>0004
603C 0205 LI R5,>EEC0
6040 0575 INV *R5+
6042 0602 DEC R2
6044 16FD JNE >6040
6046 0208 LI R8,>4BF8
604A 0205 LI R5,>EEC0
604E 06A0 BL @>05F2
6052 D475 MOVB *R5+,*R1
6054 0603 DEC R3
6056 16FD JNE >6052
6058 0380 RTWP
605A 0420 BLWP @>6000
605E 0300 LIM1 >0000
6062 C820 MOV @>605A,@>010E
6068 C820 MOV @>605C,@>0110
606E 0300 LIM1 >000F
6072 0460 B @>021C
```

Here is a program that enables GRAPH mode screen to printer dumps for an RX-80 printer, but there is no reason why it should not work for any other new EPSON printer. The program is written for a centronics printer with UNIT 4,

but to change to RS232 UNIT 2, change location 6024H to 2H. The entry point when saving the program is 60EAH. This routine sets up the word "DUMP" in the computer's symbol table and allows the statement "DUMP" to be used whenever the program is needed e.g.

```
10 GRAPH
20 DUMP
30 END
```

Some problems may occur if the BASIC program is very large, as the DUMP program uses some 6K of memory just above the users BASIC program.

SCREEN TO PRINTER DUMP

```
6000 0202 LI R2,>F120
6004 0203 LI R3,>F121
6008 0204 LI R4,>1800
600C 0208 LI R8,>0000
6010 C1A0 MOV @>EFC0,R6
6014 06A0 BL @>05F2
6018 DD92 MOVB *R2,*R6+
601A 0604 DEC R4
601C 16FD JNE >6018
601E 020A LI R10,>0018
6022 0200 LI R0,>0008
6026 C800 MOV R0,@>001E
602A 0200 LI R0,>0A00
602E 0F00 WRIT R0
6030 0200 LI R0,>1B00
6034 0F00 WRIT R0
6036 0200 LI R0,>4100
603A 0F00 WRIT R0
603C 0200 LI R0,>0800
6040 0F00 WRIT R0
6042 0200 LI R0,>1B00
6046 0F00 WRIT R0
6048 0200 LI R0,>6C00
604C 0F00 WRIT R0
604E 0200 LI R0,>1300
6052 0F00 WRIT R0
6054 0200 LI R0,>0008
6058 C060 MOV @>EFC0,R1
605C 0202 LI R2,>1800
6060 0203 LI R3,>1B00
6064 0F03 WRIT R3
6066 0203 LI R3,>4B00
606A 0F03 WRIT R3
606C 0203 LI R3,>0000
6070 0F03 WRIT R3
6072 0203 LI R3,>0100
6076 0F03 WRIT R3
6078 0203 LI R3,>0018
```

```

607C 0204 LI R4,>0020
6080 0205 LI R5,>0008
6084 C185 MOV R5,R6
6086 C1C1 MOV R1,R7
6088 C005 MOV R5,R0
608A 04C9 CLR R9
608C 0A19 SLA R9,1
608E D231 MOVB *R1+,R8
6090 06C8 SWPB R8
6092 0908 SRL R8,0
6094 1701 JNC >6098
6096 0589 INC R9
6098 0606 DEC R6
609A 16F8 JNE >608C
609C 0A89 SLA R9,8
609E 0F09 WRIT R9
60A0 C185 MOV R5,R6
60A2 C047 MOV R7,R1
60A4 0600 DEC R0
60A6 16F2 JNE >608C
60A8 A045 A R5,R1
60AA 6085 S R5,R2
60AC 0604 DEC R4
60AE 16EB JNE >6086
60B0 0200 LI R0,>0A00
60B4 0F00 WRIT R0
60B6 060A DEC R10
60B8 16D3 JNE >6060
60BA 0200 LI R0,>1B00
60BE 0F00 WRIT R0
60C0 0200 LI R0,>6C00
60C4 0F00 WRIT R0
60C6 0200 LI R0,>0000
60CA 0F00 WRIT R0
60CC 0200 LI R0,>1B00
60D0 0F00 WRIT R0
60D2 0200 LI R0,>3200
60D6 0F00 WRIT R0
60D8 0200 LI R0,>0001
60DC C800 MOV R0,@>001E
60E0 0460 B @>3F30
60E4 6D49 S R9,*R5+
60E6 0020 DATA >0020
60E8 6000 S R0,R0
60EA C820 MOV @>60E4,@>3A88
60F0 C820 MOV @>60E6,@>3B34
60F6 C820 MOV @>60E8,@>4026
60FC 0460 B @>021C

```

- - - x x x - - -

P Edwards
Cochrane Park
Newcastle-upon-Tyne
NE7 7LL

Paul is obviously not content with just writing programs.
He has found a way of adding new commands.

Here is the assembler listing for a 'FIND' command to be added to the CORTEX dictionary. To relocate the program to another address, the absolute addresses at lines 15, 28, 31, 47 and 63 will need to be changed accordingly.

If any CDOS users would like to add this, or any other M/Code routine to their BOOT file, I would be happy to supply details of how I have done it.

Has any CDOS user got a 2 pass assembler written for sharing?

```

1 ;      The 'FIND' command is entered into the command
2 ;      table by a call to 'INIT' at the end of this
3 ;      listing, and the command can then be used at any
4 ;      time.
5 ;      The syntax is
6 ;              FIND STRING<CR>
7 ;
8 ;      This will search the program and print out any
9 ;      program line containing STRING.
10
11          section find
12          6200          org          6200h
13
14          6200 0200EB09      li          r0,0eb09h      ;start of string to be found
15          6204 02016286      li          r1,store      ;temp. store for string
16          6208 DC70          fin10  movb      *r0+,*r1+      ;store byte
17          620A 16FE          jne          fin10
18          620C 0FA0556F      msg          556fh          ;new line
19          6210 04C1          clr          r1
20          6212 0706          seto        r6
21          6214 C220EFBC      mov          0efbch,r8      ;pointer into basic tables
22          6218 0648          dect        r8
23          621A 8808EFBA      fin20  c          r8,0efbah      ;end of table ?
24          621E 1227          jle          out          ;out
25          6220 0228FFFC      ai          r8,0fffch      ;r8 = r8-4
26          6224 8601          c          r1,*r8
27          6226 15F9          jgt         fin20          ;fin20
28          6228 06A06250      fin30  bl          decode      ;get prog. line in ascii
29
30          622C 0202EB04      li          r2,0eb04h      ;start of line of program
31          6230 02016286      fin35  li          r1,store      ;point to search string
32          6234 D492          fin40  movb      *r2,*r2      ;end of line ?
33          6236 13F8          jeq         fin30
34          6238 9C91          cb          *r1,*r2+      ;match ?
35          623A 16FC          jne          fin40          ;no, try next one
36          623C 0581          fin50  inc          r1
37          623E D451          movb      *r1,*r1          ;end of search string ?
38          6240 1303          jeq         found          ;yes
39          6242 9C91          cb          *r1,*r2+      ;match ?
40          6244 16F5          jne          fin35
41          6246 10FA          jmp         fin50
42          6248 000A          found  word      10          ;write line
43          624A 0FA0556F      msg          556fh          ;new line
44          624E 10EC          jmp         fin30

```

```

45
46          ; **** subroutine to decode basic line ****
47 6250 C80B62EA decode mov r11,store2 ;save return vector
48 6254 8808EFBA c r8,0efbah ;end of table ?
49 6258 1A0A jl out ;out
50 625A C078 mov *r8+,r1
51 625C 8181 c r1,r6
52 625E 1B07 jh out ;out
53 6260 06A03C80 bl 3c80h ;decode line
54
55 6264 0228FFFA ai r8,0fffah
56 6268 C2E062EA mov store2,r11 ;regain return vector
57 626C 045B rt ;return
58
59 626E 0460021C out b 21ch ;back to basic 'Ready'
60
61 6272 0201724C INIT li r1,724ch ;'FIND' coded
62 6276 C8013A20 mov r1,3a20h ;put it into table
63 627A 02016200 li r1,find ;start of prog
64 627E C8013ACC mov r1,3acch ;into table
65 6282 0460021C b 21ch ;back to Basic
66
67 6286 64 store block 100
68 62EA 0000 store2 word 0
69 nolist sym
70 end

```

- - - x x x - - -

Mr R A Green
Rotherham
South Yorkshire

Mr Green has sent us two programs, both with a view to the stock exchange.

Please try my Shares programs, they may not make money but do show where it went. The Histogram program can be altered very easily by anyone to show any data in histogram form.

```

10 ; TAB (10) "Shares Program."
20 ;
30 ; TAB (10) "Enter Latest Prices."
33 ;
35 ; TAB (10) "In Pounds."
40 ;
50 INPUT " Example Are"A4
60 ;
161 INPUT " British Telecom are"G4
162 ;
164 ;
170 DIM $X[2]
180 INPUT "Prices are for "$X[0]
190 DIM $A[4],$B[4],$C[4],$D[4]
200 DIM $L[4],$R[4],$S[4]
210 DIM Z1[3],Z2[3],Z3[3],Z4[3].Z5[3],Z6[3],Z7[3],Z8[3],Z9[3]

```

```

220 $R[0]="SS,SSS"
230 $S[0]="SS,SSS.99"
240 DATA "EXAMPLE",500,"B1234","22/2/83",124,630,A4
245 DATA " " " " " ,100,"BB123","2/1/85",0,0,A4
375 DATA "BRITISH TELECOM",800,"XXXXXXXXX","23/11/84",50,400,G4
380 DATA "END",0,"END","END",0,0,0
390 T1=18: T2=28: T3=38: T4=48: T5=58: T6=68: T7=78: T8=88
400 UNIT 2: ; "<1B><75><14>"
410 PRINT TAB (40)"FINANCIAL REVIEW"; TAB (80)$X[0]
420 PRINT
430 ; TAB (T1)"SHARE" TAB (T2)"CERT"; TAB (T3)"DATE"; TAB (T4)
"BOUGHT"; TAB(T5)"TOTAL";
440 ; TAB (T6)"CURRENT"; TAB (T7)"PRESENT"; TAB (T8)"GROSS"
450 ; TAB (T1)"HOLDING"; TAB (T2)"NUMBER"; TAB (T3)"BOUGHT";
TAB (T4)"PRICE";TAB (T5)"COST";
460 ; TAB (T6)"PRICE"; TAB (T7)"VALUE"; TAB (T8)"PROFIT"
470 Z5=0: Z6=0
480 READ $A[0],Z1,$B[0],$C[0],Z2,Z3,Z4
490 IF $A[0]="END" THEN GOTO 560
500 ;
510 ; # $R[0];$A[0]; TAB (T1);Z1; TAB (T2)$B[0]; TAB (T3)$C[0];
TAB (T4);Z2; TAB (T5):Z3;
520 ; # $S[0]; TAB (T6-2);Z4; TAB (T7-2); Z4*Z1; TAB (T8-2);
TAB (T8-2);(Z4*Z1)-Z3
530 Z5=(Z4*Z1)+Z5
540 Z6=(Z4*Z1)-Z3+Z6
550 GOTO 480
560 ; "<1B><75><1>"
570 ;
580 ; # $R[0]; TAB (30);"TOTAL VALUE IS"Z5
590 ;
600 ; # $R[0]; TAB (30);"TOTAL PROFIT IS"Z6
610 UNIT -2
620 INPUT " ANOTHER PRINT Y/N?"$D[0]
630 IF $D[0]="N" THEN END
640 IF $D[0]="Y" THEN RESTOR
645 GOTO 400
650 STOP

670 ; "Further shares may be added as"
675 ; "in the example by inserting one"
680 ; "INPUT line and one line of"
685 ; "appropriate DATA."
690 ; "Scrip issues are dealt with as"
695 ; "shown by adding one line of DATA.""700;"Line 400 changes
my printer to"
705 ; "96 columns."
710 ; "Line 560 changes it back to 80."

```

```

10 REM HISTOGRAM PROGRAM
20 DIM $D[4],$[2],S[12]
30 INPUT " Name Of Share Is "$D[0]
40 INPUT " Present Year Is 198"P
50 INPUT " Present Month Is "$C[0]
60 ; " Input Previous 12 Months Prices In"
70 ; " Pounds Starting With 12 Months Ago"

```

```

80  INPUT S[0],S[1],S[2],S[3],S[4],S[5],S[6],S[7],S[8],S[9],
    S[10],S[11]
90  UNIT 2
100 ; TAB (20)$D[0]
101 ;
110 ; TAB (15)"SHARE PRICE FOR LAST TWELVE MONTHS"
120 ; TAB (15)"-----"
130 REM FIND MAX VALUE OF Y AXIS
140 M=S[0]
150 FOR J=0 TO 11 !REM NO OF VALUES TO PLOT
160 IF S[J]<M THEN GOTO 180
170 ELSE M=S[J]
180 NEXT J
190 REM MIN VALUE TO PLOT
200 X=S[0]
210 FOR J=0 TO 11
220 IF S[J]>X THEN GOTO 240
230 ELSE X=S[J]
240 NEXT J
250 X=X-0.15
260 ; "POUNDS"
270 ;
280 Y=(M-X)/20
290 FOR I=M TO X STEP -Y
300 ; #"9.99" I; TAB (5) ":";
310 B=2
320 REM PRINT HISTOGRAM
330 FOR J=0 TO 11
340 B=B+5 !REM SPACING FOR COLUMNS
350 IF S[J]<I THEN GOTO 380
360 ; TAB (B) "####";
370 GOTO 390
380 ; " ";
390 NEXT J
400 ; TAB (66) ":"
410 NEXT I
420 ; "-----:-----"
    "-----:"
452 Z=1980+P
430 ; TAB (30)"MONTHS";TAB (61)$C[0];" "Z
431 ;
440 T=2
450 FOR J=0 TO 11
460 T=T+5
470 ; #"9.99"; TAB (T);S[J];
475 S[J]=0
480 NEXT J
490 ;
500 UNIT -2
510 END

```

- - - x x x - - -

C J Young
Lossiemouth
Morayshire

Chris has been very busy on his Cortex. Here is his version of a multicolour mode. It is relocatable, and is stored in an array. This means that you do not have to use the NEW function, and it also works with CDOS.

```

10  REM *****
20  REM *
30  REM * Multicolour Mode *
40  REM * (Fully Relocatable *
50  REM * Machine Code) *
60  REM * by *
70  REM * Chris Young *
80  REM *
90  REM *****
100 REM
110 REM * Sprites Can Also Be used in
120 REM * multicolour mode
130 REM
140 DIM MC[30]
150 MLT=ADR[MC[0]] !Set up Address
160 MTP=MLT+046H !Pixel Address
170 REM
180 REM *****
190 REM * Read Machine Code *
200 REM *****
210 REM
220 FOR X=MLT TO MLT+0ACH STEP 2
230 READ A
240 MWD[X]=A
250 NEXT X
260 CALL MLT !Set Up Screen
270 REM
280 REM *****
290 REM * Randomize Pixel Colours *
300 REM *****
310 REM
320 FOR Y=0 TO 47
330 FOR X=0 TO 63
340 CALL MTP,X,Y,INT[RND*16]
350 NEXT X
360 NEXT Y
370 GOTO 320
380 REM
390 REM * Set up Screen Data *
400 REM
410 DATA 010H,0200H,0C800H,0D800H
420 DATA 0F121H,0201H,08100H
430 DATA 0D801H,0F121H,04C2H,0C202H
440 DATA 0228H,01800H,0C042H,0241H
450 DATA 01FH,0C0C2H,0973H,0A53H
460 DATA 0A043H,0268H,04000H,06A0H
470 DATA 05F2H,06C1H,0D801H,0F120H
480 DATA 06C1H,0248H,03FFFH,0582H
490 DATA 0282H,0300H,011E8H,0380H
500 REM
510 REM * Set Pixel *
520 REM
530 DATA 0C200H,0281H,0FH
540 DATA 01501H,01008H,0281H,01FH

```

```

550 DATA 01503H,0221H,040H,01002H
560 DATA 0221H,080H,0C148H,0245H,01H
570 DATA 0C0C8H,0913H,0C101H,0934H
580 DATA 0A54H,0C1C4H,0A33H,0A37H
590 DATA 0A1C3H,0C181H,0246H,07H
600 DATA 0A1C6H,0C207H,06A0H,05F2H
610 DATA 0D060H,0F120H,06C1H,0C145H
620 DATA 0150CH,0A42H,0241H,0FH
630 DATA 0A042H,0268H,04000H,06A0H
640 DATA 05F2H,06C1H,0D801H,0F120H
650 DATA 0380H,0241H,0F0H,010F4H

```

Use of multicolour mode:

MLT machine code call sets up the multicolour mode

MTP requires 3 parameters X,Y,Colour
 X has the range 0 to 63
 Y has the range 0 to 47
 Colour is from 0 to 15

- - - x x x - - -

Mr M D Rudnicki
 Bognor Regis
 Sussex

Mark is the author of one of our best selling games - 'Burglar'. However, he is obviously keen to help the rest to us with further extensions to his graphics commands, which were included in Newsletter II.

Here are extensions to my graphics commands, and a basic loader for them, as requested by several people who wrote to me following the appearance in the last newsletter of the first commands.

FULL MACHINE CODE LOADER FOR GRAPHICS ROUTINES.

```

10000 RESTOR 10070
10010 T=0
10020 FOR I=06200H TO 06430H STEP 2
10030 READ A: MWD(I)=A: T=T+A
10040 NEXT I
10050 IF T=-133059 THEN RETURN
10060 PRINT"ERROR IN MACHINE CODE":END
10070 DATA 512,88,-10240,-3807,1728,-10240,-3807,-15344
10080 DATA 512,768,513,8224,-10239,-3808,-15279,1536
10090 DATA 7163,896,4096,-10230,-3808,1738,-14694
10100 DATA -10230,-3808,-14694,1115,1728,-10240,-3807,1728
10110 DATA -10240,-3807,-15344,1115,576,255,2608,608
10120 DATA 16384,521,3,1696,25144,-15743,1696,25128
10130 DATA -15742,1696,25128,-15741,1696,25128,-15740,1696
10140 DATA 25128,544,2048,1545,7150,544,2048,521
10150 DATA 3,1696,25144,-15739,1696,25128,-15738,1696
10160 DATA 25128,-15737,1696,25128,-15736,1696,25128,544
10170 DATA 2048,1545,7150,896,544,6144,608,16384

```


10180 DATA 1696,25144,-15344,1729,-10239,-3808,1729,896
 10190 DATA 544,6144,1696,25144,1473,1409,-11168,-3808
 10200 DATA 896,0,0,0,576,255,2608,544
 10210 DATA 30720,1696,25144,-15743,1696,25128,-15742,1696
 10220 DATA 25128,-15741,1696,25128,-15740,1696,25128,896
 10230 DATA 576,31,2592,544,23296,1696,25144,577
 10240 DATA 255,1730,578,-256,-24510,-15743,1696,25128
 10250 DATA 1731,579,-256,580,15,-24317,-15740,1696
 10260 DATA 25128,896,0,0,576,31,2592,544
 10270 DATA 23296,1696,25144,577,255,1730,578,-256
 10280 DATA -24510,-15743,1696,25128,896,0,0,0
 10290 DATA -14335,-4064,-14334,-4054,-14333,-4052,-14332,-4050
 10300 DATA -14331,-4048,576,255,2576,-16320,2576,-24512
 10310 DATA 545,23278,-16207,-16143,-16111,1221,1222,1223
 10320 DATA 1224,-16382,576,-1024,-16064,-16382,576,1008
 10330 DATA 2336,-24256,578,15,2754,-15998,-16381,576
 10340 DATA -16384,2368,-24192,-16381,576,16128,2400,-24192
 10350 DATA -15933,583,252,2695,579,3,2659,-24125
 10360 DATA -16380,576,-4096,2464,-24128,-15868,584,4032
 10370 DATA 2632,580,63,2596,-24060,-14331,-4062,-14330
 10380 DATA -4060,-14329,-4058,-14328,-4056,1056,25600,896
 10390 DATA -4064,25160,-4064,25256,0,0,0,0
 10400 DATA -14336,-4064,1218,-12111,5633,896,1730,-14334
 10410 DATA -4062,1056,25604,1408,4339,0,0,0
 10420 DATA 0,0,0,0,0,0,0,0

The commands these routines support are as follows:

1. Initialise CALL 06200H as in the last Newsletter.
2. Set char. CALL 06248H,Chr,S1,S2,S3,S4,C1,C2,C3,C4
3. Put char. CALL 062A8H,Pos,Chr
4. Get char. CALL 062C0H,Pos,ADR(var)
5. Shape. CALL 062D8H,Shape number,S1,S2,S3,S4
This is like the SHAPE command, and is used to define a sprite pattern from machine code.
6. Sprite. CALL 06300H,No,X,Y,Shape,Col.
This is like the SPRITE command, and follows exactly the same syntax, for sprite movement in machine code.
7. Move Sprite CALL 6338H,No,X,Y
This just sets new co-ords for a sprite, from machine code.
8. Character. CALL 6360H,N1,N2,C1,C2,C3,C4
This defines a character N2 from the Cortex character set, number N1, with colours C1-4.
9. Text CALL 6410H,Pos,ADE(\$string)
Prints out the contents of \$string in the new mode, provided all the necessary characters have been predefined e.g. using the Character command.

Tim Gray
Wolverhampton
West Midlands

Tim is rapidly threatening to take over these newsletters with the flood of articles and information he has been sending.

Here is a sample of his software.

SHAPE is a shape design programme that allows you to design up to 9 shapes at a time and also writes its own DATA lines to save you having to make notes. All that is then required is to PURGE the main program lines and use the data lines for your next programme.

On running the shape generation programme 9 8x8 blocks are displayed on the screen, use the cursor arrows to select a pixel and INS/DEL to set or reset it after all 9 shapes have been designed press RETURN and the programme displays the numbers of the data lines generated for each shape and restarts with a clean block of 9.

```
5      LNO=6001: KL=0
10     COLOUR 15,12: GRAPH:X1=64: Y1=-1
13     DIM A[9,4]
18     DIM $Q[9,4,2]: DIM $LIN[20]: DIM $S[2]
19     REM ***PLOT GRAPHICS***
20     SHAPE 255, 0FF81H,08181H,08181H,081FFH
30     A=4: COLOUR 1,9: GOSUB 1300
40     A=12: COLOUR 1,6: GOSUB 1300
50     A=20: COLOUR 1,9: GOSUB 1300
60     A=260: COLOUR 1,6: GOSUB 1300
70     A=268: COLOUR 1,9: GOSUB 1300
80     A=276: COLOUR 1,6: GOSUB 1300
90     A=516: COLOUR 1,9: GOSUB 1300
100    A=524: COLOUR 1,6: GOSUB 1300
110    A=532: COLOUR 1,9: GOSUB 1300
130    COLOUR 1,12: PRINT @(0,0);"DATA": PRINT "LINE":PRINT
      No's": PRINT:PRINT
140    FOR N=1 TO 9: FOR M=1 TO 4: A[N,M]=0:NEXT M: NEXT N
149    REM ***GOTO CURSER***
150    GOSUB 2000
155    REM *** ENTER THE DATA ***
160    LNO=LNO+10
170    KL=LNO
190    FOR N=1 TO 9
195    LET $S[0]=KL
200    FOR M=1 TO 4
210    $Q[N,M,1]=A[N,M]
220    NEXT M
230    $LIN[0]=$S[0]+"DATA"+$Q[N,1,1]+", "+$Q[N,2,2]+", "
      +$Q[N,3,1]+", "+$Q[N,4,1]
235    ENTER $LIN[0]
236    COLOUR 1,12: PRINT "    "$S[0]
240    KL=KL+1
255    NEXT N
```

```

260 PRINT : ; : ; " PRESS " : ; " ANY" : ; " KEY"
270 K=KEY[0]
280 IF K=0 THEN GOTO 270
290 REM ***BACK TO START AGAIN***
300 GOTO 10
1190 REM ***STOP***
1200 STOP
1300 REM ***GRID SUBROUTINE***
1330 FOR Y=A+4 TO A+228 STEP 32
1340 FOR X=0 TO 7
1350 SPUT Y+X,255
1360 NEXT X
1370 NEXT Y
1380 RETURN

2000 REM *** CURSER CONTROL ***
2010 K=KEY[0]
2020 IF K=8 THEN X1=X1-8: IF X1<64 THEN X1=64
2030 IF K=9 THEN X1=X1+8: IF X1>248 THEN X1=248
2040 IF K=10 THEN Y1=Y1+8: IF Y1>183 THEN Y1=183
2045 IF K=11 THEN Y1=Y1-8: IF Y1<-1 THEN Y1=-1
2049 REM *** CURSER ***
2050 SPRITE 0,X1,Y1,255,14
2052 REM *** BIT POSITION CALCULATE ***
2055 C1=INT[(X1/8+4)+(Y1/8*32)]
2080 CA=X1/64
2090 CB=(Y1+65)/64
2100 CC=((INT[CA]*3)-3+(INT[CB]
2110 CD=(FRA[CA])*8
2120 CE=(FAC[CB])*8
2130 CF=INT[(CE+2)/2]
2140 CG=FRA[(CE)/2]*16
2150 CH=CG+CD+16
2155 REM *** SET BIT ***
2160 IF K=22 THEN COLOUR 1,15: SPUT C1,255: COLOUR 15,12:
BIT[A[CC,CF],CH]=1
2165 REM *** POP BIT ***
2170 IF K=23 THEN COLOUR 1,6: SPUT C1,255: COLOUR 15,12:
BIT[A[CC,CF],CH]=0
2175 REM *** RETURN TO ENTER DATA ***
2180 IF K=13 THEN RETURN
2190 GOTO 2010
2200 REM ***** DATA LINES *****

```

FILL 2 is a machine code fill routine slightly different to most because it first looks to see if the pixel at the starting point is set and then uses PLOT to fill the shape. if it is or UNPLOT to unfill the shape if it is not.

CALL 7000H,x,y

Where x and y are the starting location within the shape.

The routine will fill the shape up until it finds a different colour but will return unfinished if it gets to an off screen location or runs out of stack memory.

FILL 2

FILL/UNFILL ROUTINE

7000	C0A0	MOV	@>0026,R2	708C	1D01	SBO	1
7004	1602	JNE	>700A	708E	1001	JMP	>7092
7006	2FA0	XOP	@>0030,14	7090	1E01	SBZ	1
700A	C820	MOV	@>1CBB,@>F020	7092	0641	DECT	R1
7010	C1E0	MOV	@>EFC0,R7	7094	06A0	BL	@>70EE
7014	020C	L1	R12,>1EF0	7098	80C2	C	R2,R3
7018	06A0	BL	@>70EE	709A	1606	JNE	>70A8
701C	06A0	BL	@>7188	709C	1F02	TB	2
7020	C0C2	MOV	R2,R3	709E	1305	JEQ	>70AA
7022	C100	MOV	R0,R4	70A0	06A0	BL	@>7174
7024	C141	MOV	R1,R5	70A4	1D02	SBO	2
7026	1E01	SBZ	1	70A6	1001	JMP	>70AA
7028	1E02	SBZ	2	70A8	1E02	SBZ	2
702A	0581	INC	R1	70AA	0581	INC	R1
702C	06A0	BL	@>70EE	70AC	0600	DEC	R0
7030	80C2	C	R2,R3	70AE	06A0	BL	@>70EE
7032	1606	JNE	>7040	70B2	80C2	C	R2,R3
7034	1F01	TB	1	70B4	13E2	JEQ	>707A
7036	1305	JEQ	>7042	70B6	0A80	SLA	R0,8
7038	06A0	BL	@>7174	70B8	0A81	SLA	R1,8
703C	1D01	SBO	1	70BA	0A86	SLA	R6,8
703E	1001	JMP	>7042	70BC	D800	MOVB	R0,@>EE36
7040	1E01	SBZ	1	70C0	D806	MOVB	R6,@>EE96
7042	0641	DECT	R1	70C4	D801	MOVB	R1,@>EE37
7044	06A0	BL	@>70EE	70C8	D801	MOVB	R1,@>EE97
7048	80C2	C	R2,R3	70CC	0420	BLWP	@>51BE
704A	1606	JNE	>7058	70D0	0647	DECT	R7
704C	1F02	TB	2	70D2	8807	D	R7,@>EFC0
704E	1305	JEQ	>705A	70D6	1A07	JL	>70E6
7050	06A0	BL	@>7174	70D8	04C0	CLR	R0
7054	1D02	SBO	2	70DA	C057	MOV	*R7,R1
7056	1001	JMP	>705A	70DC	D001	MOVB	R1,R0
7058	1E02	SBZ	2	70DE	0980	SRL	R0,8
705A	0581	INC	R1	70E0	0241	ANDI	R1,>00FF
705C	0580	INC	R0	70E4	109E	JMP	>7022
705E	06A0	BL	@>70EE	70E6	C820	MOV	@>F020,@>1CB8
7062	80C2	C	R2,R3	70EC	0380	RTWP	
7064	13E2	JEQ	>702A	70EE	0280	CI	R0,>0100
7066	C180	MOV	R0,R6	70F2	14F9	JHE	>70E6
7068	C004	MOV	R4,R0	70F4	0281	CI	R1,>00C0
706A	C045	MOV	R5,R1	70F8	14F6	JHE	>70E6
706C	1E01	SBZ	1	70FA	C241	MOV	R1,R9
706E	1E02	SBZ	2	70FC	C281	MOV	R1,R10
7070	0600	DEC	R0	70FE	C200	MOV	R0,R8
7072	06A0	BL	@>70EE	7100	0939	SRL	R9,3
7076	80C2	C	R2,R3	7102	0A89	SLA	R9,8
7078	161E	JNE	>70B6	7104	024A	ANDI	R10,>0007
707A	0581	INC	R1	7108	A24A	A	R10,R9
707C	06A0	BL	@>70EE	710A	0248	ANDI	R8,>FFF8
7080	80C2	C	R2,R3	710E	A209	A	R9,R8
7082	1606	JNE	>7090	7110	06C8	SWPB	R8
7084	1F01	TB	1	7112	D808	MOVB	R8,@>F121
7086	1305	JEQ	>7092	7116	06C8	SWPB	R8
7088	06A0	BL	@>7174	7118	D808	MOVB	R8,@>F121

711C	C280	MOV	R0,R10	7162	C410	MOV	*R0,*R0
711E	04C9	CLR	R9	7164	D0A0	MOVB	@>F120,R2
7120	024A	ANDI	R10,>0007	7168	0242	ANDI	R2,>F000
7124	D0A0	MOVB	@>F120,R2	716C	09C2	SRL	R2,12
7128	D26A	MOVB	@>1D42(R10),R9	716E	0262	ORI	R2,>8000
712C	2089	COC	R9,R2	7172	045B	RT	
712E	1310	JEQ	>7150	7174	8807	C	R7,@>EFC2
7130	04C2	CLR	R2	7178	1306	JEQ	>7186
7132	0228	AI	R8,>2000	717A	0A80	SLA	R0,8
7136	06C8	SWPB	R8	717C	D040	MOVB	R0,R1
7138	D808	MOVB	R8,@>F121	717E	CDC1	MOV	R1,*R7+
713C	06C8	SWPB	R8	7180	0241	ANDI	R1,>00FF
713E	D808	MOVB	R8,@>F121	7184	0980	SRL	R0,8
7142	C410	MOV	*R0,*R0	7186	045B	RT	
7144	D0A0	MOVB	@>F120,R2	7188	0204	LI	R4,>8000
7148	0242	ANDI	R2,>0F00	718C	2084	COC	R4,R2
714C	0982	SRL	R2,8	718E	1305	JEQ	>719A
714E	045B	RT		7190	0204	LI	R4,>F009
7150	04C2	CLR	R2	7194	C804	MOV	R4,@>1CB8
7152	0228	AI	R8,>2000	7198	045B	RT	
7156	06C8	SWPB	R8	719A	0204	LI	R4,>5009
7158	D808	MOVB	R8,@>F121	719E	C804	MOV	R4,@>1CB8
715C	06C8	SWPB	R8	71A2	045B	RT	
715E	D808	MOVB	R8,@>F121				

OVERPLOT is a routine that allows plotting of foreground colours only without effecting background colours.

It can be positioned anywhere in memory and is used by changing the word at 1D36 from 05F2 to the start address of this routine 5EBA in this case and back again for normal plotting.

OVERPLOT ROUTINE

5EBA	0248	ANDI	R8,>BFFF
5EBE	06A0	BL	@>05F2
5EC2	8F3C	C	*R12+,*R12+
5EC4	D020	MOVB	@>F120,R0
5EC8	0940	SRL	R0,4
5ECA	D120	MOVB	@>EE95,R4
5ECE	0944	SRL	R4,4
5ED0	D00\$	MOVB	R4,R0
5ED2	0A40	SLA	R0,4
5ED4	0268	ORI	R8,>4000
5ED8	06A0	BL	@>05F2
5EDC	8F3C	C	:R12+,*R12+
5EDE	D800	MOVB	R0,@>F120
5EE2	0380	RTWP	

- - - x x x - - -

BUG BYTES

Your problems/solutions

Mr R M Lee
Eastbourne
Sussex

Robert has solved one of the problems posed in the last newsletter.

The solution to Gary Alexander's clicking problem is very simple. The cause is the 9995 internal decremter which interrupts every 10ms causing funny clicks.

To solve this the clock has to be stopped. To do this in M/C requires R12=1EE0H and then SBZ 1 to stop the clock. SBO 1 re-enables it.

- - - x x x - - -

Julian Terry
Rainham
Kent

Julian has also written, at some length, about two points raised in the last newsletter.

With reference to C C Kuan's unrandom Random

As many User's will already know RND on a computer is generated by a pseudo random sequence. This is a mathematical expression which starting with a SEED will produce a new value and a new SEED.

e.g. RND = f(SEED)
 new SEED = f(RND)

therefore anyone with the correct function f will be able to predict the next number in the sequence. However if the correct function is not chosen then RND will not be apparently random. This is the case with the Cortex. C C Kuan tried to make the sequence random by using the TIC counter and the RANDOM command. This is where I think his/her (?) error occurred. RANDOM sets the first SEED in the sequence, but does not change the RND sequence after the first SEED - i.e. the sequence will be different each time the program is run but not subsequently and produces a cross latch since the RND function is not "Random enough"

The following functions produce very good pseudo random sequences for those interested.

a. $SEED = (1509 \times SEED + 41) \text{ mod } 2^{16}$

or

b. $SEED = (69069 \times SEED + 41) \text{ mod } 2^{32}$

a. will produce a new SEED between 0 and 65535

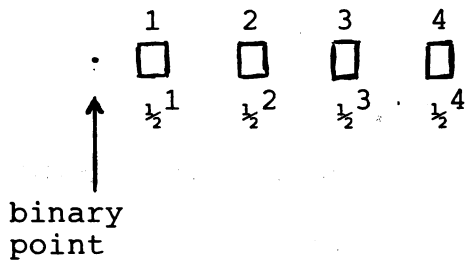
b. will produce a new SEED between 0 and $(2^{32} - 1)$

I hope they are of use.

Andy Kendall of Bristol's N=62.4

For those who don't understand what is going on here I think it is only fair to tell them.

A decimal fraction in binary must be the sum of powers of 2 between $1/2^1$ to $1/2^n$ where n is the number of binary digits right of the binary point. E.g. for a four bit binary fraction:-



A 1/2 would be represented by	.1000
1/4 " " " " " "	.0100
0.625 " " " " " "	.1010

0.4 cannot be exactly expressed as a binary fraction and is

$$\approx 0.25 + 0.125 + 0.0625 \quad (\text{in the above example})$$

ie. .0111

0.4 can never be reached but the more 1's added to the binary fraction the nearer it will get.

When printing the number to the display the computer prints the stored number not what it should be. Number formatting can be used to round the number off if desired.

Mr P N Paterson
Crewe

Mr Paterson has a tricky hardware problem. Maybe someone can suggest a solution.

My machine works satisfactorily except for one extremely annoying point and I was wondering whether anybody else had come across this. The fact is that I seem to have poor line sync for a period at the start of each TV field. Following the Frame Sync pulse. Although I have a locked picture (which is shifted slightly to the left) I have no colour. It appears that the inadequate line sync operates the TV colour killer circuit. I can achieve a colour picture by adjustment of the TV's line hold to the extreme of its range but the resulting picture is torn at the top. I have tried a number of different TV's and all are the same. Also, I have checked every component in the PAL encoder - even tried another VDP chip and voltages etc. are all correct. My oscilloscope shows that the line syncs for the first 40 or so lines are too small (measured at the modulator input)

I would be most grateful for any ideas concerning this as I feel I may have to resort to an RGB interface and monitor!!

Keep up the good work with the User Group.

- - - x x x - - -

M G Radford
Ashbourne
Derbyshire

Mr Radford has an interfacing problem, and would like some help in solving it.

I have recently experienced a problem with my Cortex for which I have so far been unable to find a solution.

I need to be able to send hexadecimal zero to a printer attached to the RS232 Interface or Centronics parallel interface. Whilst it is possible to send 01H to FFH by the means of -

```
PRINT "<01><02> .... <FF>"
```

it is not possible to send -

```
PRINT "<00>"
```

as this sends the hex data representing "<","0","0",">" which is not only incorrect data but also four characters instead of one.

The results of this also show on the monitor as <00>.

- - - x x x - - -

I M Austin
56 Harbury Road
Birmingham

The BASIC interpreter of my Cortex I functioned perfectly until recently, but now many of the words do not work properly.

To investigate this, I would like to make a comparison with the memory dump of another Cortex or visit another machine. (I live in Birmingham).

Can someone please help.

- - - x x x - - -

W D Eaves
Halkirk
Caithness

Mr Eaves has a problem with his screen display.

Using the TV, (I don't know whether or not a monitor would be affected) the screen is prone to jump occasionally, and becomes very unstable when using a black background. It appears to occur more in GRAPH Mode, but this may be just because I use black backgrounds more in GRAPH mode. I think this may be a hardware fault, but do not know where it could be - any ideas?

- - - x x x - - -

And finally in this section, a few words from your Editor.

Anyone who has purchased a copy of the '3D GRAPH' program from us, may have noticed that the option to type in your own function does not work. Mr A Lyall, the author of the program, has written to us to point out this fault. After studying the problem I have discovered that the following amendment should make the program work. Any further copies of this program sold will, of course, be the corrected version.

```
10 DIM LIN (10), FUN(9)
760 PRINT "PLEASE TYPE IN YOUR FUNCTION"
770 PRINT " E G 1/(COS[X]*SIN[Y]+1.1)"
780 INPUT $FUN[0]
790 $LIN [0]="810 DEF FNA = "+$FUN[0]
800 ENTER $LIN[0]
810 DEF FNA =
820 GOTO 470
```

- - - x x x - - -

CORRECTIONS TO NEWSLETTER II

Unfortunately we have discovered that there were a few mistakes in the 2nd Newsletter. Thank you to anyone who wrote in to point these out.

- A. In R.M Lee's letter on page 13, the memory address 18B2H should be 434H for 1200 baud. (Not 043H as shown).
- B. On page 16, the assembly line 6054H should read LDCR @ > 6071 (R9),8.
- C. In the assembly listing on page 21, the line 5FBAH should read A@>5FF4,*R10.
- D. On page 25 the following lines should be altered to read:
10000 FOR I = 6200H TO 62EEH STEP 2
10400 DATA 1728,-10240,-3807,-15279, etc
10130 DATA 577,240,-24510,1696,etc
10180 DATA 25112, etc

CORTEX SOFTWARE SCENE

The following programs are available from us, at £6.50 each plus V.A.T (see order form). They have all been written by Cortex Users. If you have any good programs, then why not send them in. We provide a full marketing service, and pay £1.50 royalties per copy sold.

- FROGGER Cortex version of favourite arcade game. Includes busy road, crocodiles, turtles etc.
- BURGLAR Guide your man around the screen, and collect the keys to open the safe. Avoid the animated obstacles, such as conveyors, and collapsing walkways. Can you complete all 16 screens each of which presents a new menace? (Similar to 'Manic Miner' or 'Jet Set Willy')
- HUNCHBACK Another arcade conversion. Guide Quasimodo along the top of the walls, jumping gaps avoiding arrows in order to ring the bells, and eventually rescue Esmerelda.
- G DESIGN Menu driven package to design your own shapes & characters. Draw on screen and then convert to data, or vice versa.
- WALL Bomb a wall, while your aircraft flies back & forth across the screen, losing height. Complete your task of demolishing the wall completely before you hit it or your time limit runs out, or you will be eliminated!
- CORTELLO Comprehensive adaptation of the classic Othello game. Play against the computer at one of two difficultly levels.
- 3D GRAPH 6 graphs to choose from, or the option to type in your own function.
- ARCHIE Another multilevel arcade game. Guide Archie (a prospective alcoholic) along ledges, and up ladders to collect wine, whilst avoiding the snakes.
- INVADERS &
ASTEROIDS Double package of two famous arcade originals Fast machine code 'shoot em up' action.

The next two packages are both formatted to output directly to a printer via the RS232C port, but could no doubt be adapted.

WINE & FORM 1 Home wine making utilities. Calculate acid mixtures etc. Print out a standard form on which to write your favourite recipes

SHARES & HISTOGRAM Calculate profits/losses on shares, and print out histograms to show you where your money went.

MAZE 3D maze adventure, in which you have to find a food chest. 5 levels of difficulty, and very good graphics make this an enjoyable game.

NIGHT ATTACK New York is under attack by a skyscraper destroying spaceship. Aim the Statue of Liberty's torch to shoot back, and save the city.

ORDER FORM

Please use this form to order your programs.

PROGRAM TITLE	PRICE EACH	QUANTITY	TOTAL
FROGGER	£6.50		
BURGLAR	£6.50		
HUNCHBACK	£6.50		
MUNCHER	£6.50		
G-DESIGN	£6.50		
WALL	£6.50		
CORTELLO	£6.50		
3-D GRAPH	£6.50		
ARCHIE	£6.50		
INVADERS & ASTEROIDS	£6.50		
MOONBASE II	£6.50		
WINE & FORM1	£6.50		
SHARES & HISTOSHARE	£6.50		
MAZE	£6.50		
NIGHT ATTACK	£6.50		
SUB TOTAL			
VAT @ 15%			
TOTAL			

Please forward cheque to Powertran Cybernetics Limited,
West Portway Industrial Estate, Andover, Hants.



53 BROUGHTON ROAD,
CROFT,
LEICESTER
LE9 6EB

(PROPRIETORS: P.A. ROE AND R. ROE)

TEL: SUTTON ELMS (0455) 282679

JOSY-1 Joystick and Sound Module

FEATURES

- * Joystick Port (Kempston, Atari etc. compatible)
- * Sound Generator with 3 tones plus noise source
- * 8 inputs and 8 outputs with 25 way connector option
- * Double sided PCB, 100mm x 75mm, fits inside Cortex case
- * Machine coded Joystick and Sound routines

DESCRIPTION

The MarkroSoft JOSY-1 module is designed to add a permanent joystick and sound effects capability to the Cortex computer.

It comprises a small PCB fitted with a 9 way D type plug, for the joystick and 6 IC's including the Texas Instruments SN76489 sound generator chip. An optional 25 way D plug can be added to give a full 8 bit plus strobe I/O capability. The sound output is used with an external audio system, or TV sound may be used if the TV modulator on the Cortex PCB is changed for a sound and vision version.

Two software routines are provided with the module, written in machine code and are callable from Basic. The joystick routine returns the same code as the Ascii code for the keypad arrows. A Basic call with tone number, volume and period is all that is needed to drive the sound chip. Three simultaneous tones and noise can be produced.

JOSY-1 is available as a blank PCB, a complete kit of parts or a fully assembled and tested module. An 18 page User Guide is provided together with a tape or disc containing the machine code routines, a Basic demonstration of sound effects and a FREE copy of the Spacebugs II game.

Constructors should be aware that this module is designed to fit permanently inside the Cortex case and some soldering is required on the Cortex PCB to connect it to the JOSY-1 module.

Ebus interface is not required for this module.

PRICES

PCB plus User Guide and Demo tape	£10-95
Complete kit of parts for module with 9w plug	£19-95
Assembled and tested module with 9w plug	£24-95
Optional 25 way D type PCB mounting connector	£3-25
Software on CDOS compatible disc (instead of tape)	£1-50

post and packing 75 p

Please allow 21 days for delivery

MARKROSOFT SOFTWARE FOR THE CORTEX COMPUTER SYSTEM

The following programmes are now available for the Cortex. The JOSY-1 joystick and sound generator module may be used with any programme marked thus [J].

SPACEBUGS II- An updated version of one of the first games available for the Cortex now with joystick and sound effects added using the JOSY-1 module or keypad. [J]

CAPMAN - A variation on the Pacman theme. Written in Basic for keypad or joystick. It is a 7K programme with 10 screens and plenty of colourful action. [J]

SHAPES 'n' SPRITES - A graphics programme to help in defining shapes and sprites. Four planes of 1 to 4 shapes can be created, edited and overlaid on top of each other, in any colour, to build up a multi-coloured sprite. The shape data is displayed in hexadecimal format and may be saved on disc, paper or cassette.

SOUTH SEA ADVENTURE - A classical style adventure game. Direct your 'Man Thursday' to collect the treasure and escape the attention of the pirates.

COMMTEX - A communications package written for the Cortex to allow access to bulletin boards and Microlink. Any modem connected to the RS232 port and operating at the standard 300/300, 75/1200 and 1200/1200 baud rates is supported by the Commtex software. Utilities are provided for off line preparation and storage of messages. The receive buffer may be examined, edited, printed or dumped onto disc. The package is written in Basic with machine code I/O routines. A user guide and programme listing is supplied to allow extra features to be added if required.

GRAPHICS - Use this programme with the JOSY-1 joystick to create colourful scenes for games or just for the fun of it. The pictures are 'painted' using a joystick or the keypad and helped by functions such as Fill and Circle. The scenes may be stored on disc in named files and reloaded as required to edit or use in applications. A routine is given to add to user programmes which will put a stored picture on the screen using a Basic Call. This programme together with Shapes 'n' Sprites makes a complete graphics 'toolkit' for most applications. [J]

RETURN TO THE DEJI - No prizes for guessing the origin of this one. Designed around the Starwars theme it has three interconnected battle scenes to give your joystick some exercise. [J]

The programmes are supplied on 5 1/4" single or double density CDOS compatible discs at £5-95 for one programme and £3-95 for each additional programme. Post and packing is included in the price.

SL85-3008

Tim Gray
1 Larkspur Drive
Featherstone
Wolverhampton
West Midlands
WV10 7TN

Mr Gray has now finished the design work on his external video board for the Cortex. It fully synchronises the VDP chip to an external video feed from camera or video tape and does the switching to provide a combined video output.

The board is available from him as an unpopulated PCB with circuits, component lists and construction notes for £30.00 or fully built and tested with fitting instructions for £100.00.

He has also written a menu driven graphics drawing package with lots of plotting commands including circle, polygon and a machine code fill command as well as sprite speed control shape design large character text and a fast screen dump to disk. Screens saved to disk can then be recalled for use in other programmes, games etc.

It is available for CDOS version 1.10. All enquiries to the above address.

FLOPPY DISC ELECTRONICS PARTS LIST

<u>QUANTITY</u>	<u>PARTS</u>
1	PCB as per circuit diagram
1	TMS2797NL
1	TMS9911
1	7406
3	7407
1	74LS02
1	74LS04
1	74LS08
1	74LS32
1	74LS38
1	74LS74
1	74LS138
1	74LS139
1	74LS163
2	74LS244
1	74LS259
1	74LS297
1	LM339
8	14 pin IC socket
8	16 pin IC socket
2	20 pin IC socket
3	40 pin IC socket
8	0.1 μ f cap 250v
1	0.22 μ cap 100v
10	1/4w resistors
1	150R 6 pin SIL
1	1N4148 diode
1	33p ceramic