

MAY '86

7



Stop Press
See inside for details of
OPEN EVENING

UK EINSTEIN USER GROUP
NEWSLETTER

38' RAM

Issue Number Seven

May 1986

EDITORIAL

Surprise surprise here we are again AND on time, (it won't last).

The reason we are on time this month is because we have some important news to tell you. On Friday the 6th June we are holding an open meeting at the Village Centre at OXSHOTT (no I hadn't heard of it either) near LEATHERHEAD Surrey. This meeting is primarily for those members who live in the London and Surrey areas but a map has been included so that any member can find us if they want to.

The idea behind this is so that we can meet some of you in person, and as a forerunner to a national meeting later in the year. Screens will also be there showing their new printer amongst other things. We will be demonstrating the Speculator and various other software. We will also be giving away free software so bring a blank formatted disk along.

Now on with the Newsletter.

Later in this issue is a review on Screen Plus ,to coincide with this we have another competition (this one is tricky ,I don't even know the answer),and the prize is our review copy of Screen Plus.

All you have to do is decode the message below ,the clue might help, send the answer and your method of finding the answer to me at the following address, to arrive by the 20th of June.

Keith Stokes
Hillcroft
Codmore Hill
Pulborough
West Sussex
RH20 1BQ

83 70 70 79 27 61 72 69 71 59
73 61 71 59 64 62 15 64 50 61
64 51 59 45 58 06 40 44 53 06
09 42 10 41 06 -04 47 41 -07 43
38 34 43 25 -13 27 37 -02

clue:

SOLVE THIS COMPETITION WITH CHARACTER

Two reviews for the price of one

During the week I recieved a 'phone call asking me if I would like to spend the weekend in the Bucks country side (my thanks to Dave and Linda for putting up with me)and have a look at something called a Silicon Disc.

For those who are not sure what this is I will try to explain.

A silicon disc is a small piece of hardware consisting of a large amount of RAM (256k) which plugs into the computer. Through the software also contained within the unit it appears to the computer and acts as just another disc drive. The only way to access it is as a disc drive it is NOT a RAM UPGRADE .

The unit plugs into the pipe at the back of and is powered by the computer ,but unlike the 80 column card you can not plug anything else into

it (I have heard that there is an internal version which would get around this problem).Having plugged it in you now load the two short programs which come with it.The first being SD (Silicon Disc Driver) which interfaces between the standard operating system and the silicon disc.The second is SF (silicon Disc Format)which formats the unit just as you do with a new disc before you use it.

You can now use the unit just like a disc drive ,Dir, Copy, Erase, Save,Load etc,but the most important thing is it is so much quicker as there is not a mechanical head searching a rotating disc.

Two important things to remember is to copy any files on the silicon disc to a standard drive before switching off as all is lost when there is no power.The second is to reload SD after resetting or pressing CTL/BREAK so that you can access the drive and all files will still be present.

Now for some figures.

Using two standard 3"drives with dBase II on 0 and a 60k 772 record file on 1 it took 5 minutes 15 sec to list all the files in numerical order.

Using drive 0 as above but with the data file on the Silicon disc it took 2 min 31 sec to list.

By using the Silicon disc on its own with all the files copied into it the time was further reduced to 1 min 41 sec.

The price for this speedy piece of hardware is about £89.00 no hard figures yet but well worth the money if you use large data files or want to speed up your disk access when using Wordstar and the like.

While I was with Dave I discovered that his son, Carl , had a Spectrum and so set about testing the Speculator (it's a hard life being a reporter).We found that Carl had five of the twenty programs which are supposed to run on the Einstein,most of these were "BACKUPS" of the originals .

The titles we tried are: ATIC ATAC
THE HOBBIT
ARCADIA
STARION
HUNCH BACK

All but one of these loaded and ran perfectly and, according to Carl ran just as they do on the Spectrum.The one program we could not load was Starion and this was the only original tape we had !!.We tried several volume settings but to no avail although it loaded and ran on the Spectrum.

If you like playing games or have a Spectrum and want to play your games on the Einstein then this is for you.

Screen Plus

By

Martin Page

When asked to do this report I grasped the opportunity with both hands, having seen the demonstration at the Business Computing Center. I was very impressed at the way the 'colour bleed' problem had been overcome, this as it turns out is not the case. SCREEN PLUS makes the best job of the limitations imposed by the graphic chip by letting you see, pixel by pixel, where the problems occur.

The disc contained nine demo screen files & the SCREEN program on one side, GRAPHIC & LOADER.OBJ on the other, more about that later. The 10 page manual, though brief, was simplistically informative but did tend to under state the programs capabilities and limits.

On booting up you start in DOS and have to type in SCREEN. You are then presented with the menu and an annoying whistle (You can turn your sound down).

Your key entry is via the function keys F0 - F4 (I would have liked to have seen F5 - EXIT). F0 - Disc commands, allows you to load save and peek the screens on file. On peeking the files they are very impressive indeed, especially the Clingon attack on the starship, and fast. The whole screen is filled in the blinking of an eye.

You have to load the file before you can 'get at it' and here you have

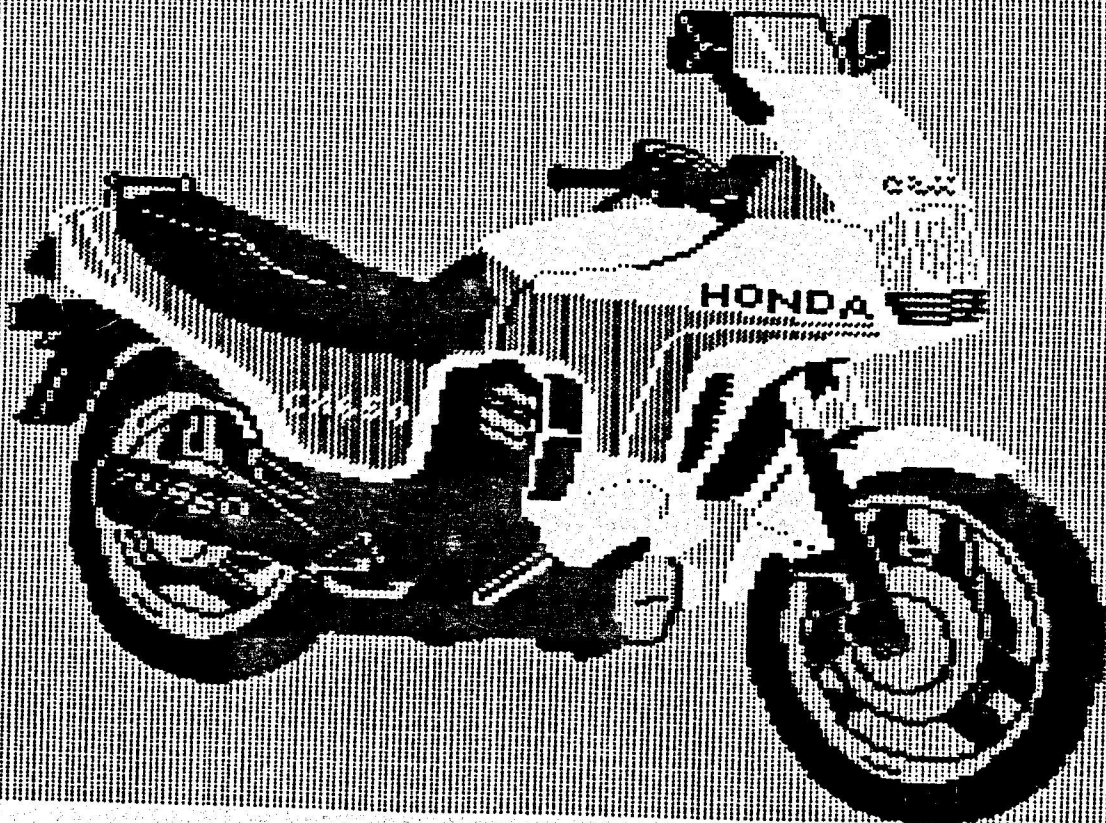
to be quick fingered to only enter what you want as the machine code is very fast. Once loaded the picture comes up screen MAG1 and from here on in you require the use of the joystick. Once you've mastered the control commands your off and running, then you start to realise just what is entailed in producing such masterpieces.

It was at this point my enthusiasm for it started to wane. To edit a picture was fairly straight forward but to produce one from scratch was a daunting prospect, having to colour EVERY pixel.....If only you could use DRAW, ELLIPSE, FILL even PICPEN to give you a start. A telephone call to Sintaxsoft told me "YOU CAN". The instruction manual tells you how to use the pictures from BASIC but doesn't mention that you can go the other way. This, then, made screenplus a very powerfull utility program.

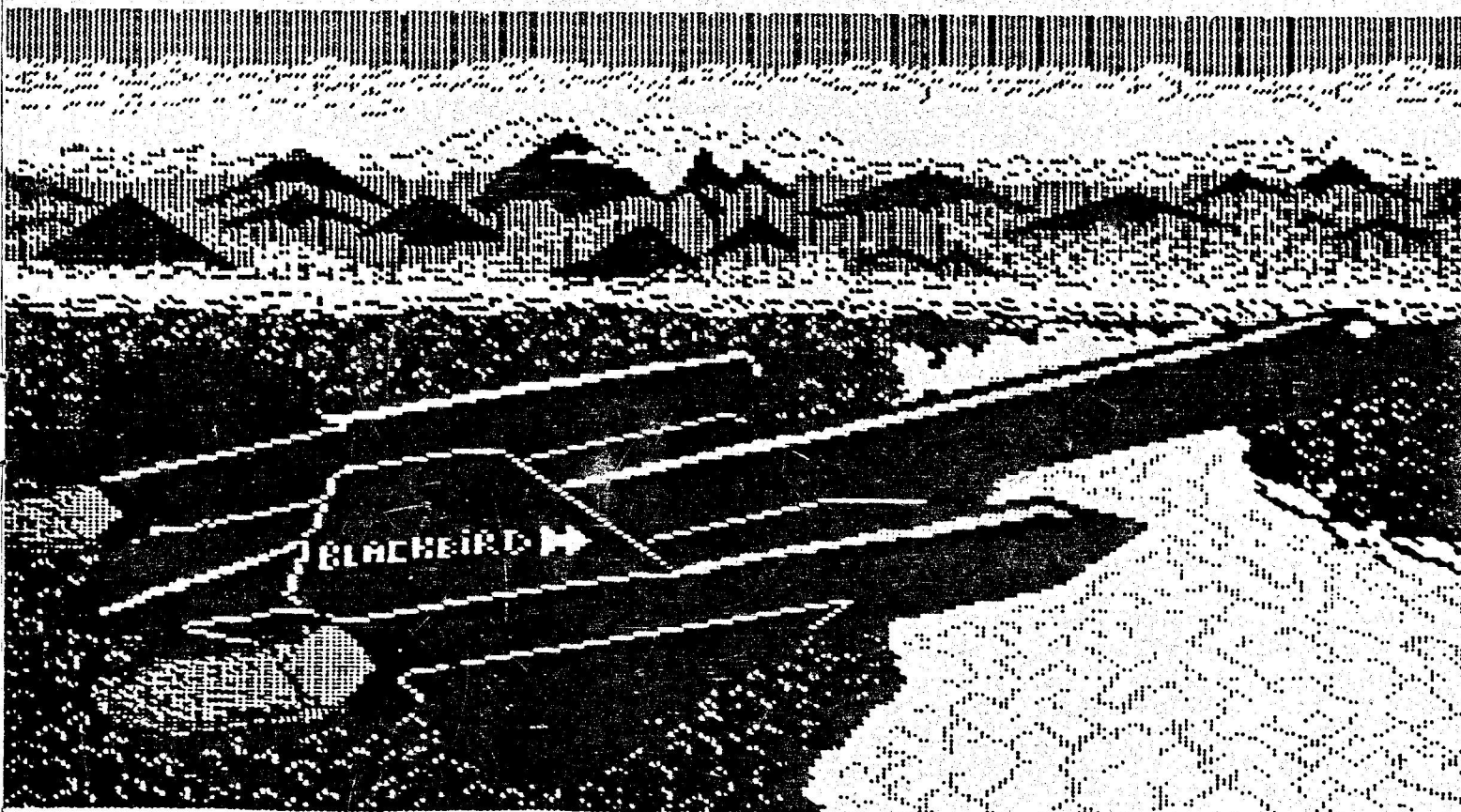
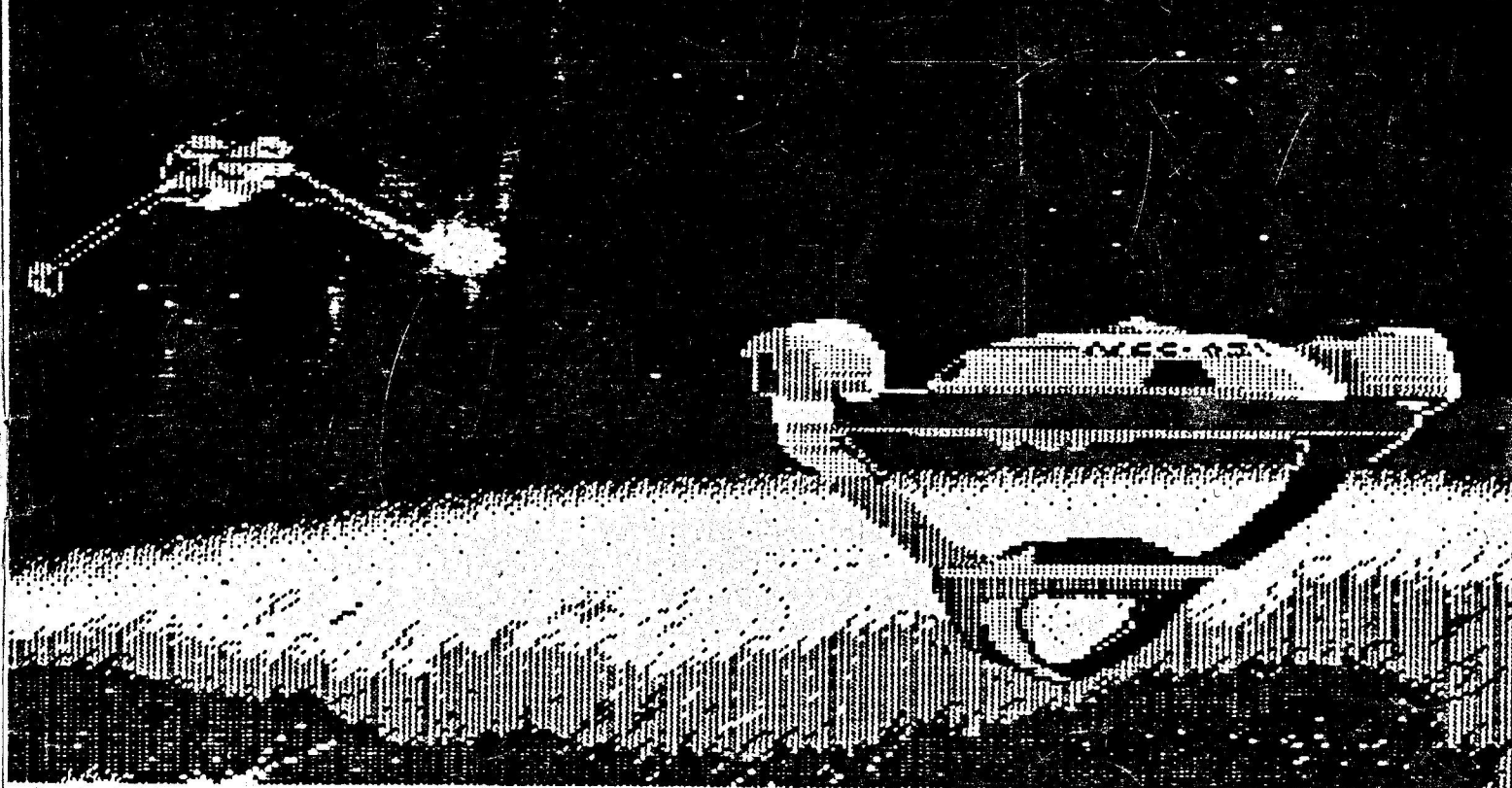
When you've finished your masterpiece, and saved it to disc, you can then change the each of the colours in turn to put a different slant to it, until you are totally satisfied with the effect.

The F4 PRINT DISPLAY option, I must say, gets nothing but praise. this is most impressive and alone is worth the price of the package. It gives a sideways on printout of approx 9" x 5". If you think that is good, then the fact that it copes with the half tones and midtones corresponding to the respective colour is even better. There is, however, a sting in the tail. Unfortunately this doesn't work on every printer, only EPSON compatible types, but is so good you may be tempted to change your printer!

The GRAPHIC program is an added bonus to the package. I personally found it somewhat tedious to operate and restrictive to use, in as much as any programs written using sprites or characters from it had to be loaded using the LOADER.OBJ program. If you are writing programs solely for your own use then the rotation and mirroring are very usefull, if not then a simple sprite program, such as the one in the Einstein User, could be more flexible.



U.S.S. GRISSEM under attack from a Klingon 'Bird of Prey' Scoutship



SCIENCE and SORCERY - THE ADVENTURER'S COLUMN
(DAMMIT I'VE DIED AGAIN!!)

QUEST/CASTLE QUEST

=====

The games arrive on one disc with no documentation, just instructions to load and run the games (although as all you have to do is load BASIC and run the appropriate game I don't think these are really vital). Running Castle Quest gives you a screen asking you if you want instructions. These give you a description of the aim of the game. The quest you find yourself on is to locate and escape the castle with four treasures : Mail of Mithril, the Black Crystal, the Elf Stone and the Elven Sword. There are a number of problems to solve at each level of the game, and to pass on to the next level you must find the necessary password hidden on the present level. As with many other adventure games, you find various items on your travels through the castle, each of which you must use at some point in the game to solve the rather simple problems.

Quest is very similar game to Castle Quest. The style of the two games is obviously the same, both being written by the same person. To be perfectly honest, after completing Castle Quest I really did not feel like playing Quest. I played through some of the game to get enough of a feel for it to write this review, but I soon became bored with it, and gave it up. I'm afraid that these games are not really much of a challenge to any but the most inexperienced adventurers. This is partly due to the limitations of BASIC as a language for adventure games, and partly due to the fact that most of the problems presented to you are reasonably easy to solve. These, combined with the limited location descriptions and the fact that the game can only understand two word input result in a rather unexciting and simple game. It shouldn't take you too long to work your way through your Quest, to find the four sectors of the Eye of Morpheus and use them to extinguish the Eternal Fires of Hell in order to bridge the crevasse and claim the treasure. It may be that this game improves substantially beyond the point that I finished it, but I would be very surprised if that proves to be the case.

If you are looking for a couple of games to play as an introduction to adventures, you could try these, but don't judge all adventure games by these, they are very basic.

PROGRAM BUILDER

In the February Edition of the Newsletter I described my trials and tribulations in trying to get the above program running under XBAS or EBCBASIC and my subsequent promise to try to get the program running under Xitan's XBASIC.

I then went on to list the troubles I had found with the bugs in the XBASIC. After lots of phone calls and about 6 letters I finally got action from Tatung who updated the disc so that most, but not all of the bugs disappeared. As to the remaining bugs Tatung seem to have opened a new department especially for ignoring customers complaints, even the Managing Director ignores letters addressed to him!

In spite of the bugs I finally got the program running in it's basic form and I submitted it to our worthy editor for inclusion in the next newsletter. I just missed out on the March issue so it was scheduled for the April issue.

Meanwhile I still wanted to use XBAS so I wrote to Crystal Research stating my problems. To my surprise Crystal Research actually answer letters and give helpful advice. I have now got the program running under XBAS with lots of enhancements that I didn't have in the XBASIC version.

During a long discussion with our worthy editor it was agreed that he would not publish the XBASIC version but would await the XBAS version. I have since bought Crystal Research's System 5 with an even better basic and a disc load time four times faster than the old DOS.

However for the benefit of those with the old DOS I am sending the

copy on the disc to the editor with the old DOS, although it works equally well with either but those who get a copy of the month's programs on disc from UKEUG may find difficulty in loading if they did not have the new DOS.

So much for the preamble, here is the program. It runs in conjunction with program FRAME given in the December newsletter, but with no processes and with slight changes to the actual program FRAME.

Program BUILD reads a program text from a source file and allocates line numbers within the FRAME structure, prepares a menu of the processes contained in the source text places all sub-routines and DEF statements at the end of FRAME and DATA statements at the end of each separate process.

To use the program all the user has to do is to write the source text using an editor, I use Wordstar but it is important when using Wordstar that the SOURCE text file is opened as a Non-document file by selecting "N". Failure to select "N" will mean that in the SOURCE file Page Markers will appear, and, for some reason which I haven't fathomed, the Page Marker causes a DIM statement to appear in the final compiled program. Those of you who have ED80 will find that this works OK, and, I suspect so will XED which is supplied with System 5. However WDPRO will not work.

There are a number of rules to be followed in preparing source text. Each process, including an initialisation process must start off with a Title, followed by a Purpose, the Initial letter of the title is used as the key letter in the menu and must be unique for each process used and must not be the same as the command letters used in FRAME, i.e. TSX and ?, I suggest you use A to N. The Purpose is not critical as long as it conveys to the user what the process is for.

After the Title and Purpose there follows either the initialising statements or the algorithm of the process. At the end of the initialising statements and at the end of all processes the word END must appear, followed if there are further processes by a new Title and Purpose. The final process must have END followed by the word FINISH to indicate to program BUILD that there are no further processes to follow.

The source program text follows standard BASIC statements but with certain limitations and requirements. Firstly complex statement lines using IF THEN ELSE must be avoided if the THEN or ELSE statement refers to a line number as the BUILD program, as it stands at the moment, hasn't the facility of allocating line numbers in these complex statements.

In spite of the limitations mentioned above program BUILD will allocate line numbers to GOTO statements provided that the GOTO statement appears at the beginning of the program line. Since the person writing the source program will not know what the actual line number of the line that the GOTO statement wants to go to it is necessary to use a form of relative addressing, and the statement GOTO is abbreviated to GTO and the line number is given as a positive or negative number indicating the number of lines forward or back you wish to Go To e.g. GTO-13 will GOTO 13 lines back from the GOTO statement. One point to remember at this point is, if a subroutine appears in the text between the GOTO statement and the line you wish to Go To all lines in the subroutine should not be included in the calculations as the subroutine will be moved, by program BUILD, to the end of program FRAME.

Subroutines are somewhat easier, each subroutine is given a unique identification letter following the mandatory word SUB, e.g. SUBA, SUBB etc., then follows the text of the actual subroutine ending with the word RTN. Calls to the subroutine are made by the word GOS, e.g. GOSA, GOSB etc.

Data statements are local to each process and are allocated the line numbers xx900 in each process. The data statements in the source program can, for convenience be placed at the beginning of each process, with one proviso, if there are several lines of data the basic statement RESTORE should appear immediately after the first line of data, this ensures that the RESTORE always returns to the first data statement.

DEF statements follow the standard XBAS format, but as mentioned above the BUILD program moves these to the end of the FRAME program, in fact to lines 17000 et seq. DEF statements are therefore Global to all processes

and care should be given to allocating discrete identifiers to each DEF statement.

In order to illustrate some of the above facilities I am given a demonstration SOURCE.XBS based on Mr Raper's original Mortgage Calculation program. As you will see this does not include GOSUBs, DEFs or DATA statements, however these are included in the demonstration SOURCE text included with the listing of BUILD.XBS and FRAME.XBS. It should also be noted that program BUILD calls up the source program as SOURCE.XBS, thus if you are developing several programs on the same disc you will have to call the source programs SOURCE1, SOURCE2 etc and make the necessary changes in line 70 of program BUILD and possibly in lines 790 and 880 to differentiate between the NEWPROG.ASC intermediate programs which are automatically written onto the disc.

NOTE. these intermediate programs NEWPROG.ASC must always be saved as ASCII files. This is to get round the problem of not having a true MERGE in XBAS, however by LOADING an ASCII file on top of an existing XBS file the ASCII file will merge on top on the XBS file. (my thanks to Crystal Research for the useful tip).

A further point to notice on the demonstration SOURCE text below is that remarks following the brackets after the line of text are for the benefit of the reader and play no part in the program example and should not be included if you are using the demonstration as an actual example.

MORTGAGE	<Title>
TO CALCULATE MORTGAGES	<Purpose>
END	<End of Initialising>
PRINCIPAL	<Title of first process>
TO CALCULATE PRINCIPAL	<Purpose of first process>
INPUT "INTEREST RATE ";I:I=I/100	{
INPUT "TERM IN YEARS, MONTHS ";Y,M:TERM=Y+M/12	{
INPUT "MONTHLY REPAYMENTS ";R:R=R*12	{ Algorithm
P=R/(I+((1+I)^TERM-1))	{
FMT 6,2	{
PRINT "PRINCIPAL AMOUNT ";P	{
END	{ End of Process
INTEREST	{ Title
TO CALCULATE INTEREST	{ Purpose
INPUT "PRINCIPAL ";P	{
INPUT "TERM IN YEARS, MONTHS ";Y,M:TERM=Y+M/12	{ Algorithm
INPUT "MONTHLY REPAYMENT ";R:R=R*12	{
TEMP=0.1	{
REPEAT	{
I=TEMP	{
TEMP=R/P-(I/((1+I)^TERM-1))	{
UNTIL ABS(TEMP-I)<.000001	{
FMT 2,3	{
PRINT "INTEREST RATE ";(I*100)	{
END	{ End of Process
DURATION	{ Title
TO CALCULATE DURATION	{ Purpose
INPUT "PRINCIPAL ";P	{
INPUT "INTEREST RATE ";I:I=I/100	{
INPUT "MONTHLY REPAYMENT ";R:R=R*12	{ Algorithm
TERM=LOG(P*I/(R-P*I)+1)/LOG(1+I)+1/24	{
Y=INT(TERM):M=INT((TERM-INT(TERM))*12	{
FMT 2,0	{
PRINT "TERM IS "Y" YEARS, "M" MONTHS"	{
END	{ End of Process
REPAYMENT	{ Title
TO CALCULATE REPAYMENTS	{ Purpose
INPUT "PRINCIPAL ";P	{
INPUT "INTEREST RATE ";I:I=I/100	{
INPUT "TERM IN YEARS, MONTHS ";Y,M:TERM=Y+M/12	{ Algorithm


```
R=P*(I+(((1+I)^TERM-1)))/12
FMT 4,2
PRINT "MONTHLY PAYMENT IS ";R
END
FINISH
```

C
C
C
C End of Process
C Finish

To run the program assuming that you have BUILD.XBS, FRAME.XBS and SOURCE.XBS on the disc LOAD and RUN BUILD.XBS. After a short while the following will appear on the screen:-

```
Your new program is almost ready.
After 'FRAME.XBS' has loaded
and the 'READY' appears
LOAD 'NEWPROG.ASC'
and SAVE as 'MYPROG.XBS'
```

I have included several REM statements in BUILD to explain, as far as possible, how the program functions. One Line requires explanation however, Line 90 F(0)=0:F\$(0)="REM" is put in to overwrite garbage which seems to appear in the first line of SOURCE.XBS after it has been READ. I can offer no explanation why it happens or why Line 90 cures it, and I have written to Crystal Research about it, try running BUILD without Line 90.

```
10 REM THIS IS 'BUILD'
20 REM FOR EINSTEIN USE WITH XBAS
30 REM (C) COPYRIGHT F.R.PETTIT MA(OXON) AND V.J.DAY
40 DIM F(200), F$(200):REM for line Nos and line text
50 SEP 0
60 N=1:M=1000:L=0:L1=-10:L2=0:L3=14990:L$="GOSUB":L1$="GOTO":B$="TSX?"
70 OPEN "0:SOURCE.XBS",FD$:REM Provides access to source text
80 INPUT# FD$:A$:REM Lines 80 to 240 handle the initialisation module
90 F(0)=0:F$(0)="REM"
100 F(N)=2
110 F$(N)="PRINT "+CHR$(34)+A$+CHR$(34)
120 N=N+1
130 INPUT# FD$:A$
140 F(N)=4
150 F$(N)="REM "+A$:
160 N=N+1
170 FOR N=N TO 30:REM could be up to about 45
180 INPUT# FD$:A$
190 IF A$="END" GOTO 250
200 F(N)=2*N
210 F$(N)=A$:REM line text
220 NEXT N
230 PRINT "INITIAL MODULE HAS TOO MANY LINES"
240 STOP
250 FOR N=N TO 200:REM Lines 250 to 740 handle up to 14 modules with
    a total of 150 lines
260 INPUT# FD$:A$
270 IF L=0 THEN C$=LEFT$(A$,1)
280 IF LEFT$(A$,3)="SUB" THEN 290ELSE 360
290 R$=MID$(A$,4,1)
300 R=ASC(R$)
310 R=R-64
320 L3=L3+10:GOSUB 1000
330 F(N)=L3
340 F$(N)="RETURN"
350 GOTO 740
360 IF LEFT$(A$,3)="GOS" THEN 370ELSE 440
370 R$=MID$(A$,4,1)
380 R=ASC(R$)
390 R=R-64
400 F(N)=M+L
```

```

410 F$(N)=L$+STR$(E(R))
420 L=L+10
430 GOTO 740
440 IF LEFT$(A$,3)="GTO" THEN 450ELSE 520
450 R1$=RIGHT$(A$,3)
460 R1=VAL(R1$)
470 R1=R1*10+M+L
480 F(N)=M+L
490 F$(N)=L1$+STR$(R1)
500 L=L+10
510 GOTO 740
520 IF LEFT$(A$,3)="DEF" THEN 530ELSE 560
530 L1=L1+10
540 F(N)=16000+L1
550 F$(N)=A$:GOTO 740
560 IF LEFT$(A$,3)="DAT" THEN 570ELSE 600
570 L2=L2+10
580 F(N)=M+900+L2:F=F(N)
590 F$(N)=A$:GOTO 740
600 IF A$="RESTORE" THEN 610ELSE 620
610 A$=A$+STR$(F)
620 IF A$="END" THEN 630ELSE 640
630 M=M+1000:L=0:L2=0:N=N-1:GOTO 740
640 IF A$="FINISH" THEN 770ELSE 650
650 F(N)=M+L:L=L+10
660 IF L<=20 THEN F$(N)="PRINT "+CHR$(34)+A$+CHR$(34):
    ELSE F$(N)=A$:REM Normal line
670 IF L=10 THEN N=N+1:ELSE 700:REM Lines 650 to 670 provide module
    map at lines 18000 et seq
680 F(N)=18000+M/100
690 F$(N)="REM "+A$+" IS AT"+STR$(M)
700 IF L=20 THEN N=N+1:ELSE 740: REM lines 700 to 730 make up the
    'instructions' lines
710 F(N)=500+M/100
720 F$(N)="PRINT "+CHR$(34)+" "+C$+" "+A$+CHR$(34)
730 B$=B$+C$
740 NEXT N
750 PRINT "insufficient space"
760 STOP
770 SEP 44
780 K=N-1:CLOSE FD$:REM Lines 780 to 840 write the new program file
790 CREATE "0:NEWPROG.ASC",FD$,0
800 FOR N=0 TO K
810 PRINT# FD$;F(N);F$(N)
820 NEXT N
830 PRINT# FD$;110;"L$ = ";CHR$(34);B$;CHR$(34)
840 CLOSE FD$
850 PRINT "Your new program is almost ready":PRINT:PRINT
860 PRINT "After 'FRAME.XBS' has loaded"
870 PRINT "and the 'Ready' appears"
880 PRINT "LOAD 'NEWPROG.ASC'"
890 PRINT "and SAVE as 'MYPROG.XBS'"
900 LOAD "FRAME.XBS"
1000 F(N)=L3:E(R)=L3
1010 F$(N)="REM"
1020 INPUT# FD$;A$
1030 N=N+1:L3=L3+10
1040 IF A$="RTN" THEN 1050:ELSE 1060
1050 RETURN
1060 F(N)=L3
1070 F$(N)=A$
1080 GOTO 1020

```


(using the BBCBASIC assembler)

Well how did it go? Within hours of the newsletter going out I was rapped on the knuckles. "Your article is NOT on Machine Code programming, it is on Assembler" I was forcefully told. Well what do you expect from a beginner I replied! I suppose I should explain the difference since I now know. Machine Code programming is putting all the data into the machine in MOS and working out everything by yourself like what address to jump to and where each call goes to and all sorts of complicated things like that. Assembler is putting the Mnemonics into an assembler program and letting it do all the hard work of calculations, which a computer is so good at.

Well now that's out of the way lets get on with the real stuff. Apologies:-Page 4 Line 20 should have read DEFB 158

Page 5 line 7 should have had a listing after it but if the BASIC was typed in correctly then the listing would have been produced by the program.

On to this months article. I use last months routine to put messages to the screen as a subroutine in this months program. The aim this month is to collect characters from the keyboard and respond to them. Other things get thrown in as well and hopefully as they are explained a better understanding of the processes will be gained. I know I have learnt a lot getting this one working.

The main core of the listing for last month was initially designed to cope with several months tutorials but, being a learner myself, I was not aware of how much space a small program would actually take up.

The new CORE listing is as follows:-

```

10 HIMEM=32768
20 PROC_ass(0)
30 PROC_ass(3)
40 PRINT"PRESS SPACE BAR TO CONTINUE"
50 key=GET:IF key<>32 THEN GOTO 50
60 CALL 32768:STOP
100 DEFPROC_ass(opt)
110 P%=32768
120 REM CONSTANTS ENTERED HERE
300 [OPT opt
310 ;MACHINE CODE LINES INSERTED HERE
10000 ]
10010 ENDPROC
    
```

This should allow much more room for programs within the core of our listing. The way to use it is type in the above and then save it. Then for each months programs load it back in and enter the new lines as printed in the NEWSLETTER.

The reason for a gap after line 120 is because I did not realise that constants had to be entered outside the main MC listing. (Thats one thing I have learnt)

THIS MONTHS ROUTINE

This months program offers 4 options, Change the text foreground colour, Change the text background colour, Change the backdrop colour or return to BASIC.

The options are selected by pressing the keys from 1 to 4. This bit is easy, if the key pressed is CHR\$(49) (the 1 key) then call routine 1, if the key pressed is CHR\$(50) (key 2) then call routine 2, if the key pressed is CHR\$(51) (key 3) then call routine 3, if the key pressed is CHR\$(52) (key 4) then jump to routine 4. On return from the routine (1,2 or 3) it will go back to the menu. 4 will return to BASIC. If any other key is pressed it will also go back to the menu.

Now a description. TCOL is a constant, it is the location where the current text foreground and background colours are stored. It

stores both of them in the form foreground * 16 + background. i.e. if the foreground colour is white and the background colour is dark red then the number stored in the LOCATION TCOL will be $16 \times 15 + 6 = 246$.

Moving down to the program itself. The first thing to notice is the LABELS. Labels start with a period '.' and are used a bit like line numbers in BASIC. If you want to jump to a particular part of the program then give it a LABEL and when you want to go there just put in JUMP LABEL. The JUMP acts like GOTO in BASIC. You can also CALL LABEL, this acts like GOSUB in BASIC. The period is only needed at the point where you want it to goto, you don't need it in a jump or call statement.

The first occurrence of a call in the program is at line 360 where we have a CALL PRM, there is no period on this label because it is not the actual location of the routine. In line 540 we have .PRM LD B,(HL), here the period indicates that this is where the location of PRM and whenever the assembler comes across PRM without a period it should put in the location of the .PRM. Clever things computers. The next new item is in line 450 and is the CP 49. What it is doing is comparing the number in the 'A' register with the value following (in this case 49) and setting flags on the result. In comparing it subtracts what is in the 'A' register from the value given and if the result is zero it sets the Zero flag. If the answer is less than zero it sets the carry flag. (Remember in school when you were subtracting a big number from a smaller number you had to CARRY 1). It also sets other flags depending on other things but I haven't found out what they are yet. Getting back to line 450, and comparing the value in the 'A' register with 49. 49 is the CHR\$ value of 1 (one), so if the character value in 'A' is 1 then the zero flag will be set. Line 460 then says CALL the routine IF the Zero flag is set so if any other key than 1 was pressed it won't do this one. It will do the same check for 2 and three. The same check is done for 4 but this time it will JUMP to the routine if the Zero flag is set. This is because we want to use the RET instruction to return to basic and for the others we wanted to 'return' to our own menu. If none of the tests pass then the key pressed was not one that was wanted so it is ignored and the menu is re-done. Lines 540 to 620 are last month's routine used as a subroutine. Line 730 Jumps back to the beginning of this subroutine if the carry flag is set. This is determined by the subroutine 'valid' more of which later. Line 790 loads the 'A' register with the value stored at location TCOL. Line 800 does the same as the BASIC 'MOD' keyword in the form 'value in the 'A' register MOD 16'. (one more than the number given) (I'm not sure why this is but it works) Line 820 loads the location TCOL with the value in the 'A' register. (I will shorten this to 'A' in future) Line 970 is the same as the BASIC ' $16 \times (\text{INT}('A'/16))$ ' and will be the subject of an article later but, suffice it to say at the moment, it works.

Now we come to lines 1120 to 1150. The video processor is accessed via PORTS. The Z80 instruction to put data to a PORT is OUT (port number), value. To change the backdrop in M/C one of the registers in the Video Processor needs to be changed to the new backdrop colour. First we have to tell it what the new value is, line 1120 (the new value is in 'A') then we have to tell it which register is to take on the new value. In this case it is register 7 that we want to alter. To do this we have to add 128 to the register number and send it out along the port. The reason we load the 'B' register with $128 + 7$ before transferring it to the 'A' register and out to the port is because there needs to be a slight delay between sending the data and sending the destination register. (If it seems a bit back to front, speak to TEXAS the chip manufacturer, 'cause I don't know)

Line 1170 has a LABEL .end because it jumps here from the menu. When it hits the RET mnemonic it actually returns to BASIC. The .end LABEL could be placed at any of the RET mnemonics but I separated it out for clarity! NOW the really complicated bit. (It took me nearly 7 hours to get lines 1180 to 1310 to do what I wanted them to, but I am a beginner!!!!) Lines 1180 and 1190 check to see that the value in register 'A' is '9' or

less and passes it to line 1230 if it is, otherwise lines 1200 and 1210 check to see if it is the letter A or higher. If the value is between '9' and the letter A then it is rejected. Line 1220 subtracts 7 from register 'A' to compensate for the invalid characters between '9' and the letter A. Line 1230 subtracts 48 from register A to give a value from 0 to 15 rather than the ASCII '0' to 'F'. Lines 1240 and 1250 make sure that it is not less than 0 and 1260 and 1270 make sure it is not more than 16.

The whole thing could, no doubt, be done in a much more sophisticated way but it does work. Well lesson number two is over! You have now learnt to read the keyboard as well as put things to the screen and lots more about registers and COMPARES. Next month I will try to sort out something on accessing the disc. This will actually utilise some of the DOS commands.

```

120 TCOL=64312
320 .start LD A,12 ;Clear
330 RST 8 ;screen
340 DEFB 158 ;
350 .menu LD HL,mmess1 ;get first message
360 CALL PRM ;and print it
370 LD HL,mmess2 ;get next message
380 CALL PRM ;and print it
390 LD HL,mmess3 ;get next
400 CALL PRM ;print it
410 LD HL,mmess4 ;get next
420 CALL PRM ;print it
430 .getkey RST 8 ;M/C call for
440 DEFB 156 ;read keyboard and get character in 'A' register
450 CP 49 ;COMPARE what is in 'A' register with 49
460 CALL Z,chtxt ;if the difference is ZERO call this routine
470 CP 50 ;COMPARE it with 50
480 CALL Z,chbg ;if difference is zero call routine
490 CP 51 ;COMPARE it with 51
500 CALL Z,chbdrp ;if zero call routine
510 CP 52 ;COMPARE with 52
520 JP Z,end ;if zero GOTO routine
530 JP menu ;if any other key go back to beginning
540 .PRM LD B,(HL) ;get no of characters in message into 'E' register
550 INC HL ;point to first character of message
560 .PRM1 LD A,(HL) ;get it into the 'A' register
570 RST 8 ;M/C call
580 DEFB 158 ;print it
590 INC HL ;point to next
600 DEC B ;decrease no of characters in message
610 RET Z ;return if finished
620 JP PRM1 ;else do it again
630 .chtxt LD A,12 ;clear
640 RST 8 ;screen
650 DEFB 158 ;again
660 LD HL,mmess1 ;point to message
670 CALL PRM ;print it
680 LD HL,tmess1 ;point to message
690 CALL PRM ;print it
700 RST 8 ;M/C call
710 DEFB 156 ;for get character
720 CALL valid ;call is it 1-9 or A-F see notes in that section
730 JP C,chtxt ;if no good go back and start again
740 ADD A,A ;the clever bit, this line multiplies by 2
750 ADD A,A ;and by 2 again, thats 4 ;text foreground
760 ADD A,A ;and by 2 again, thats 8 ;multiplied
770 ADD A,A ;and by 2 again, thats 16 ;by 16
780 LD B,A ;save it

```



```

790 LD A,(TCOL)      ;get original text colours
800 AND 15           ;get rid of foreground colour, see notes
810 ADD A,B          ;put in new foreground
820 LD (TCOL),A      ;save it where it should be
830 RET              ;return to menu

840 .chbg LD A,12    ;clear
850 RST 8             ;screen
860 DEFB 158         ;again
870 LD HL,mmess2     ;get message
880 CALL PRM          ;print it
890 LD HL,tmess1     ;get message
900 CALL PRM          ;print it
910 RST 8             ;M/C call
920 DEFB 156         ;for get character
930 CALL valid        ;check is it ok
940 JF C,chbg        ;if not go back and get another
950 LD B,A            ;save it
960 LD A,(TCOL)      ;get original text colours
970 AND 240          ;get rid of background colour
980 ADD A,B          ;add new background colour
990 LD (TCOL),A      ;save it where it should be
1000 RET              ;return to menu

1010 .chbdrp LD A,12 ;clear
1020 RST 8           ;screen
1030 DEFB 158       ;again
1040 LD HL,mmess3   ;get message
1050 CALL PRM        ;print it
1060 LD HL,tmess1   ;get message
1070 CALL PRM        ;print it
1080 RST 8           ;M/C call
1090 DEFB 156       ;for get character
1100 CALL valid      ;is it ok
1110 JF C,chbdrp     ;if not get another character
1120 OUT (9),A       ;see notes
1130 LD B,135        ;about ports
1140 LD A,B          ;and
1150 OUT (9),A       ;video addressing
1160 RET             ;return to menu
1170 .end RET;THIS ONE RETURNS YOU TO BASIC::see notes
1180 .valid CP 58     ;this is the
1190 JF C,ok          ;routine to check
1200 CP 65            ;to see if
1210 JF C,nogood      ;the key pressed
1220 SUB 7            ;is ok
1230 .ok SUB 48       ;see notes
1240 CP 0             ;to see
1250 JF C,nogood      ;how it
1260 CP 16            ;actually
1270 JF NC,nogood     ;works
1280 .good CCF        ;
1290 RET              ;
1300 .nogood SCF      ;
1310 RET              ;
1320 .mmess1 DEFB 26
1330 DEFM "1         Change text colour "
1340 DEFB 10
1350 DEFB 13
1360 .mmess2 DEFB 32
1370 DEFM "2         Change background colour "
1380 DEFB 10
1390 DEFB 13
1400 .mmess3 DEFB 30
1410 DEFM "3         Change backdrop colour "

```



```

1420 DEFB 10
1430 DEFB 13
1440 .mess4 DEFB 11
1450 DEFM "4      END "
1460 DEFB 10
1470 DEFB 13
1480 .tmess1 DEFB 40
1490 DEFM "press a key between 0 and 9 or A and F"
1500 DEFB 10
1510 DEFB 13

```

RUBIK CUBE
by
Peter Heffernan

```

10 REM
20 REM *****
30 REM *                               *
40 REM *   CUBE FOR EINSTEIN         *
50 REM *                               *
60 REM *   BY PETE HEFFERNAN         *
70 REM *                               *
80 REM *****
90 REM
100 MAG0:SPRITEOFF:CLS:BCOLO
110 PRINTCHR$(20):REM Turn cursor off
120 PRINT@5,3;"*****"
130 PRINT@5,4;"*   INSTRUCTIONS           *"
140 PRINT@5,5;"*   FOR THE CUBE             *"
150 PRINT@5,6;"*****"
160 PRINT@3,8;"ONLY THREE FACES OF THE CUBE WILL"
170 PRINT@3,9;"BE DISPLAYED AT ANY TIME"
180 PRINT@3,10;"THE LETTERS AROUND THE CUBE INDICATE   THE ROW TO BE
    ADJUSTED.AFTER THE      PROMT(WHAT MOVE)YOU INPUT THE"
190 PRINT@3,13;"LETTER OF YOUR CHOICE,AT WHICH POINT   YOU WILL BE
    ASKED FOR A DIRECTION      ie UP^ DOWN: RIGHT] LEFT[,"
200 PRINT@3,17;"NOTE THE DIRECTION KEYS ARE NOT THE   CURSOR KEYS"
210 PRINT@3,21;"PRESS ANY KEY TO START"
220 K=INCH
230 IFK>0THEN GOTO240
240 ORIGIN0,0
245 MAG0
260 REM Set up initial colours
280 F1=15 :F2=15:F3=15:F4=15:F5=15:F6=15:F7=15:F8=15:F9=15
290 L1=7:L2=7:L3=7:L4=7:L5=7:L6=7:L7=7:L8=7:L9=7
300 R1=4:R2=4:R3=4:R4=4:R5=4:R6=4:R7=4:R8=4:R9=4
310 S1=12:S2=12:S3=12:S4=12:S5=12:S6=12:S7=12:S8=12:S9=12
320 T1=10:T2=10:T3=10:T4=10:T5=10:T6=10:T7=10:T8=10:T9=10
330 B1=6:B2=6:B3=6:B4=6:B5=6:B6=6:B7=6:B8=6:B9=6
350 REM Set up alpha letters around the cube
370 SPRITE1,84,88,3,65
380 SPRITE2,84,57,5,66
390 SPRITE3,84,28,9,67
400 SPRITE4,100,8,11,68
410 SPRITE5,127,8,6,69
420 SPRITE6,154,8,15,70
430 SPRITE7,177,12,4,71
440 SPRITE8,193,24,12,72
450 SPRITE9,209,36,14,73
460 CLS
470 PRINT@1,9;"PRESS"
480 PRINT@1,11;"(R)TO RESET"
490 PRINT@1,13;"(X)TO EXIT"

```

```

500 PRINT@1,15;"(T)TO TURN"
510 PRINT@1,16;"WHOLE CUBE"
520 Q=0
540 REM Draw and colour front face
560 GCOLF1
570 ORIGIN0,0:GOSUB760
580 GCOLF2
590 ORIGIN27,0:GOSUB760
600 GCOLF3
610 ORIGIN54,0:GOSUB760
620 GCOLF5
630 ORIGIN27,-31:GOSUB760
640 GCOLF4
650 ORIGIN0,-31:GOSUB760
660 GCOLF6
670 ORIGIN54,-31:GOSUB760
680 GCOLF7
690 ORIGIN0,-62:GOSUB760
700 GCOLF8
710 ORIGIN27,-62:GOSUB760
720 GCOLF9
730 ORIGIN54,-62:GOSUB760
740 ORIGIN0,0
750 GOTO780
760 DRAW93,96TO113,96TO113,72TO93,72TO93,96:FILL100,80
770 RETURN
790 REM Draw and colour top face
810 GCOLT7
820 ORIGIN-6,2:GOSUB1000
830 GCOLT8
840 ORIGIN22,2:GOSUB1000
850 GCOLT9
860 ORIGIN50,2:GOSUB1000
870 GCOLT4
880 ORIGIN8,16:GOSUB1000
890 GCOLT5
900 ORIGIN36,16:GOSUB1000
910 GCOLT6
920 ORIGIN65,16:GOSUB1000
930 GCOLT1
940 ORIGIN23,30:GOSUB1000
950 GCOLT2
960 ORIGIN51,30:GOSUB1000
970 GCOLT3
980 ORIGIN79,30:GOSUB1000
990 ORIGIN0,0:GCOL7:GOTO1050
1000 DRAW102,102TO112,112TO132,112TO122,102TO102,102:FILL110,106
1010 RETURN
1030 REM Draw and colour side face
1050 GCOLS1
1060 ORIGIN 5,-2:GOSUB1240
1070 GCOLS4
1080 ORIGIN5,-33:GOSUB1240
1090 GCOLS7
1100 ORIGIN5,-65:GOSUB1240
1110 GCOL S2
1120 ORIGIN20,11:GOSUB1240
1130 GCOL S5
1140 ORIGIN20,-20:GOSUB1240
1150 GCOLS8
1160 ORIGIN20,-51:GOSUB1240
1170 GCOLS3
1180 ORIGIN35,26:GOSUB1240

```



```

1190 GCOL S6
1200 ORIGIN35,-6:GOSUB1240
1210 GCOLS9
1220 ORIGIN35,-37:GOSUB1240
1230 ORIGIN0,0:GOTO1260
1240 DRAW169,100TO169,76TO178,85TO178,109TO169,100:FILL175,90
1250 RETURN
1260 PRINT@1,2;"WHAT MOVE(A-I)"
1270 P=INCH
1280 IFP=66THEN1550:REM B MOVE
1290 IFP=70THEN2060:REM F MOVE
1300 IFP=69THEN1940:REM E MOVE
1310 IFP=72THEN2340:REM H MOVE
1320 IFP=68THEN1810:REM D MOVE
1330 IFP=71THEN2200:REM G MOVE
1340 IFP=65THEN1410:REM A MOVE
1350 IFP=67THEN1670:REM C MOVE
1360 IFP=73THEN2460:REM I MOVE
1370 IFP=84THEN2620:REM T MOVE
1380 IFP=88THEN2600:REM X EXIT PROG
1390 IFP=82THEN CLS:GOTO240:REM RESET R
1400 GOTO1270
1410 REM A MOVE
1420 PRINT@1,3;"LEFT(L)OR RIGHT(R)"
1430 S=INCH
1440 IFS=91THEN1510
1450 IFS=93THEN1470
1460 GOTO1430
1470 REM RIGHTA
1480 SWAPT7,T1:SWAPT1,T3:SWAPT3,T9:SWAPT4,T2:SWAPT2,T6:SWAPT6,T8
1490 SWAPF1,L3:SWAPL3,R3:SWAPR3,S1:SWAPF2,L2:SWAPL2,R2:SWAPR2,S2:SWAPF3,
    L1:SWAPL1,R1:SWAPR1,S3
1500 IFQ=5THEN1640ELSE460
1510 REM LEFT A
1520 SWAPT7,T9:SWAPT9,T3:SWAPT3,T1:SWAPT8,T6:SWAPT6,T2:SWAPT2,T4
1530 SWAPF1,S1:SWAPS1,R3:SWAPR3,L3:SWAPF2,S2:SWAPS2,R2:SWAPR2,L2:SWAPF3,
    S3:SWAPS3,R1:SWAPR1,L1
1540 IFQ=5THEN1610ELSE460
1550 REM B MOVE
1560 PRINT@1,3;"LEFT(L)OR RIGHT(R)"
1570 S=INCH
1580 IFS=91THEN1610
1590 IFS=93THEN1640
1600 GOTO1570
1610 REM LEFT B
1620 SWAPF4,S4:SWAPS4,R6:SWAPR6,L6:SWAPF5,S5:SWAPS5,R5:SWAPR5,L5:SWAPF6,
    S6:SWAPS6,R4:SWAPR4,L4
1630 IFQ=5THEN1730ELSE460
1640 REM RIGHT B
1650 SWAPF4,L6:SWAPL6,R6:SWAPR6,S4:SWAPF5,L5:SWAPL5,R5:SWAPR5,S5:SWAPF6,
    L4:SWAPL4,R4:SWAPR4,S6
1660 IFQ=5THEN1770ELSE460
1670 REM C MOVE
1680 PRINT@1,3;"LEFT(L)OR RIGHT(R)"
1690 S=INCH
1700 IFS=91THEN1730
1710 IFS=93THEN1770
1720 GOTO1570
1730 REM RIGHT C
1740 SWAPB7,B9:SWAPB9,B3:SWAPB3,B1:SWAPB8,B6:SWAPB6,B2:SWAPB2,B4
1750 SWAPF7,S7:SWAPS7,R9:SWAPR9,L9:SWAPF8,S8:SWAPS8,R8:SWAPR8,L8:SWAPF9,
    S9:SWAPS9,R7:SWAPR7,L7
1760 GOTO 460

```

```

1770 REM LEFT C
1780 SWAPB7,B1:SWAPB1,B3:SWAPB3,B9:SWAPB4,B2:SWAPB2,B6:SWAPB6,B8
1790 SWAPF9,L7:SWAPL7,R7:SWAPR7,S9:SWAPF8,L8:SWAPL8,R8:SWAPR8,S8:SWAPF7,
    L9:SWAPL9,R9:SWAPR9,S7
1800 GOTO460
1810 REM D MOVE
1820 PRINT@1,3;"UP(^)OR DOWN(:)"
1830 S=INCH
1840 IFS=94THEN1900
1850 IFS=58THEN1870
1860 GOTO1830
1870 SWAPL7,L1:SWAPL1,L3:SWAPL3,L9:SWAPL4,L2:SWAPL2,L6:SWAPL6,L8
1880 SWAPF1,T1:SWAPT1,R7:SWAPR7,B7:SWAPF4,T4:SWAPT4,R4:SWAPR4,B4:SWAPF7,
    T7:SWAPT7,R1:SWAPR1,B1
1890 IFQ=5THEN2000ELSE460
1900 REM D UP
1910 SWAPL7,L9:SWAPL9,L3:SWAPL3,L1:SWAPL8,L6:SWAPL6,L2:SWAPL2,L4
1920 SWAPF1,B7:SWAPB7,R7:SWAPR7,T1:SWAPF4,B4:SWAPB4,R4:SWAPR4,T4:SWAPF7,
    B1:SWAPB1,R1:SWAPR1,T7
1930 IFQ=5THEN2030ELSE460
1940 REM E MOVE
1950 PRINT@1,3;"UP(^)OR DOWN(:)"
1960 S=INCH
1970 IFS=94THEN2030
1980 IFS=58THEN2000
1990 GOTO1960
2000 REM E DOWN
2010 SWAPF2,T2:SWAPT2,R8:SWAPR8,B8:SWAPF5,T5:SWAPT5,R5:SWAPR5,B5:SWAPF8,
    T8:SWAPT8,R2:SWAPR2,B2
2020 IFQ=5THEN2160ELSE460
2030 REM E UP
2040 SWAPF2,B8:SWAPB8,R8:SWAPR8,T2:SWAPF5,B5:SWAPB5,R5:SWAPR5,T5:SWAPF8,
    B2:SWAPB2,R2:SWAPR2,T8
2050 IFQ=5THEN2120ELSE460
2060 REM F MOVE
2070 PRINT@1,3;"UP(^)OR DOWN(:)"
2080 S=INCH
2090 IFS=94THEN2120
2100 IFS=58THEN2160
2110 GOTO2080
2120 REM F UP
2130 SWAPS7,S9:SWAPS9,S3:SWAPS3,S1:SWAPS8,S6:SWAPS6,S2:SWAPS2,S4
2140 SWAPF3,B9:SWAPB9,R9:SWAPR9,T3:SWAPF6,B6:SWAPB6,R6:SWAPR6,T6:SWAPF9,
    B3:SWAPB3,R3:SWAPR3,T9
2150 GOTO 460
2160 REM F DOWN
2170 SWAPS7,S1:SWAPS1,S3:SWAPS3,S9:SWAPS4,S2:SWAPS2,S6:SWAPS6,S8
2180 SWAPF3,T3:SWAPT3,R9:SWAPR9,B9:SWAPF6,T6:SWAPT6,R6:SWAPR6,B6:SWAPF9,
    T9:SWAPT9,R3:SWAPR3,B3
2190 GOTO 460
2200 REM G MOVE
2210 PRINT@1,3;"UP(^)OR DOWN(:)"
2220 S=INCH
2230 IFS=94THEN2260
2240 IFS=58THEN2300
2250 GOTO2220
2260 REM G UP
2270 SWAPF7,F1:SWAPF1,F3:SWAPF3,F9:SWAPF4,F2:SWAPF2,F6:SWAPF6,F8
2280 SWAPS1,B9:SWAPB9,L7:SWAPL7,T7:SWAPS4,B8:SWAPB8,L4:SWAPL4,T8:SWAPS7,
    B7:SWAPB7,L1:SWAPL1,T9
2290 GOTO460
2300 REM G DOWN
2310 SWAPF7,F9:SWAPF9,F3:SWAPF3,F1:SWAPF8,F6:SWAPF6,F2:SWAPF2,F4

```



```

2320 SWAPS1,T7:SWAPT7,L7:SWAPL7,B9:SWAPS4,T8:SWAPT8,L4:SWAPL4,B8:SWAPS7,
      T9:SWAPT9,L1:SWAPL1,B7
2330 GOTO460
2340 REM H MOVE
2350 PRINT@1,3;"UP(^)OR DOWN(:)"
2360 S=INCH
2370 IFS=94THEN2400
2380 IFS=58THEN2430
2390 GOTO2360
2400 REM H UP
2410 SWAPS2,B6:SWAPB6,L8:SWAPL8,T4:SWAPS5,B5:SWAPB5,L5:SWAPL5,T5:SWAPS8,
      B4:SWAPB4,L2:SWAPL2,T6
2420 GOTO460
2430 REM H DOWN
2440 SWAPS2,T4:SWAPT4,L8:SWAPL8,B6:SWAPS5,T5:SWAPT5,L5:SWAPL5,B5:SWAPS8,
      T6:SWAPT6,L2:SWAPL2,B4
2450 GOTO460
2460 REM I MOVE
2470 PRINT@1,3;"UP(^)OR DOWN(:)"
2480 S=INCH
2490 IFS=94THEN2520
2500 IFS=58THEN2560
2510 GOTO2480
2520 REM I UP
2530 SWAPR7,R1:SWAPR1,R3:SWAPR3,R9:SWAPR4,R2:SWAPR2,R6:SWAPR6,R8
2540 SWAPS3,B3:SWAPB3,L9:SWAPL9,T1:SWAPS6,B2:SWAPB2,L6:SWAPL6,T2:SWAPS9,
      B1:SWAPB1,L3:SWAPL3,T3
2550 GOTO460
2560 REM I DOWN
2570 SWAPR7,R9:SWAPR9,R3:SWAPR3,R1:SWAPR8,R6:SWAPR6,R2:SWAPR2,R4
2580 SWAPS3,T1:SWAPT1,L9:SWAPL9,B3:SWAPS6,T2:SWAPT2,L6:SWAPL6,B2:SWAPS9,
      T3:SWAPT3,L3:SWAPL3,B1
2590 GOTO460
2600 REM X (EXIT PROG)
2610 CLS:SPRITEOFF:RST:BCOL4:END
2620 REM TURN CUBE
2630 Q=5
2640 PRINT@1,3;"LEFT([)OR RIGHT(])"
2650 PRINT@1,5;"UP(^)OR DOWN(:)"
2660 S=INCH
2670 IFS=91THEN1510
2680 IFS=93THEN1470
2690 IFS=94THEN1900
2700 IFS=58THEN1870
2710 GOTO2640

```

Bulletin boards, Electronic mail, and Veiwdata systems.
by

Michael Fitzgibbons

In a recent blaze of publicity the Department of Trade and Industry (DTI), said that it was about to spend 1 million pounds equipping every middle and secondary school in Britain with a modem, this being a follow up to the micro in school project of a few years ago, this is the final link in the setting up of a new educational database called NERIS, (National Educational Resources Information Service), for those of us that have been out of school for more years than they care to remember it might be time to try to catch up on some of the things that most school boys know can be done with the micro and a modem. First what is a modem, the name comes from (MODulator DEModulator), when ever we use a telephone to contact another person, the words that are spoken in to the microphone are converted in to an electrical signal (modulated) before being passed down the telephone line, and in the same way electrical signals are converted in to sound

(demodulated) and output via the speaker in the telephone handset, and as computers use electrical signals and not sound then if two computers are to converse via the telephone then we need some device to let us convert between the two, this being the task of the modem. The modem uses two sets of audio tones to represent the computer data, one set being the ORIG (originate) and the other ANS (answer), the modem can be used in either FULL DUPLEX or in HALF DUPLEX mode, by convention the computer system that starts the data link uses the ORIGINate tones, and the computer system that receives the call uses the ANSwer tones, some systems can only work with data going in one direction at any moment in time, this is the HALF DUPLEX mode, and if the system supports FULL DUPLEX then data can be passed in both directions at the same time, the speed of communication must also be standardised and there are several different BAUD rates that are commonly used, BAUD is a measurement of the speed of transfer of data characters, most common being 300 and 1200 BAUD, some systems use two different speeds in each direction eg 1200/75 where one system sends at 1200 baud and receives at 75 baud, the other system sends data at 75 baud and receives the data at 1200 baud, there are quite a large number of standards available, and so for the purpose of this article we will describe one of the most common standards, 300/300, where data is passed from both locations at the same speed, but each of the terminals uses a different set of tones.

Now that we know a little of the protocols used we can look at a typical bulletin board, and what it has to offer, and as the Einstein computer uses a DOS that is very similar to CP/M, then a look at one of the RCP/M (remote cp/m) bulletin boards will be of most interest, The bulletin board that I use is located in Leconfield nr Beverly North Yorkshire, and is run by a SYSOP (systems operator) called Martin Taylor, the bulletin board can be split up into several specific parts, the first of these is entered when you first contact the board, you will be asked to supply your name and a suitable password for future use, and you must remember the password for the next time you contact the board. The next part is the MBBS Functions, each of the functions are activated by pressing a single letter followed by the enter key.

MBBS FUNCTIONS. A = Auto wrap toggle
 K = Kill message
 S = Scan messages.
 E = show bulletins
 M = (more) toggle
 T = Talk with SYSOP.
 C = exit to CP/M
 N = Set £ of Null
 U = New User Message.
 E = Enter message
 F = Change password
 W = Welcome message.
 G = Goodbye (hangup)
 Q = Quick Summary
 X = eXpert mode toggle.
 H = show help file
 R = Read messages
 ? = Print menu.

MBBS Function (A,B,C,E,G,H,K,M,N,P,Q,R,S,T,U,W,X, or ?)
 (Ctrl S pauses / Ctrl C aborts / Ctrl x skips)

There is a second menu available (help file H <CR>), this gives more information on the functions available, now we will press the C key and then the ENTER or <CR> key, we are now asked to enter a password, this is not the password that we chose on entering the system, this is a secondary password based on a com file that is included with most implementations of CP/M, but not with the Einstein, the program name is PIP, the prompt refers to the classic GREAT EXPECTATIONS as a hint to the password. We are now confronted by a second menu, this time not of options or functions, but of

programs and commands that will help to find, upload, download, or display the programs stored in the different user areas.

WIS (wherisit) Find a program WISMEX*.x will say where it is.

MBBS RE-enter MBBS.

DIR See whats here DIR / gives help.

LUX libname "log in" to specific .LBR files

TYPE filename.ext Look at ascii text files (normal or squeezed).

SEND filename.ext Download specific file from here to you, (XMODEM-S).

SENDK filename.ext Download in K blocks.

RECV filename.ext Upload specific file from you to here, (XMODEM-R).

WHATSFOR name Describe a program, by itself gives the lot.

WHATSNEW or NEW Whats been uploaded gives latest first.

FWD or SYSMAP Gives a list of user areas.

BYE disconnect (quick).

BYE C disconnect and leave message for SYSOP.

HELP More RCP/M command descriptions.

Remember change user areas like 5: or UTILITY: (note colon: !)

Entering CP/M.

For those amongst us that are not familiar with the way that CP/M works, there might be some confusion about user areas, under CP/M a disk is split up into 16 user areas, also the disk drives are usually identified by a letter, what would be drive 0 on the Einstein would be drive A under CP/M, also we could store programs on one disk but split up into user areas, we might have a program whose name is FRED.COM on A0: and a program BILL.COM on A1: let us now enter USER 1, and ask for a DIRectory, we would get BILL.COM, if we now enter USER 0, and ask for a DIRectory we would now get FRED.COM, so we could use USER areas 0 to 15 to split up our programs, the prompt on screen would be A1> if we were in user area 1, and A2> if we were in user area 2. The command TYPE is the same as the Command DISP on the Einstein, and will display ascii files on the screen. The reference to normal or squeezed ascii files is when a normal ascii file is treated in a way that will compact the file in to a smaller than normal amount of disk space, this allowing more files to be stored on a disk, each user area is entered by entering a user number, eg A1> A4:, this will now alter the prompt to A4>, and the programs in user area A4 will be available.

Now for a look at the other commands, WIS is used to find a program stored on disk, if we wanted to find FRED.COM we would enter WIS FRED.COM the computer would search all user area's on all disks until a match was found, it is possible that there might be two programs that have the same name but different extent, then we could use the wild card search, lets say we want to find FRED.COM and a document file that gives instructions of the use of FRED.COM called FRED.DOC, then we would enter WIS FRED.* the asterisk being the wild card, but if there were to be another file with the name FRED.BAK then the system would report not only the location of FRED.COM and FRED.DOC but also FRED.BAK and infact all programs starting with FRED.whatever, when the system reports back it indicates what disk and what user area the wanted files reside on.

MBBS is the comand to leave CP/M and to return to the main bulletin board menu.

DIR is the same as the DIR command on the Einstein.

LUX allows for us to log in to a program with the extent .LBR eg FRED.LBR, this is a library file, the file contains one or more programs, the normal use of a library file is to store the program and all other files that are a part of it, eg FRED.COM also FRED.DOC and FRED.BAK etc, in one place and they then could be downloaded one after the other.

TYPE is the same as DISP on the Einstein when looking at ascii files, but when we want to look at or download a hex file we would enter HEXTYPE filename.HEX etc.

SEND SENDK RECV are all commands based around the XMODEM program, and are used to send and receive programs, more on these later.

WHATSFOR/FOR give a short description of the programs available, the last list that I downloaded went to thirty pages on my dotmatrix printer !!!, and if anyone wants a copy of the same (provided that I dont get

swamped) can have it for the cost of postage, if the number who want a copy is large then I will arrange for photocopies to be made, and the cost of them must also be included.

WHATSNEW or NEW will give a list of the latest programs to be loaded on to the board, starting with the latest.

PWD SYSMAP gives a list of the user area's including what they are assigned for, eg A12:HEX FILES> or A0:CPM> etc.

BYE is the quick way to leave the board, never just hang up the phone to end the connection, it causes problems by delaying the time taken before someone else can use the system, and remember the SYSOP will know who did it !!!.

BYE C leave the system but leave a message for the SYSOP before hanging up.

HELP gives more information on the commands available.

Maria Cottage,
Baird Court,
1 Station Road,
Bexhill-On-Sea,
East Sussex,
TN40 1RE

Here are some "ramblings" about using a serial printer with the Einstein. Hopefully some of your members will find this useful in saving them some of the many hours of "trials and tribulations" that I have had.

The tip in the November issue is very good; the only problem is that there are different versions of MOS. The code to look for is "CF 9F". The CF is a ROM call instruction and the 9F is the code for the parallel printer. (See pages 15 to 19 of the September Einstein User Magazine). My "CF 9F" is at FA3F, and changing 9F to A0 (using MFA3F in MOS and typing A0, <ENTER>, Y <ENTER>) does allow a serial printer to be used via the RS232 port in most instances. DON'T FORGET TO SET THE BAUD RATE!!! Also remember that what used to be sent to the "centronics" parallel port will now be sent to the RS232 port. This works particularly well in BBC BASIC. <CTRL> P will not activate your serial printer and <CTRL> A will provide a screen dump to the printer. So far, the only way I have been able to set an "echo" facility (the printer 'echoing' the PRINT statements from the VDU) is by using *OPT and running through a section twice. Is there any one else out there who has been able to set the "echo" facility in BBC BASIC? Of course, one does not need to change the "CF 9F" code to use the serial port in this case. *OPT 1 will cause all further output to be sent to the RS232 port.

Now for the bad news. I have spent many hours trying to set my KUMA software to run with my serial printer and have been unsuccessful to date. Not being all that familiar with machine code I don't know where to look for the instruction that sets the printer going. When I enquired from Dr. Mike Baylis he said that it would not be possible to get any KUMA software to run with a serial printer. How about it you "Hackers" can you find out how to re-direct the PRINT command in their software to the RS232 port?

Mr Bob Towers seems to have almost worked miracles with the RS232 port. Can you help Mr Towers? I stumbled on your ddoouubbllee pprriinntt problem by accident today. When the RS232 port is enabled "twice" it gives this double print. For example, in BBC BASIC, if the "CF 9F" code has been changed to "CF A0", <CTRL>P has been used to toggle the printer and then *OPT 1 is used to print anything, all the characters appear twice. Somewhere along the line you must have enabled the RS232 twice. I was very interested to read that you have been able to get an echo between the RS232

and the screen. Can you please let me know how you did this.

To date I have not been able to get my serial printer to give a printout from MOS, even though I've changed the 9F code to A0. Anybody else been more successful?

I had a similar problem to that of S. Petersen of Denmark in having a disc that would not "boot". I still don't know why, but I got around it by using a blank disc and "backing up" a "booting" disc, then erasing all the programs I didn't want and using "COPY" to copy the programs on the original "no-booting" disc and, Presto! A copy that "boots".

Finally, if you're about to despair with your serial printer then the answer may be to set a parallel to serial interface. One company that supplies a reasonably cheap one is DIGITASK in East Grinstead, Tel (0342) 24631. (Price £48.24 incl p&p and VAT). I spoke to Chris there who promised a full refund if the interface did not work with the Einstein. I haven't succumbed yet so can't say how good it is, but if my frustration continues much longer I think it's going to be the only way out.

I hope I won't be shot down in flames for this, but I must say that I was very disappointed in THE BRAIN. Having been so widely promoted I really expected a full colour, glossy magazine. Instead I received a nicely bound "News Letter" which I felt was similar to the one received from you. Definitely not what I expected at that price.

Please forgive me if I've rambled on a bit, but I did feel that our User Group is the friendly kind of place where one could be free to "chat". The Editor has my full permission to scrap, delete, alter or do whatever he wants with my letter. (Even out it in File 13).
Yours faithfully,
TONY ROBSON.

