INDEPENDENT NATIONAL USER GROUP

FOR THE BBC MICROCOMPUTER

BEEBUG NEWSLETTER

VOLUME 1 NUMBER 4 JULY 1982

CONTENTS

Main Articles

Editorial	1
Patchwork program	4
Key Define	5
Teletext	7
Structuring (Part 2)	11
Software Review	14
Member's Corner	16
Useful Articles	16
User Port (Part 2)	17
Free Memory	19
BBC Bugs Fix	21
Screen Dump	23
Input Function	24
Program Transfer	27
String Handling Tip	28
Beebug Hams	29
Points Arising	30
Chunky Invaders	31
Discounts	34
User Group Index	34
Competition	35
Software Library	35
If you write to us	36

Hints and Tips

3 OSBYTE INKEY.

4 Screen paging
INPUT and INPUTLINE.

6 *FX 16,0 (faster programs).

10 String variable sort.

22 Amazing Sounds
Silly Renumbering.

26 Randomise.

29 Cassette LOAD & SAVE problems.

30 *TV 0,1
RS423 receive.

BRITAIN'S LARGEST SINGLE-MICRO USER GROUP

EDITORIAL

BEEBUG notes

As you should be able to tell by the feel of the magazine, we have considerably increased its size this issue. The extra eight pages should somewhat make up for the fact that there is no August issue. (We produce 10 issues per year omitting August and December). It should also mean that we do not have to squeeze everything in as we did last month. Unfortunately things seem even tighter this issue, and in fact, we have had to leave out articles on 'Logic' and 'Screen Manipulation' in favour of more urgent items. This expansion also coincides with the appearance of a small amount of advertising. We have put this in, not for reasons of revenue, because at this level the revenue is not very great, but because we think that a limited amount of advertising for products related to the BBC machine can be positively useful. It is our intention to increase the amount of advertising in future issues, in order to give a fuller cross-section of products offered. But we will be keeping advertising space to a small proportion of the total magazine. For this reason we are limiting advertisements to half page spreads.

The extra space has allowed us to put in a greater number of articles: and for the first time, some of these are by BEEBUG members. We hope this trend will continue. If you would like to write for us, it would probably be best to warn us in advance, to check that the subject that you intend to cover is one that we want to include. The copy date for the next issue is 15th August, though we would like indications of substantial articles before then. If you are writing for us, the most useful form of presentation would be a cassette file written with the 'Mini text editor' published last month, and available in fuller form through the software library.

The closing date for our first software competition has now passed, and we have had a good batch of entries. Because many arrived at the last minute, we have not been able to put winners names in this issue, as we had rather optimistically hoped. Winners will all be informed by post by 15th August, and their names will appear in the next newsletter. Because many members will only just have received their machines, we will be running a second competition, open to all members, with a closing date of 30th September. The best prizewinners from both competitions will additionally be offered a commission to write further programs for our software library. Members with some programming experience may alternatively prefer to offer software of particular merit for consideration on a commission basis.

We opened our software library last month with 4 cassettes containing a total of 9 programs, and the response to this has been good. This month we shall be adding 2 more cassettes, and a full list of titles on offer to members is given elsewhere in the magazine. In this issue we have also presented the first of a series of software reviews.

Acorn News

Most members should by now have received a mail shot from Acorn via BL. This contains a list of Acorn dealers, the address of the new Acorn service centre, a 6 months guarantee card, and an advertisement from Acornsoft, as well as a sheet on BEEBUG. We had to put in our text for this over two months ago, which explains the incorrect membership size. We are now, in fact, Britain's largest user group dedicated to a single micro, and we have well over 7000 members, not the 3000 suggested.

The Manual, and O.S. ROM 1.0

Both the full user guide and the 1.0 operating system have now been finalised. The 1.0 operating system should now be available in EPROM to those who order the floppy disc interface. As of a week or two ago the 1.0 OS is now with Hitachi being ROMmed. It is expected that 1.0 ROMs will be available in about 3 months time (see back issues for explanantion of ROMs and operating systems).

A few completed versions of the full user guide are also in circulation. It is 540 pages in length, so you had better check that your letter box is big enough. The guide is not yet available in large numbers. This should not take too long however, though obviously "late May" was a somewhat optimistic prediction from Acorn. Anyway it should make a nice Christmas present.

BL on Bs

When we talked to Hermann Hauser at the Computer Fair early in April, he told us that model As ordered on March 31st should be despatched within 6 weeks, while model Bs should be despatched within 10 weeks. The situation for As has improved significantly beyond this, to the extent that they are now ex-stock (just). But they got it all wrong with the model B predictions - and not for reasons of component failure. People who ordered on March 31st have been given delivery dates of about 5 months hence; and there is unfortunately now a backlog of around 11,000 model B orders. Ideally they should be being fulfilled at around 1000 per week, but production has not reached this figure for a long time (if ever). Production however is apparently moving towards this mark, and Acorn hopes to eliminate the backlog by 1st September. By that time, they hope, a third manufacturer (in Wales) will have just begun to supply machines, so adding considerable capacity.

Some machines are currently in production abroad, but contrary to rumours, these appear to have been destined solely for the foreign market. Cleartone's takeover has apparently not affected production (Cleartone produce model As only); though a strike at ICL recently caused a two-week delay in testing model Bs.

Model Bs currently being despatched appear to be from those ordered in December and January. If you have straightforward model B orders from before December that are unfulfilled, it might be worthwhile contacting Acorn about this.

Beeb Bugs

After some negotiation, we have secured the release of a fix for two important bugs in the cassette filing system affecting data and program storage. See the contents list for "BBC BUGS" article.

"Acorn User"

The long promised Acorn user magazine is out, and no one can say that it isn't glossy. If you have read the User Group release on BEEBUG you may be a bit confused, since they printed with it the name of our press officer, rather than that of David Graham, who actually wrote the article.

STOP PRESS

We have just heard that the fee for the new 1.0 ROM if your machine is already fitted with the 0.1 ROM (rather than the EPROM), will be £10!

HINTS HINTS HINTS HINTS HINTS

OSBYTE INKEY$

If you are programming in machine code, you don't want to enter Basic every time you read the keyboard. You can avoid this (even on O.S. 0.1) by calling OSBYTE 129. The call address of all OSBYTE calls is &FFF4 (indirected through &020A).

The routine waits for a character from the currently selected input channel until either a character is detected or a given time limit expires. On entry A must contain &81, and X and Y should contain the time limit in centiseconds (Y hi-byte, X lo-byte). On exit X contains read character. C=0 indicates the character is valid. C=1 indicates that an error condition has arisen, A determines the type of error). Y=&FF on exit indicates a timeout condition.

We are grateful to Acorn/BBC for permission to give details of this OSBYTE call.

PATCHWORK

This program produces different PATCHWORK patterns depending on the values for width and height that you enter. Provided that 'width' and 'height' are not simple multiples of one another, the patterns will continue to change for a long time. You can freeze the screen by pressing the 'space' bar. Pressing 'S' twice in succession allows you to start again.

This program uses 'exclusive or' plotting. You will be interested to see that it displays 5 colours all at the same time in mode 5, when mode 5 is only supposed to show 4 colours.

The program originated from a member - John Yale of Wimborne, Dorset, who wrote it originally for the RML 380Z microcomputer. S.W.

```
  1 REM      PATCHWORK
  2 REM         by
  3 REM Sheridan williams
  4 REM  from an idea by
  5 REM     John Yale
  6 REM--------------------
 15 gap%=8
 17 mode=5
 20 X=RND(-TIME)
200 MODE 7
230 INPUT"Screen  width (100-1279) ",
width%
232 IF width%<1 width%=RND(900)+379
235 PRINT'"Best effects are obtained
by making the height not an easy multip
le of the width"
238  PRINT TAB(8,15)"Press 'space' to
 freeze"
239   PRINT TAB(5,17)"or press 'S' tw
ice to start again."
240 INPUT TAB(0,5)"Screen height (100
-1023) ",height%
245 IF height%<1 height%=RND(900)+123
250 MODE mode
251 VDU 19,0,4,0,0,0
252 VDU 19,1,1,0,0,0
253 VDU 19,2,3,0,0,0
254 VDU 19,3,7,0,0,0
260 VDU 29,(1279-width%)/2;(1023-heig
ht%)/2;
265 VDU 24,0;0;width%;height%;
270  XN%=2*width%:YN%=2*height%
280  X%=width% DIV 2:Y%=height% DIV 2
290  xinc%=1:yinc%=2
300  MOVE X%,Y%
305 REPEAT
310  X%=X%+xinc%
320  GCOL 3,1:DRAW X%,Y%
330 IF X%>width% THEN X%=X%-width%:MO
VE 0,Y%:DRAW X%,Y%:GOTO 330
340 IF X%<0 THEN X%=X%+width%:MOVE wi
dth%,Y%:DRAW X%,Y%:GOTO 340
350 xinc%=-xinc%-gap%*SGN(xinc%)
360 IF ABS(xinc%)>XN% THEN xinc%=xinc
%-SGN(xinc%)*XN%
370  Y%=Y%+yinc%
380  GCOL 3,2:DRAW X%,Y%
390 IF Y%>height% THEN Y%=Y%-height%:
MOVE X%,0:DRAW X%,Y%:GOTO 390
400 IF Y%<0 THEN Y%=Y%+height%:MOVE X
%,height%:DRAW X%,Y%:GOTO 400
410  yinc%=-yinc%-gap%*SGN(yinc%)
420 IF ABS(yinc%)>YN% THEN yinc%=yinc
%-SGN(yinc%)*YN%
430 UNTIL INKEY$(0)<>""
440  IF CHR$(GET AND 223)="S" THEN 20
0 ELSE 305
```

HINTS HINTS HINTS HINTS HINTS

Mr R. Q. McMinn of Birmingham says:

I find that when looking through a long program that pressing CONTROL-SHIFT allows stepping through the listing. 'Escape' then allows you to stop for alterations.

[That really is useful, we've been trying to do that ever since we got our micro - Ed.]

Here's a quick reminder from me (S.W.):
If you want to enter strings with quotes (") and commas (,) etc in them you can't use INPUT, but you <u>can</u> use INPUTLINE. For example:

```
INPUTLINE"First line of your address",address1$
```

Another tip is that you can include in the INPUT statement much of what you can do in the PRINT statement, for example:

```
10 INPUT TAB(10,10)"1st Number",N1,TAB(20,20)"2nd Number",N2
```

KEY DEFINE

Putting the 10 user defined function keys (or "softkeys") on the BBC machine was a good idea, and so was providing a removable transparent plastic cover under which you can slide notes relating to the keys. There is a wealth of applications to which these can be put. Our mini text editor featured last month provided some examples of possible uses. Here we will take a closer look, dealing first with the syntax, and working towards a variety of applications.

Syntax

The provisional user guide p196 gives details of the key set command. Eg if you enter *KEY 0 "WORD", this will cause the word WORD to appear every time that key f_0 is pressed. The guide does not say that you can leave out the quotations, and this has the advantage that you can actually print a quotation mark on the screen. ie *KEY 0 " will cause a quote to appear on the screen when f_0 is pressed. If you are using quotes to open and close the command, you can actually get quotation marks printed simply by doubling up on the number that you need. Thus *KEY "PRINT""X""" will print PRINT"X" when you press f_0.

The provisional guide tells you how to get a key to execute a 'return' - ie by inserting |M. Thus *KEY 0 LIST|M will cause a listing each time key f_0 is pressed. The | character is produced by shifting the key to the left of the cursor control key. It has the same effect as if you pressed the 'CTRL' key during normal input. You may have already discovered that CTRL M (ie press CTRL and M at the same time) causes a return. There are other equally useful CTRL functions, such as CTRL L to clear the screen, or CTRL V followed by a number between 0 & 7 will give a mode change. Thus *KEY 7 |V7 provides an economical way of getting f_7 to produce mode 7. (See the P.U.G. p188 for other control characters.) Note incidentally the relationship between the control letters and the VDU or CHR$ calls to achieve the same function - To take an example, VDU 10, or PRINT CHR$10 cause the cursor to move down one line. The control code for this is J (the 10th letter of the alphabet).

Keys 10 to 15

As we suggested last month, soft key 10 is actually the 'BREAK' key, and this can be programmed in the same way. You cannot stop it doing its resetting function as far as we know, but you can make it do what you like after that. Eg *KEY 10 OLD|M would prevent accidental loss of programs. Note though, that if you hit the break key twice in succession with the right interval between presses you will get a 'beep', the screen will display the memory size and it will do a semi-cold start erasing all set keys.

In operating system 1.0 you can use the 5 editing keys as soft-keys after typing *FX4,2. This is not possible on OS 0.1 (though it looks from the specification sheets as if it was). On OS 0.1 you can however use *FX4,1 to allow the editing keys to be read with INKEY$() etc. Their values are COPY=&87, LEFT=&88, RIGHT=&89, DOWN=&8A, UP=&8B.

The soft key buffer (an area of memory from &0B00 to &0BFF) is (as this suggests!) 256 bytes wide, and this would appear to be shared amongst the 10 keys. This means that you can define key 0 with a string of a little over 200 characters if you want to, but there will be very little room left for any of the other keys, and when you try to define them, you will get a "Bad key" report. Unfortunately this also occurs if you try to redefine a soft key when the buffer is almost full. Because of this, if you are putting key definitions into a program that nearly fill the soft key buffer, then you should precede them with a clear key command. (eg *KEY 0 clears key 0), otherwise you will get a "Bad key" error the second time that you run the program. We are told that this minor bug has been taken care of in OS 1.0.

Because of the reasonably generous buffer allocation, each key may contain a number of commands in immediate mode separated by colons or return characters, so as to have the effect of a complete program. Eg:

*KEY 0 INPUT A:FOR B=1 TO 10:P.A^B:NEXT|M

Pressing key f_0 causes an input to be requested, and when this is entered (with a return) the 10 powers are printed. A more useful sequence is the following:

*KEY 0 DIM P%-1:P.HIMEM-P%|M

This prints out the total remaining memory, taking variable storage into account (and can be used to see if a program that runs on a model B will also run on a model A - at least 16384 bytes should be unused if this is the case).

A series of FX calls can conveniently be put into a key buffer, provided they are separated by a 'return' character rather than a colon - this can be useful for setting the serial interface for a printer - eg *KEY 0 *FX5,2|M*FX8,4|M

Quite a few members have written in to say that they always run a short program before using their machines, so as to initialise the soft keys. You can either type NEW to erase the program after running it, or else you can leave it as the first few lines of any program that you develop (assuming that the program that you are developing doesn't use the softkeys for other purposes.) The following is a list of some of the functions that members have assigned to the softkeys:
RUN, RENUMBER, LIST, MODE 7, OLD, NEW, PRINT TIME, initialise TIME to clock time, *MOTOR, *MOTOR 0, PRINT remaining memory.

We have used some of these in the key setting routine below. We have not included TIME setting or reading routines since many programs use the TIME function in ways that upset its use as a time of day clock. We have also avoided NEW since we felt that programs could be too easily lost with inadvertant use of this, followed by a Basic line entry etc. *MOTOR has not been used, but *CAT usefully turns on the cassette motor, while the 'escape' key will switch it off. Key f_3 gives a printout of memory remaining (when variables are taken into account). Keys 4,5 and 6 are left undefined. Key 7 changes to Mode 7. Key 8 gives a printer listing of a program using LISTO 7, after printout LISTO 0 is activated, since as you have probably noticed, when you use the 'copy' key in editing you create ragged edges if the listing that you edit from was performed under LISTO management. Key 9 performs Mode 7 then LIST, while key 10 executes OLD after 'Break', because of a bug you should avoid pressing the 'Break' key so programmed, when there is no program in the machine. Finally, as an aid to memory you may find it is useful to slip a piece of paper under the perspex strip above the keyboard with the key functions written on it.

Acknowledgements - Thanks are due to Brian Carrol, Chris Bingley, Peter Raper and others for their ideas on the softkeys, some of which have been incorporated into this article. D.E.G.

```
 90 REM BEEBUG KEY SET ROUTINE
100 *KEY 0 RUN|M
110 *KEY 1 *CAT|M
120 *KEY 2 CHAIN""|M
130 *KEY 3 DIM P%-1:P.HIMEM-P%|M
140 *KEY 4
150 *KEY 5
160 *KEY 6
170 *KEY 7 |V7
180 *KEY 8 LISTO7|M|BL.|M|CLISTO0|M
190 *KEY 9 |V7|NL.|M|O
200 *KEY 10 OLD|M
210 MODE7:PRINTTAB(0,10)"FUNCTION K
EYS ARE SET-PRESS ANY"
220 PRINT"KEY TO CLEAR PROG"
230 wait=GET
240 NEW
```

HINTS HINTS HINTS HINTS HINTS

SPEED HINT

Richard Harris tells us that *FX 16,0 turns off the Model B A/D converter interrupt system, and slightly speeds up processing (though I would not have thought that there could be much gained here ed.). He also suggests that a subroutine at the beginning of a program is faster than a procedure, but that a subroutine at the end of a long program is slower. It does not appear to make any difference where the procedure is in the program.

TELETEXT

To judge from members comments there is a great deal of interest in the Teletext mode - mode 7 - on the Beeb. It offers the full 8 colours even on a model A, it has a very readable character set, and it is reasonably rewarding to explore given a colour monitor or TV.

We present here two articles on the 'Teletext mode'. The first is accompanied by a very useful illustrative program sections of which are referred to at many points during the article. The program is listed at the end of this article. The second article is intended to cover some of the points not fully covered in the full user guide, and assumes that you have this to hand, or at least have read the first article.

May we thank the very many people who have written in with details of the teletext mode.

MODE 7 and the Teletext facilities PART 1

by Graham Greatrix

In MODE 7 the display characters are supplied from a Teletext character generator ROM. The characters can be displayed normal height, double height, flashing, non-flashing, and with any foreground and any background colour. The characters include upper case, lower case, and a set of graphics characters based on a slot 2 pixels wide by 3 pixels high. (See the following article). Each new line of the display is assumed to be alphanumeric, normal height, white characters on a black background, unless accompanied by a combination of control codes which alter these conditions. The following table summarises these codes:

Dec	Hex	Letter	Effect	Dec	Hex	Letter	Effect
129	81	A	Alpha Red	145	91	Q	Graphics Red
130	82	B	Alpha Green	146	92	R	Graphics Green
131	83	C	Alpha Yellow	147	93	S	Graphics Yellow
132	84	D	Alpha Blue	148	94	T	Graphics Blue
133	85	E	Alpha Magenta	149	95	U	Graphics Magenta
134	86	F	Alpha Cyan	150	96	V	Graphics Cyan
135	87	G	Alpha White	151	97	W	Graphics White
136	88	H	Flash On	152	98	X	Conceal Display
137	89	I	Flash Off	153	99	Y	Contiguous Graphics
138	8A	J	End box	154	9A	Z	Separated Graphics
139	8B	K	Start box	155	9B		.....unused.....
140	8C	L	Normal Height	156	9C		Black Background
141	8D	M	Double Height	157	9D	]	New Background
142	8E		...unused....	158	9E	↑	Hold Graphics
143	8F		...unused....	159	9F	-	Release Graphics

The column headed 'Letter' will be referred to in the final part of this set of articles. Some examples will illustrate the use of these codes. If we wish to print BEEBUG in yellow letters on a blue background, we would require a statement like this:

```
50 PRINT CHR$132;CHR$157;CHR$131;"   BEEBUG"
```

Refer to the table to see what each code does. A code will apply to the whole of a line unless changed by a succeeding code - hence the blue background extending across the screen. When a new line is reached the control codes cease to apply and need inserting again.

Now let's print BBC MICROCOMPUTER in yellow double-height letters with MICRO flashing, all on a blue background. This would be accomplished by:

```
70 PRINT SPC4;CHR$132,CHR$157;CHR$131;CHR$141;"BBC";CHR$136;"MICRO";CHR$137;
        "COMPUTER   ";CHR$156
80 PRINT SPC4;CHR$132,CHR$157;CHR$131;CHR$141;"BBC";CHR$136;"MICRO";CHR$137;
        "COMPUTER   ";CHR$156
```

Line 70 prints the top half of the letters, and line 80 prints the bottom half. After running the program notice the space between "BBC" and "MICRO" and also between "MICRO" and "COMPUTER". These are actually occupied by the control codes for 'flash on' and 'flash off' respectively. In text, it is nearly always possible to insert control codes where spaces would normally occur.
The graphics characters can be seen in green by running lines 100 to 150 of the program listed at the end of this article.

Try inserting the code for separated graphics into the two PRINT statements. Separated graphics are used to reduce the intensity of the display producing a different effect. The graphics characters provide a screen resolution of 80 pixels across by 75 pixels deep. Codes 35, 95 and 96 are not in their usual order, and a slot with all six pixels illuminated is code 255.

The graphics characters cover the range 32 to 127 decimal, and the alphanumeric characters cover the same range. Try this:

```
180 PRINT"0123abcd";CHR$151;"0123abcd"
and also
190 PRINT"ABCDEFGH";CHR$151;"ABCDEFGH"
```

The upper case characters are the same even when preceded by a graphics control code.
The spaces occupied by control codes could be very inconvenient in any complex multicoloured graphics picture. CHR$158 can be used to avoid these spaces by overwriting them with repeats of the previous character. For example, if we require a band of colour consisting of five yellow slots followed immediately by five blue slots, we could use:

```
220 PRINT CHR$147;STRING$(5,CHR$255);CHR$148;STRING$(5,CHR$255)
```

but notice how the blue control code causes the appearance of an unwanted space. Now try:

```
250 PRINT CHR$147;STRING$(3,CHR$255);CHR$158;CHR$148;STRING$(5,CHR$255)
```

This time the display is what was required. There are two control codes between the strings, and their spaces have been overwritten by a repeat of the third yellow graphics slot, this is the reason for reducing the length of the yellow string from five to three slots. Finally try lines 270 to 310 of the program listed below.

You can copy graphics characters into PRINT statements. The copy cursor will collect up hidden control codes as well as the visible characters. Programs can be modified, and saved, with their PRINT statements in full colour, and completely uncluttered by all those CHR$ codes.
Now is the time to experiment. Have a closer look at the CEEFAX, ORACLE and PRESTEL displays for a few ideas. It's hard work, but very rewarding.

Teletext Part 2 by Colin Bignell

Graphics characters

The full user guide informs us that each of the six cells in the teletext graphics character contributes a certain number to the ASCII value of a character code, but in the galley proofs that we have seen, it omits to give the values concerned. These are fundamentally as in the diagram opposite. You just add up the numbers in the cells that you require, and add 32 to this, giving the character number. Thus the shape  is produced by character number 2+4+8+64+32=110. And thus the command PRINT CHR$145;CHR$110 will print the shape in red. There are unfortunately a few exceptions to this, as you can see from running lines 100 to 150 of the program listed above. These are most notably 35, which is actually generated by code 96; 96 generated by 95; and 127 generated by code 255.

1	2
4	8
16	64

Using the graphics codes given will yield continuous graphic characters. It is, however, possible to separate the characters into discrete cells (with a thin background line dividing them), by using the code 154 (&9A) after any of the graphic

colour control codes. For example:

```
PRINT CHR$&97;CHR$255;CHR$154;CHR$255
```

will show a solid rectangle and six discrete cells for CHR$255 in the first and second positions respectively.

In the parts of the full manual that we have seen, several special codes are given, but only one code is given to stop a special effect before the end of a line. In fact, control code 156 (&9C) will cancel code 137 and restore the coloured background to black; and control code 140 will cancel code 141 and restore single-height characters. In the latter case, however, it is not apparently possible to print on the lower of the two lines used by the double height letters. Try the following:

```
10 PRINT CHR$141;"TEST 1  ";CHR$140;"TEST 2"
20 PRINT CHR$141;"TEST 1  ";CHR$140;"TEST 3"
```

Only TEST 1 will appear in double height; TEST 2 will be single height.

There is one further control code available 152 (&98). This has the effect of suppressing any further printing following it, and can be used to write hidden messages for games or quizzes. To cancel the code, either overwrite it with a space, or follow it with one of the alphanumeric colour codes, eg:

Type: `PRINT TAB(0,12);CHR$152;" TEST"`

now either `PRINT TAB(0,12);" "` or `PRINT TAB(1,12);CHR$&87`

will restore the message TEST. In the latter case `PRINT TAB(1,12);" "` will hide it again. This code will only operate for a black background. On a coloured background, it is only necessary to omit the second text colour instruction to hide a message, and insert the correct colour control code to reveal it. Eg:

```
PRINT TAB(0,14);CHR$157;"TEST"
```

will give a hidden message and

```
PRINT TAB(1,14);CHR$&81
```

will reveal it in red.

Softkeys and Teletext Characters by Graham Taylor

The softkeys can be programmed to produce mode 7 teletext graphics and control characters directly; they act like normal keys once programmed. Thus the CHR$ and VDU statements are not needed. The control characters can be typed in directly from the softkeys. These control characters can be used, within the quotes of any PRINT or INPUT statement. The format is: *KEY5¦!¦ letter Thus *KEY5¦!¦S will produce "Graphics Yellow" whenever f_5 is pressed. The letter to be substituted is listed in the table in "Teletext Part 1".

A mode 7 listing created in this way contains coloured PRINT statements! It is also possible to COPY a coloured teletext line into a PRINT statement.

```
   10 CLS:PRINT''''"  MODE 7 AND THE
 TELETEXT FACILITIES"'''
   20 PRINT "        At each stage, p
 ress"
   30 PRINT "       SPACE BAR  to con
 tinue."
   40 A=GET:PRINT:PRINT
   50 PRINT CHR$(132);CHR$(157);CHR$(
 131);"   BEEBUG"
   60 A=GET:PRINT
   70 PRINT"    ";CHR$(132);CHR$(157)
 ;CHR$(131);CHR$(141);"BBC";CHR$(136);
 "MICRO";CHR$(137);"COMPUTER   ";CHR$(
 156)
   80 PRINT"    ";CHR$(132);CHR$(157)
 ;CHR$(131);CHR$(141);"BBC";CHR$(136);
 "MICRO";CHR$(137);"COMPUTER   ";CHR$(
 156)
   90 A=GET:PRINT
  100 FOR I= 32 TO 63 STEP 7:FOR J=0
 TO 6
  110 PRINT;I+J;CHR$(146);CHR$(I+J);"
 ";CHR$(135);
  120 NEXT:PRINT:NEXT
  130 FOR I= 92 TO 127 STEP 6:FOR J=0
  TO 5
  140 PRINT;I+J;CHR$(146);CHR$(I+J);C
 HR$(135);
  150 NEXT:PRINT:PRINT:NEXT
  160 PRINT
  170 A=GET:PRINT
  180 PRINT"0123abcd";CHR$(151);"0123
 abcd"
```

```
  190 A=GET:PRINT
  200 PRINT"ABCDEFGH";CHR$(151);"ABCD
EFGH"
  210 A=GET:PRINT
  220 PRINT CHR$(147);STRING$(5,CHR$(
255));CHR$(148);STRING$(5,CHR$(255))
  230 A=GET:PRINT
  240 PRINT
  250 PRINTCHR$(147);STRING$(3,CHR$(2
55));CHR$(158);CHR$(148);STRING$(5,CH
R$(255))
  260 A=GET:PRINT
  270 PRINT CHR$(147)"7s3s3s3s35"
  280 PRINT CHR$(147)"-,5<$<$,55"
  290 PRINT CHR$(147)" £a£1sq£!5"
  300 PRINT CHR$(147)"  j 551,,%"
  310 PRINT CHR$(147)"  ""£!££"
  320 A=GET:PRINT
```

HINTS **HINTS** **HINTS** **HINTS** **HINTS**

The following hints, including this alphabetic sort function was sent in by Pete Cockerell of Leigh-On-Sea, Essex

```
1000 DEF FNsort(A$)
1010 IF LEN(A$)<2 THEN =A$
                  ELSE =FNinsert(LEFT$(A$,1),FNsort(MID$(A$,2)))
1020 DEF FNinsert(X$,Y$)
1030 IF Y$="" THEN =X$
1040 LOCAL T$
1050 T$=LEFT$(Y$,1)
1060 IF X$<=T$ THEN =X$+Y$
               ELSE =T$+FNinsert(X$,MID$(Y$,2))
```

To test if it works try PRINT FNsort("DCAB") Because this demonstrates mutual recursion, it is probable that it will run out of stack space on large strings. But then you would not use it for sorting arrays anyway.

TECHNOMATIC LTD.

Official *BBC* Stockists

Model A to Model B upgrade Kit £65.00

16K RAM 8 X 4816AP-3 100nS £21.60

FULL RANGE OF CONNECTORS AVAILABLE EX-STOCK

PRINTERS

SEIKOSHA GP100A £189.00

EPSON MX80F/T3 £360.00

NEC PC8023 £375.00

(Carriage £8 per printer)

MONITORS

14" BMC Colour 18MHz Bandwidth RGB Input £240.00 + £8 Carriage

SANYO Computer Grade Cassette Recorder £24.50

PLEASE SEND FOR OUR *BBC* LEAFLET FOR FULL DETAILS

We also stock a large range of CPUs, Memories, TTLs, CMOS & Connectors

Please add 40p P&P and 15% VAT to the order value

TECHNOMATIC LTD.

17 Burnley Road, London NW10 1ED. Tel. 01-452 1500/450 6597

Retail Shops: 15 Burnley Road, NW10. 305 Edgware Road, W2

STUCTURING PART 2

REPEAT...UNTIL

This can be used as an alternative to the FOR loop. The FOR loop suffers from the main disadvantage that we often use it when we do not know how many times it will loop; for instance when searching an array serially for a particular value. When we find the value in question we jump out of the loop. Jumping out of FOR loops is bad programming practice because it still leaves the loop active, with all the associated values on the stack; it is annoying to have to stop the loop artificially by setting the loop variable to its final value. For example, one way to terminate a FOR loop properly is to proceed as follows. Suppose we wish to exit a loop when Y=10 then we would write:

```
10 FOR X=1 TO 100                    40  section of program
20  section of program               50 NEXT X
30  IF Y=10 then X=100:GOTO 50
```

It is artificial to have to set the value of X to the final value of the loop so instead we need a REPEAT...UNTIL structure. Example: suppose we want to search an array sequentially for the value 123.

UNSTRUCTURED

```
100 FOR X%=1 TO 1000
110  IF A(X%)=123 THEN 140
120 NEXT X%
130 PRINT"Not found":STOP
140 rest of program
```

STRUCTURED

```
100 X=0
110 REPEAT: X=X+1
130  IF X>1000 THEN PRINT"Not found":STOP
140 UNTIL A(X)=123
150 rest of program
```

WARNING - REPEAT loops execute considerably more slowly than FOR loops in BBC Basic, so when speed is paramount you may have to disregard the REPEAT structure and use FOR loops.
See the BOMBER program in the last issue for an example of a program with very few GOTOs.

BBC Basic goes some of the way to enabling you to write structured programs. Some of the features missing are WHILE loops, a BEGIN...END block structure, CASE statements, and the ability to define your own variable types in addition to real, Integer (%), and string ($).

A member wrote in saying that you can always simulate a CASE statement as follows:

```
40 value=3
50 case1=100:case2=200:case3=322:case4=780
60 ON value GOTO case1,case2,case3,case4
```

In this example line 60 would then GOTO line 322. My advice is DON'T use this approach, because although BBC Basic will accept a variable in a GOTO statement, these GOTOs will not be renumbered when you renumber your program. It will be very difficult to sort this out if you forget.

The REPEAT...UNTIL loop has the drawback that it is always executed at least once; this is because the condition for exit is placed at the end of the loop. A WHILE loop, on the other hand has the condition placed at the beginning, and need not be executed at all. It is a shame that the WHILE loop has not been implemented as it is slightly more useful in my view than the REPEAT loop.

An example of a program with a WHILE loop is the following, which uses Euclid's algorithm for finding 'highest common factors':

```
10 INPUT"First factor",N1,"Second factor",N2
20 REPEAT
30   WHILE N1>N2: N1=N1-N2: ENDWHILE
40   WHILE N2>N1: N2=N2-N1: ENDWHILE
50 UNTIL N1=N2
60 PRINT"H.C.F. = ";N1
```

Defined Functions

When you want a procedure to produce a value, it is advisable to use a defined function rather than a procedure. As I said last month, BBC Basic won't allow output parameters in procedures, and this restricts their use. For example you cannot write a general procedure to swap the contents of any two variables, eg:

```
10 a=11
20 b=76
30 PROCswap(a,b)
40 PRINT a,b      this should print 76  11 but doesn't.
```

Here is a defined function using Euclid's algorithm (mentioned earlier) that returns the 'highest common factor'.

```
  10 INPUT"First factor",N1,"Second factor",N2
  20  PRINT"H.C.F. is ";FNhcf(N1,N2)
  30 END
1000 DEF FNhcf(x,y)
1010 REPEAT
1020   IF x>y THEN x=x-y
1030   IF y>x THEN y=y-x
1040 UNTIL x=y
1050 =x
```

Line 1050 may look weird, but it assigns the value x to the function, and instructs the function to return to the calling program. It may be easier if I define a non-mathematical function (ie a string function). Here is a function that returns a string with all the trailing spaces stripped off:

```
  10 INPUTLINE"Enter something with
     spaces at the end",before$
  20  PRINT"It   was>";before$;"<"
  30  after$=FNstrip(before$)
  40  PRINT"It's now>";after$;"<"
  50 END
  60 REM------------------------
1000 DEF FNstrip(string$)
1010 string$=string$+CHR$32
1020 REPEAT
1030   string$=LEFT$(string$,LEN(string$)-1)
1040 UNTIL RIGHT$(string$,1)<>CHR$32
1050 =string$
```

Line 1010 is annoyingly necessary because the REPEAT loop always executes once before it tests.

Recursion

This topic is too long to consider in this article, so I will cover it in a future article. However it is relevant here because it allows us to define functions very easily in some cases. For instance we can code our HCF function as:

```
DEF FNhcf(x,y)
LOCAL r
IF x<y THEN =FNhcf(y,x)
r=x MOD y
IF r=0 THEN =y ELSE =FNhcf(y,r)
```

Here the function calls itself. Functions that call themselves are called recursive functions. [There is rather a neat pair of functions that are mutually recursive in the "hints and tips" elsewhere in this issue.]

As a comparison between two functions, one defined recursively and one not, I have chosen one that is easy to understand. It will find the sum of the elements of an array A(1) to A(n):

Non-Recursive

```
DEF FNsum(n)
LOCAL sum
FOR i%=1 TO n
 sum=sum+A(i%)
NEXT i%
=sum
```

Recursive

```
DEF FNsum(n)
IF n=1 THEN =A(1) ELSE =A(n)+FNsum(n-1)
```

Summary

A problem should be reduced into sections, each section worked on separately and incorporated into a procedure or function. Each procedure may use other procedures.

The master program should be short enough to perceive as a whole, say 40 lines at a maximum, and there should be readily at hand a chart showing all the procedures used and their inter-dependency.

As few GOTOs as possible should be used. And always use meaningful variable names.

BOOK

I have been notified that a book entitled "Structured Programming on the BBC Microcomputer" written by Roy Atherton, and published by Ellis Horwood will appear in a few month's time. BEEBUG has asked for a pre-publication review copy, so that a review should coincide with its release. A book on structured programming by Roy Atherton should be very good. Roy has long been a proponent of structured programming and with good reason. S.W.

HINTS HINTS HINTS HINTS HINTS

R. Benitez of Ealing, London sent us a routine to read a character from anywhere on the screen, unfortunately we published one last month, but he also says:
All the 6845 registers can be accessed using the format:

```
to read     ?&FE00=register number : byte=?&FE01
to write    ?&FE00=register number : ?&FE01=byte
```

PRINTER CONTROL CHARACTERS

We are grateful to Archie Ewing for bringing this to our attention. If you try to send printer control characters embedded in text to be printed, the printer will ignore them, as some members have found. The solution is hidden away in the VDU calls in the Provisional User Guide. The way to send control characters is to use VDU 1. That is to say, suppose that you want to send a line feed character (ASCII 10), you just execute VDU 1,10. To send two line feeds, use VDU 1,10,1,10. Incidentally, this call is effective whetner or not the printer happens to be activated with a VDU 2 (or a CTRL B).

BBC COMPATIBLE hardware

ELECTRONICS APPLIED,
4, DROMORE ROAD,
CARRICKFERGUS, Co. ANTRIM BT38 7PJ
Please add 55p order P & P

BEEBUG MEMBERS 5% discount

COMPUTER TO CASSETTE LEADS
We can supply three types:-
(1) Computer to 5 Pin DIN. £3.95
(2) Computer to two 3.5mm plugs and one 2.5mm plug (most common).
(3) Computer to three 3.5mm plugs.

DELUX LEADS
manual overide of motor control allows you to rewind + f.f. etc



built in speaker
(low vol !)

5 pin DIN version also available.
£6.40

Games Paddles
With "fire"/"serve" button
£10.95 per pair.



SOFTWARE REVIEW

This month we start what we hope will be a regular series of reviews. If members have bought any other items of software that they would like to review, please send the review to the editorial address, marking the envelope "Software Review". Please try and stick to a similar format for the review. Please note that the reviews are entirely personal, and BEEBUG cannot be held responsible for views of the reviewers.

SPACE WARP Supplied by: Bug-Byte Cost: £11.50
Hardware requirements: 32k Model A or B
Reviewer: Rob Pickering

The Space Warp package comes as a cassette with 14 page booklet, and a slip of paper to place under the perspex strip to remind you of the softkey functions.

I was glad, at first, to discover that the cassette was recorded at 1200 baud. However, I soon changed my mind when it took 45 minutes before I could get it to run, because I had such severe loading problems. The cassette contains more than one program, the first program loads the second. In the manual it states that it is so complicated to load, that it is too complicated for them to tell you how to make a back-up copy, so with the loading problems I only played the game about six times.

Once running, the helpful user manual seems to tell you everything you would expect to find in a science fiction novel, but nothing clear enough about the objectives of the game, and how to play it. I read things over and over again, but still couldn't seem to get far. Wondering if my computing experience was clouding my vision, I got some total beginners to try it on their own. They came to the same overall conclusion. It leaves the user frustrated, because it is virtually 'teach yourself'.

In conclusion, SPACE WARP is probably a good game, a sort of STARTREK type game played in real-time, with sound effects and visual movement. Unfortunately I could not get a good idea of how good it is in just a few days. This is a good point for two reasons, either you will put it in a cupboard and ignore it forever with frustration, or you will find it worth perservering. If it is this difficult to learn, then it could be worth perservering as it may means that once learned it won't be too trivial.

BEEBGAMMON Supplied by:Bug-Byte Cost: £8
Hardware requirements: 32k Model A or B
Reviewer: Sheridan Williams

I could not get the 1200 baud copy to load. That makes two out of three of Bug-Byte's cassettes that were difficult to load. (This one and "Space-warp" just reviewed). Perhaps they ought to sort out their recording problems.

This program came with an A5 card outlining how to play, but it didn't tell you that there was a 300 baud copy on the unlabelled side. I only found that when I got desperate.

There are three skill levels, and the game is "you versus the computer". There is no facility for two people to play each other. It plays reasonably well at its best level, but I didn't play a long enough series to tell if it was luck or my skill.

GOLF
Supplied by: Bug-Byte, 98-100 The Albany, Old Hall Street, Liverpool L3 9EP
Cost: £7
Hardware requirements: 32k Model A or B
Reviewer: Sheridan Williams

A golf simulation game. The green colour is a bit dazzling, and you need to turn down the set's brightness. Allows one or two players. Fairly good graphics, showing aerial view of each of the fairways. Once you are on the green this converts to a worm's eye view. The game is fun to play, but I was disturbed by the number of times I holed out from over 200 yards; maybe I shouldn't be looking after BEEBUG but be on a golf circuit instead. You can select a range of clubs, and the type of hit.

SNAKE Supplied by: Computer Concepts, 16 Wayside, Chipperfield, Herts. WD4 9JJ. Tel: 09277 62955 Cost: £6.67
Hardware requirements: 32k Model A or B Reviewer: Pauline Henderson

On the cassette there were copies of the program at both speeds, and there was no problem with loading at all.

The game itself is really fun, and very, very addictive. You must imagine that you are the head of a snake, you move around the board, using the left and right cursor keys, gobbling up red blobs. For each red one you eat you gain one point, one segment of body is added to your length, and a mauve blob appears somewhere on the screen. If you gobble a mauve blob you get two points, two segments added, and a green square appears. If you hit an edge or a green square you get killed. Occasionally a white blob appears, this gets you bonus points for the speed at which you can reach it to gobble it up. The 'Death March' is played when you die, and although fun at first, it becomes dreadfully boring waiting while it finishes.

It is very exciting because eventually you are perhaps 100 segments in length, and hence your body trails around the screen. If you collide with yourself then 'death'. You have two other controls, the 'shift' key which speeds up your motion, and the 'space' bar which fires a laser destroying any obstacle in its path within 5 squares. Once you have gobbled up everything that you can, the game pauses before displaying page 2, which is a new layout, I reached page 3, but I presume that there are more. It can be played on a B&W set as the blobs are all slightly different in shape.

Very addictive, not to be played with a few beers inside you. Easy to get your left and rights muddled when travelling down the screen. Really excellent game.

SNAPPER Supplied by: AcornSoft Cost: £9.96
Hardware requirements: 32k Model B (needs 6522)
Reviewer: Sheridan Williams

This is a direct copy of the pub-type game "Puckman". The object is to negotiate a course gobbling up dots. You are being chased by ghost-like shapes which you have to evade otherwise they get you. In order to be able to kill them you must get to a blob and gobble that, once this blob is eaten the ghosts turn blue, and can now be eaten. Once eaten their eyes live and return to their breeding ground for regeneration.

This really is a superb game graphically and conceptually. I enjoyed playing it very much. [That's why this magazine is late! DEG.]

DISASSEMBLER Supplied by:Program Power Cost: £5.95 (special offer)
Hardware requirements: 16k Model A
Reviewer: Sheridan Williams

Another review program with no documentation. It prompts you for several inputs, but does not make it clear whether they are in decimal or hex. There is no mention of where you can place the appropriate *FX calls if you have a serial printer, so you have to enter them directly. For some unfathomable reason the program stops after each run, and has to be restarted with 'RUN'.

The program works, although I did not have time to check all the disassembled codes. Rather Expensive at nearly £6.

MUNCHYMAN Supplied by:Program Power Cost: £5.95 (special offer)
Hardware requirements: 16k Model A
Reviewer: Sheridan Williams

This is a poor relation to Acornsoft's "Snapper", however it does run in 16k. It is fun to play, but what a shame it does not have auto repeat on the keys. I do wish there was a standard set of keys for movement on the screen. The standard cursor keys are OK for left and right, but for up and down they are a pain. Why didn't Acorn/BBC design a proper directional cursor pad? Can I suggest that left right are always either the appropriate cursor keys, or that they are the 'Z' and 'X' keys. For up and down I would suggest the ":" for up, and the "/" for down.

GOMOKU Supplied by:Program Power Cost: £2.95 (special offer)
Hardware requirements: 16k Model A
Reviewer: Sheridan Williams

Gomoku has always been one of my favourite board games. The object is trivially simple - get five crosses in a row in any direction before your opponent. For a game with such simple rules it is amazing how much thought is required to win. This

program plays quite well, and I would class it as good value for money at the price above.

It is not an addictive pub-type game but can provide some stimulating thought.

ELDORADO Supplied by:Program Power Cost: £8.00
Hardware requirements: 32k Model A or B
Reviewer: Sheridan Williams

This is based on the "Adventure" principle. Your mission is stated below, copied exactly as it appears on the screen:

"You are in the county of Eldorado where cowboys are common and Indians terroise (sic) the settlers.
Gold prospectors are common and there is one particular man:
Billy McCusky who is said to have made his fortune where many before have failed. After his unfortunate death a Legend has grown about the whereabouts of his hidden treasure.
Will you be the one to find his hidden treasure ??
When you have found as much treasure as you can, go to the Hotel and type SCORE."

You have to give commands like "TAKE LAMP" or "KILL DONKEY". By using such commands you can explore and hopefully find the treasure.

Another quite fun game that can be recommended.

**********MEMBERS CORNER**********

M. W. Ashby, 7 Linton Rd, Oxford writes:
I have an easy (no components) modification to the "Bush Ranger" (Chassis BM6514 or BM6714) B&W portable that turns it into a very high quality video monitor. Anyone wanting details should write to me.

Chris Melville, 1 Whinway, Albany Village, Washington, Tyne and Wear NE37 1AU (Tel: 0632 462306) asks us to broadcast his interest in using the BBC micro as a basis for a music keyboard controlled synthesiser. He says that he has the 6502 programming capability but is looking for someone to help him on the hardware side.

C.J.Morris of DJB Engineering Ltd, Peterlee, County Durham, SR8 2HX says we wish to become involved in the emulation of VDU terminals that we currently use, and use of the Econet for interconnection of users. If you can help please contact me.

USEFUL ARTICLES

LITERATURE SURVEY

A great deal has been written in the various computing magazines about the Beeb. Some of it, but by no means all of it, has been useful material. We have brought together what we consider to be the best of the bunch, and present it here in the first of what we hope will be a regular feature of our newsletter.

Personal Computer world
Jan 1982 BBC Micro benchtest (review) by Chris Sadler and Sue Eisenbach (5 pages)
Jul 1982 Sound Advice by Mark Holmes (2 pages)(On SOUND & ENVELOPE)
Mapping out the BEEBON by David Christensen (5 pages)
(Many useful operating system addresses.)

Electronics & Computing
Interfacing the BBC Micro, series may, June, July, by Paul Beverley.

Computing Today
Jul 1982 BBC programming by Mike James, useful for screen mapping. (4 pages)

L.A.Dunkling says that he would be grateful for a mention of his professional interest in language-teaching programs. Writers of such programs might care to contact me via - BBC English by Radio and Television, PO Box 76, Bush House, London, WC2B 4PH.

USER PORT PART 2

We continue the article on upgrading and use of the user port, with details of pushbutton and joystick data entry, and with notes on data output.

Push Button Input

The circuit in fig 1 below shows a set of 4 switches or push buttons wired as games paddles or control buttons for the user port. The 4 resistors take the 4 data lines PB7-PB4 to the +5v rail - on line one of the ribbon cable - (this takes them 'high' more reliably than the internal circuitry of the 6522). The switches are each arranged to take one data line to 'low' when pressed or activated. The 4 buttons could be mounted on a small box and used in games and other programs as games paddles (switches with levers and centre-off would be best in this case, when a single pole double throw switch could earth either PB7 or PB6 say, or neither). Fig 2 shows the connection for this, and fig 3 shows how to connect up a standard joystick for digital (rather than analogue) control through the port.

What is now required is a program to sort out which keys have been pressed (or in which direction the joystick lever points) - and the particularly useful point about all this is that any number of keys (up to 8) can be pressed and acted upon simultaneously - which of course is not the case with the keyboard - pressing two keys at once has a disabling effect. The accompanying program achieves this key or switch detection. It initialises the port to "input" in line 130, and the main body of the program is found within the REPEAT..UNTIL loop at line 135. This calls two procedures, one that reads 6522 register zero (which is the data register for port B), and assigns TRUE or FALSE to the variables UP%, DOWN%, LEFT% and RIGHT% depending on the value read in. The second procedure, PROCCURSPOSN moves a cursor in response to the position of the four switches or paddles or joystick connected to PB7-PB4 of the user port as in fig 1. Line 1004, which reads the data at the port, subtracts it from 255 in order to render it more usable. This is done since as things stand, when the 'up' button is pressed the port will read 128 LESS than it did without it being pressed. Subtracting the port value from 255 reverses the effect causing the value of PADDLE% to increase by 128 when the 'up' is detected, which is more convenient. Once the assignments of TRUE or FALSE have been made, the procedure PROCCURSPOSN is used to update the position of the cursor, and draw a solid block at the relevant place. As it stands the previous position is not erased, so the effect is to draw solid lines across the screen under paddle or joystick control. You will notice that if you press LEFT and DOWN simultaneously you will automatically get a diagonal line, since the program responds to each switch quite independently.

The program is meant only to demonstrate a way to incorporate responses to the port into a program, and obviously similar routines can be used in many different implementations. If you require a quicker response, then it will be best to incorporate the two procedures together so that if an 'UP' is detected, the cursor is immediately moved upwards, or at least the cursor control variables X% and Y% are immediately altered correspondingly.

Data Output

Data output is easily achieved on all 8 bits of the user port, by setting the data direction register to 255, and then setting data register B to the value to be output, which again can be any integer value between 0 and 255. If you have a high impedence voltmeter (such as a digital voltmeter) you can immediately see the effect of this. Join the voltmeter between pin 20 of the user port, and earth (say pin 19). Run the 6522 handler program given last month, and put 255 into register 2 of the 6522, then alternately place zero, then 255 into register 0. This will cause the voltage on pin 20 (ie PB7) to go from around 0.1 volts, to around 5 volts, and this programmable control voltage can be used to make the computer control various devices.

Each of the 8 output lines can be individually controlled using data generated by the table opposite. If for example, you wish to send an output (ie set the voltage high) on lines PB7 and PB0, then you just send the value 129(=128+1) to the data register B of the 6522 (ie register 0). This is easily accomplished using the 6522 handler program, or by executing ?&FE60=129 (See notes on PEEK and POKE in issue 2). Again, since the total of all individual values is 255, sending 255 to register 0, will make all 8 output lines go high.

To use these voltages to control external devices you need to buffer, to protect and amplify, the signal. The most straightforward way to do this is to use an I.C. such as a 74LS17 which will amplify 6 lines simultaneously, producing a current large enough to drive a relay from each line. We give connections to the 74LS17 in fig 4. In this case PB7 and PB6 each drive a relay, while PB5 and PB4 light LED indicator lamps. Any relay will do providing it operates from around 10 volts, and requires no more than 30mA to drive it. As may be seen from the diagram, you will need either a separate power supply or battery for this set up to drive the relays. The relay contacts may then be used to switch whatever you require, but take care not to exceed the ratings of the relay contacts, and do NOT contemplate switching mains voltages unless you are fully experienced in this area.

Switching Software

We now require some software to perform the switching. The procedure PROCportoutput achieves this:

```
DEF PROCportoutput
?&FE62=255
?&FE60=ABS(pb7*128+pb6*64+pb5*32+pb4*16+pb3*8+pb2*4+pb1*2+pb0)
ENDPROC
```

Set the variables pb7 to pb0 to zero at the start of the program, then during the program set any of these to 1 (or TRUE) that are required to be switched on, then call the procedure. To turn any bit off subsequently, set the appropriate variable (ie pb7 to pb0) to 0 or to FALSE.

Mixed Output and Input

The data sent to the data direction register at &FE62 is, in fact made up in the same way as the port output data, with a 1 setting a given bit of the port to output, and a zero to input. Thus if you wanted to use bits 0 to 3 for output, and 4 to 7 for input, the value 15 (=1+2+4+8) sent to register two of the 6522 will achieve this.

Conclusion

As we suggested earlier, there is a great deal more that the 6522 will do, including timing and counting, and the provision of programmable control lines. We do not, at present, have time to cover this, and we would refer you to manuals on the 6522, or to books such as L. Scanlon "6502 Software Design" which contains a section on the 6522.

D.E.G.

```
100 REM DRAWING WITH A PADDLE ROUTINE
105 X%=19:Y%=14
110 MODE4
120 PROCCHR
130 ?&FE62=0
135 REPEAT
140 PROCPADPOSN
150 PROCCURSPOSN
160 UNTIL INKEY$(0)<>""
200 MODE7
210 END
1000 DEF PROCPADPOSN
1004 PADDLE%=255-(?&FE60)
1010 UP%=FALSE:DOWN%=FALSE:LEFT%=FALSE
:RIGHT%=FALSE
1020 IF PADDLE% AND 128 THEN UP%=TRUE
1030 IF PADDLE% AND 64 THEN DOWN%=TRUE
1040 IF PADDLE% AND 32 THEN LEFT%=TRUE
1050 IF PADDLE% AND 16 THEN RIGHT%=TRU
E
1060 ENDPROC
1500 DEF PROCCURSPOSN
1510 IF Y%>0 AND UP% THEN Y%=Y%-1
1520 IF Y%<30 AND DOWN% THEN Y%=Y%+1
1530 IF X%>0 AND LEFT% THENX%=X%-1
1540 IF X%<38 AND RIGHT% THEN X%=X%+1
1545 PRINTTAB(X%,Y%)SPC1
1550 PRINTTAB(X%,Y%)CHR$255
1560 ENDPROC
2000 DEF PROCCHR
```

CONTD.

```
2010 VDU23,255,255,255,255,255,255,255,255,255,255
2020 VDU23,0,11;0;0;0;
2030 ENDPROC
```

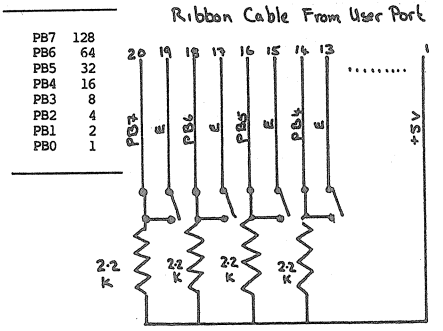


fig 1 Push Button Input

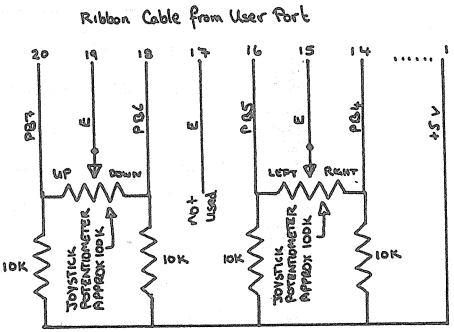
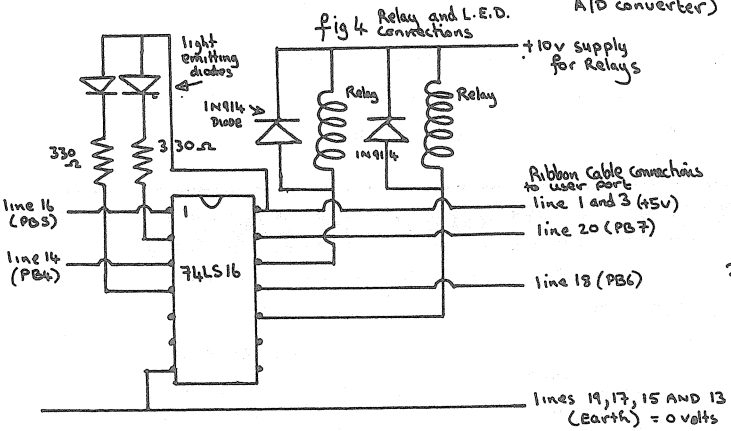


fig 3 Joystick wired for digital input (full analogue input requires use of A/D converter)



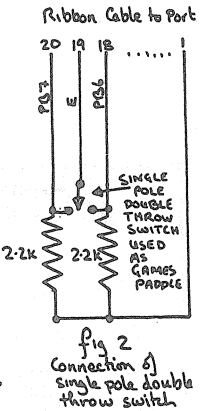


fig 2 Connection of single pole double throw switch

HINTS HINTS HINTS HINTS HINTS

FREE MEMORY

There is no single command in BBC Basic to print the amount of memory remaining such as the variants on FRE(X) to be found in many micros. But the same effect can be achieved by the following sequence:

```
DIM P% -1: PRINT HIMEM-P%
```

This gives the amount of memory remaining after variable storage has been taken into account. This line has been incorporated into the key setting program given elsewhere in this issue.

MICROWARE (LONDON) LTD PRESENT THE "ZL"
RANGE OF DISK DRIVE SUBSYSTEMS
FOR THE BBC MICRO.

Microware

BARE DRIVES FROM ONLY £ 105.00

IN PLASTIC ENCLOSURES £ 135.00

DUAL UNITS WITH OWN P.S.U. £ 295.00

INCLUDES 12 MONTHS WARRANTY ON CASED SUBSYSTEMS

EPSON PRINTERS

EPSON MX80T/2 £ 295.00

EPSON MX80FT/2 £ 325.00

EPSON MX100 £ 495.00

Full range of printers carried in stock. Come and see us for a free demonstration.

PRICES DO NOT INCLUDE POSTAGE AND PACKING OR VAT.

MICROWARE (LONDON) LTD. 637 HOLLOWAY RD. LONDON N19.

PHONE 01 272 6398 FOR FURTHER DETAILS.

ACTIVE COMPONENTS

PRICES DOWN

BBC MICRO UPGRADE KITS

BBC 1	8 x 4816 100ns (IC61-68)	25.50
BBC 2	Printer & User I/O Kit IC69, 70 + PL9,10	8.45
BBC 4	Analogue Input Kit IC73,77 + SK6	6.75
BBC 5	Serial I/O & RGB kit IC74,75 + SK3,4	11.45
BBC 6	Expansion bus & tube kit IC71,72,76 + PL11,12	6.25

BBC CONNECTORS

BBC 21	Connectors for printer compiete	13.00
BBC 22	Connector for user port with 36" cable	2.15
BBC 44	Analogue input plug with cover	2.25
BBC 55	5 pin & 6 pin DIN plugs for Serial I/O & RGB input	0.99
BBC 66	Connector for bus port with 36" cable	3.50

NB As some of the components in these kits are in very short supply, please check for delivery date when ordering.

VISA

24 HOUR TELEPHONE SERVICE FOR CREDIT CARD USERS
☆ All prices exclude VAT and Carriage (0.75 + VAT) on orders under £10 ☆
☆ All orders despatched on day of receipt with full refund on O/S items if requested ☆
☆ Order receipted & returned with goods. If full VAT invoice required please add 0.50 + VAT ☆

ACTIVE COMPONENTS (MAIL ORDER)
Hewitt House, Northgate Street, Bury St. Edmunds, Suffolk IP33 1HQ
Telephone: (0284) 701321 Telex: 817670

BBC BUGS FIX

We publish here an explanation and a fix for the two major bugs in the Beeb's cassette filing system. These affect data storage, and also occasionally render saved programs unreadable. We would stress that we have seized the earliest opportunity to publish this fix, which is in current use at the BBC, but which has not yet been fully tested and approved by Acorn. We thought that you would like to see it as soon as possible. This article is reprinted by permission, from "ESCAPE", the newsletter of the BBC Engineering Society, Home Computing Section. We are most grateful both to the author, and to "ESCAPE" for their permission to publish.

The Cassette Filing System in the BBC Microcomputer (Operating System version 0.10) has a couple of rather unfortunate "bugs" which at best cause annoyance and at worst prevent the use of cassette data files in certain circumstances.

The bugs are as follows:

1. Occasional corruption of the first character of a "block" written to tape. As this first character is the "sync" byte (2A hex) by which the block is recognised, the result is that the block cannot be found when an attempt is made to read it back. This usually gives rise to a "Block?" error message. Nothing can easily be done to recover such a corrupted block.

   In the case of a SAVEd program, the problem generally afflicts only the first block (block 0), subsequent blocks being recorded properly. However, this effectively makes the entire program unreadable as loading is not initiated until block 0 is found. The corruption occurs on a pseudo-random basis and sufficiently infrequently (one in sixteen times on average) that saving a program twice generally gives at least one good copy.

   In the case of data files things are rather worse. Every block is equally likely to be affected so the chance of recording an error-free file is in inverse proportion to its length. Other factors can conspire to make even a relatively short file almost impossible to record without missing blocks.

   This bug occurs only at 1200 baud, not at 300 baud.

2. Loss of characters when a string is written to a cassette data file with the PRINT# statement. This occurs if a data "block" is filled at a critical point in writing out a string (to be precise, after the string marker and length bytes have been sent but before the characters comprising the string have been sent). The result is that the characters of the string are lost and on reading back the file (using INPUT#) a "Type mismatch" error will generally be issued.

   The extent of the problem depends on the number of strings in a file and the overall length of the file. The more strings there are and the longer the file, the greater the chance that the fault will occur. A file consisting of many short strings is the worst case.

   This bug will not occur if the data file is only one block long, and affects only strings written with PRINT#.

It is recognised that although both the above bugs are avoidable by suitable techniques (such as running at 300 baud and writing strings with BPUT#) they may impose unacceptable restrictions on the use of the computer. Therefore some effort has been expended in devising a software patch to correct them. The following program is the result:

```
  100 REM PATCH TO FIX TWO CFS BUGS
  110 REM IN OS EPROM 0.10
  120 REM (C) R.T.RUSSELL, 8-5-82
  130 FORPASS=0TO1:P%=&DD0:GOSUB180:N
EXT
  140 ?&218=FIX1:?&219=FIX1 DIV 256
  150 ?&20A=FIX2:?&20B=FIX2 DIV 256
  160 *KEY10 ?&218=&D0:?&219=&D:?&20A
=&D6:?&20B=&D|M
  170 END
  180 [OPT PASS*2
  190 .FIX1 PHA:JSR &F521:PLA:RTS
  200 .FIX2 CMP #&91:BNE GO:CPX #0:BN
E GO
  210  TSX:LDA &102,X:CMP #&F7:BEQ TR
AP
  220  LDX #0:.TX LDA #&91:STA &FE09:
RTS
  230 .GO JMP (&DB60)
  240 .TRAP PLA:PLA
  250  JSR &F9D8:JSR &FB7B
  260  JSR TX:JMP &F7FB
  270 ]RETURN
```

This program should be CHAINed each time the computer is switched on and after every "full" reset (pressing the BREAK key twice). KEY 10 is set up so that a single press of BREAK will not disable the patch (always assuming that you do not subsequently reprogram KEY 10 yourself). Once run, the above program itself is no longer required; it has loaded a machine-code patch into memory locations DD0 to DFE (hex). These locations are normally unused by BASIC or the Operating System.

The above program inevitably makes absolute references to routines within the Operating System and will work only with OS version 0.10 (EPROM or ROM). It will NOT work with other versions of the Operating System (it will cause the system to crash) so should always be loaded separately rather than be included in other programs.

There is one further, less serious, known bug in the CFS which is worth mentioning. The effect of this is that the EOF# function does not work properly if the data file is an exact multiple of 256 bytes long (an exact number of blocks). In this case the end-of-file condition is not detected until an attempt is made to read past the end of the file; if INPUT# is being used, a "Type mismatch" may result. It is, in any case, usually better to include a "record count" at the beginning of a data file so that EOF# is not needed.

Richard Russell.

HINTS HINTS HINTS HINTS HINTS

M. J. Bates of Winchmore Hill, London writes:
You must try:
*KEY 0 SOUND 2,-15,100,1:SOUND 3,103,100,1|M
Try the key about 6 times and listen in amazement. To get a different sound change the pitch on the SOUND 2 command.

SILLY RENUMBERING

If you RENUMBER 30000 under Operating System 0.1 it does NOT say 'SILLY', but what is worse, if line numbers are taken over 32767 they go over the top, and restart at zero, jumbling your program forever. Thanks to Chris Bingley and others for the note on this bug.

SCREEN DUMP

Several members have sent in programs for dumping the contents of the screen to a dot matrix printer. we have been unable to test these, and present them as received. Tony Goodhew's and Nigel Langley's were supplied on cassette, so there should be no transcription errors here.

Epson MX80T

The following program was sent by A. Goodhew of Peterborough. The program will dump MODE 7 screens to an Epson MX80T printer.

```
100  CLS:PRINT TAB(7)"DUMP7 by Tony Goodhew"'
110  PRINT"A procedure to copy a MODE 7 screen"
120  PRINT" to an EPSON MX80 printer with low"
130  PRINT TAB(7)"resolution graphics."'
140  VDU&91:FORG=160TO191:VDU G:NEXTG:PRINT
150  VDU&95:FORG=224TO255:VDU G:NEXTG:PRINT
160  PRINT':FOR I=33 TO 64:PRINTCHR$(I);:NEXTI:PRINT'''
170  ZZ=1::LL=13:PROCDUMP7:STOP
970    REM ********************* PROCDUMP7 ***********************
980    REM **If ZZ=1 then screen is copied else procedure aborts **
990    REM **Value of LL sets the number of lines to be copied   **
995    REM ************************************************************
1000 DEF PROCDUMP7
1010 LOCAL L,P,PP,X,A$,C
1020 LL=INT(LL):IFZZ<>1 THEN1200
1030 IFLL<1 OR LL>25 THEN1200
1040 FORL=1 TO LL
1050   P=L*40+HIMEM-1:PP=P
1060   X=?P:IFX<>32 THEN1080
1070   P=P-1:IFP<>L*40THEN1060
1080   A$=""
1090   FORC=HIMEM+40*(L-1) TO P
1100     X=?C
1110     A$=A$+CHR$(X):NEXTC
1120   FORC=1 TO LEN(A$)
1130     X=ASC(MID$(A$,C,1))
1140     IF(X<32) OR (X>=128 ANDX<160) THENX=32
1150     IF (X>=91 ANDX<=96) OR (X>=219
AND X<=223) THENX=32
1160     IFX>=224 THENX=X-32
1170     VDU1,X:NEXTC
1180   VDU1,13,1,10
1190   NEXTL
1200 ENDPROC
```

Epson MX80F/T2

Brian Kerr of Teddington, Middlesex sent in this procedure for dumping screens to an Epson MX80 F/T2 printer.

```
DEF PROCscreendump
bcol=0:REM predominant background colour
VDU 1,27,1,51,1,24
FOR Y=1023 TO 0 STEP -32
 VDU 1,27,1,76,1,128,1,2
 FOR X=1 TO 1279 STEP 2
  data=0
  FOR Z=0 TO 7
   IF POINT(X,Y-Z*4)<>bcol THEN data=data+2^(7-Z)
  NEXT Z
  VDU 1:PRINT CHR$(data);
 NEXT X
 VDU 1,10,1,13
NEXT Y
ENDPROC
```

The user must insert the logical colour number of the predominant background colour, then the printer will print a dot for everything but this colour. This will take about 20 minutes to print. what about a machine code routine?

Seikosha GP100A

N.C.J.Langley of Weeping Cross, Stafford writes:

This assembler program is for use in MODE 4. It converts the BBC 320x256 graphics format into a form which the Seikosha GP100A printer (and presumably the GP80A) can read.

After the program has first been run, another program can be written or loaded over it, as it is totally in machine code, held at location HIMEM. The program will only work in MODE 4, there was no point in making it run in any other mode because the printer is not capable of printing the whole screen in a higher mode.

```
 10  MODE 4
 20  HIMEM=HIMEM-300
 30  P%=HIMEM
 40 FOR PASS=0 TO 1
 50 [OPT PASS*3
 60 .start
 70  LDA#&58:STA&71
 80 LDA#&00:STA&70
 90 LDA#1:JSR&FFEE:LDA#8:JSR&FFEE
100 .newline LDA#40:STA&72
110 .newchr LDA#8:STA&73
120 LDX#0:LDY#0
130 .loadchr LDA(&70),Y:STA&74,X
140 INY:INX:CPX#8:BNEloadchr
150 .nextprchr LDA#0:STA&7C
160 LDX#0:LDY#7
170 .printerchr CLC:ROL&74,X:ROR&7C
180 INX:DEY:BNEprinterchr
190 SEC:ROR&7C
200 LDA#1:JSR&FFEE:LDA&7C:JSR&FFEE
210 DEC&73:BNEnextprchr
220 CLC:LDA&70:ADC#8:STA&70:BNE&57A6;o
ldpage
230 INC&71:LDA&71:CMP#&80:BEQ&57B7;end
240 .oldpage DEC&72:BNEnewchr
250 LDA#1:JSR&FFEE:LDA#&0D:JSR&FFEE
260 JMPnewline
270 .end LDA#1:JSR&FFEE:LDA#&0D:JSR&FF
EE:RTS:]
280 NEXT
290 END
```

We would like to thank Terry Bromilow of Glenfield, Leicester, who sent a Basic version of the Seikosha program. We have not given it here because Terry advised us that it was very slow.

INPUT FUNCTION

This month we give an INPUT function. Essentially this handles a major part of the process of data input via the keyboard, and includes partial checks for invalid data. A program is instantly more appealing and professional looking if the questions that require a user response appear fixed on the screen instead of scrolling. It is also nice to be able to disable those keys that are not required, and restrict the number of characters input.

Another important feature that is often overlooked is that of validating (checking) the responses for correctness. For example if the computer asks for your age and you enter 196 you would expect that to be rejected. Similarly if it asked your sex and you replied '1.3' you would expect that to be rejected too.

The INPUT function proposed has the following features:

1. It allows input from anywhere on the screen and positions the cursor there.
2. You can lock out those parts of the keyboard that you don't require. For example you can lock out the alphabetic keys when asking for an age.
3. You can limit the number of characters that are to be accepted.
4. If you want a 'string' answer it can be padded out with trailing spaces to make it a fixed length. (This feature will be useful for direct access files on disc).
5. A null (just 'return' pressed) response can be easily detected.
6. A space of the required length will be cleared on the screen.
7. A beep will sound if a wrong key is pressed.

The function starts at line 1000 (see listings at the end of this article) and is easy to call. Suppose we want to ask someone how many children they have, and we want to display the question 'No of Children' around the centre of the screen. We note that an integer response is required, and that the answer is very unlikely to be more than two digits long, so we use:

```
100 PRINT TAB(5,10);"No of children?"
110 REPEAT
120  nchildren=FNinput(21,10,2,0,"I")
130 UNTIL nchildren>=0 AND NOT NUL
```

There are 5 parameters to this function, they are:
Parameter 1 - The x-coordinate for input.
Parameter 2 - The y-coordinate for input.
Parameter 3 - The number of characters to be input.
Parameter 4 - If a string, then the length of the string to be returned (0 means leave it unpadded).
Parameter 5 - "I", "S", or "N" for integer, string, or numeric respectively.
(See the user guide for details of FN calls and the use of parameters.)

In order to lock out any keys that you don't want used, place their ASCII codes between lines 1220 and 1230 of the procedure definition, and then GOTO line 1119. For example, if (for some reason) you don't want a comma to be used, then insert 1222 IF ch_code=44 THEN 1119

If you don't like the error beep them simply replace line 1119 with 1119 REM

In specific areas of computing you get libraries of procedures, you should get used to using a procedure even if you don't understand how it works. As long as you have a description of the input parameters and the procedure's effect, you should be able to use it.

The first program listed is one that demonstrates 'Integer' and 'String' input, and has good screen handling. It calls the INPUT function listed. Note how if you just press 'return' to any field it hops back to the previous field. The only real way to see just how useful this function is, is to try it, so go ahead. The function itself is given after the demonstration program (ie lines 1000 to 1320). S.W.

```
  10 REM To test PROCinput
  20 CLS
  30 PROCcentre(2,40,"P E R S O N A L
  D E T A I L S")
  40 PROCcentre(3,40,"----------------
----------------")
  50 PRINT TAB(15,6);"NAME:"
  60 PRINT TAB(15,7);"----"
  70 PRINT TAB(2,10);"SEX:"
  80 PRINT TAB(2,11);"---"
  90 PRINT TAB(30,10);"AGE:"
 100 PRINT TAB(30,11);"---"
 190 REPEAT
 200 name$=FNinput(20,6,15,0,"S")
 210 UNTIL NOT NUL
 290 REPEAT
 300 sex$=FNinput(6,10, 6,1,"S")
 310 UNTIL sex$="M" OR sex$="F" OR NUL
 315 IF NUL THEN 190
 320 PRINT TAB(6,10);:IF sex$="M" THEN
PRINT"Male  " ELSE PRINT"Female"
 390 REPEAT
 400 age=FNinput(34,10,2,2,"I")
 410 UNTIL age>=0
 420 IF NUL THEN 290
 900 PRINT''''
 910 PRINT"Details entered correctly"
 999 END
```

```
1000 DEF FNinput(X%,Y%,max_length%,pad_
length%,Type$)
1010 REM-------------------------------
1020 REM Replacement for INPUT
1030 REM Set max_length=max no ch
1035 REM                to be input
1040 REM pad_length is length of
1042 REM      string to be returned.
1050 REM Set Type$="S" for strings
1052 REM         or "N" for numeric
1054 REM         or "I" for integer
1058 REM If Type$="S" then the result
1060 REM will be padded to pad_length
1064 REM          if pad_length>0
1070 REM NUL is set to TRUE if no
1072 REM   characters are input
1080 REM X%,Y% are screen addresses
1085 REM for the cursor position
1090 REM-------------------------------
1095 LOCAL input_string$,ch_code,Length
,no_point,ch$
1099 REM-------------------------------
--------------
1100 input_string$="":Length=0:no_point
=(Type$="N")
1105 *FX 4,1
1110 PRINT TAB(X%,Y%);STRING$(max_lengt
h%," ")+STRING$(max_length%,CHR$(127));:
GOTO 1120
1119 SOUND 1,-10,100,2
1120 ch$=GET$:ch_code=ASC(ch$):IF ch_co
de=13 THEN 1310
1160 IF ch_code=127 THEN 1250
1170 IF Length=max_length% THEN 1119
1180 IF Type$="S" THEN 1220
1190 IF Length=0 AND ch$="-" THEN 1230
1200 IF Type$="N" AND no_point AND ch$=
"." THEN no_point=FALSE:GOTO 1230
```

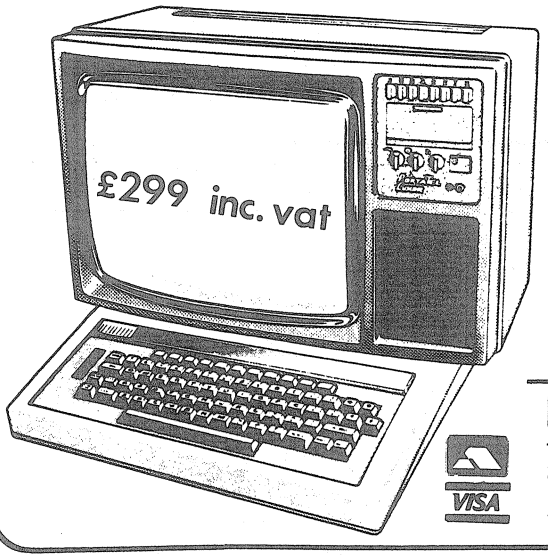
```
 1210 IF ch_code<48 OR ch_code>57 THEN 1
119
 1220 REM Place any ch to be ignored her
e then:GOTO 1119
 1221 IF ch_code<32 OR ch_code>97 THEN 1
119
 1229 REM End of ch to be ignored
 1230 input_string$=input_string$+ch$:Le
ngth=Length+1
 1240 PRINT ch$;:GOTO 1120
 1245 REM Delete pressed so remove ch
 1246 REM------------------------------
 1250 IF input_string$="" THEN 1119
 1255 IF RIGHT$(input_string$,1)="." THE
N no_point=TRUE
 1260 Length=Length-1:input_string$=LEFT
$(input_string$,Length):GOTO 1240
 1300 REM Exit procedure
 1301 REM----------------
 1310 *FX 4,0
 1315 IF pad_length% THEN input_string$=
LEFT$(input_string$+STRING$(pad_length%,
" "),pad_length%)
 1320 NUL=(Length=0):PRINT " ":IF Type$=
"S" THEN =input_string$ ELSE =VAL(input_
string$)
```

HINTS HINTS HINTS HINTS HINTS

RANDOMISE

In some Basics there is a function called RANDOMISE whose purpose is to set the pseudo random number generator called up by the RND function to a random point in its sequence. The reason for this is that every time you switch on the computer from cold, the computer will begin its random number generator at the same point. This means that with a game, say, the machine will choose the same strategy after every cold start. To avoid this, simply execute the following statement X=RND(-TIME) at the start of the program. This 'seeds' the random number generator with a value that depends on the state of the Beeb's internal clock; and this will be different every time (unless you set the clock in the line above!).

Timeshare your Colour Monitor with the Family



COLOUR TV
Plus RGB
Plus PAL VIDEO

BBC Micro lead included
Excellent resolution, geometry

PortaTel LUXOR 14" Colour Monitor
Model No. RGB3711 TV Receiver

PortaTel conversions limited
television and electronic engineers
25, sunbury cross centre, staines road west,
sunbury-on-thames, middlesex. tw16 7bb
telephone: sunbury-on-thames 88972

PROGRAM TRANSFER

In this short article we give a routine that allows data transfer in both directions between a BBC machine and an RML 380Z. The RS423 port of the Beeb is used for this purpose, and the software includes a machine code patch to effect the input of data to the Beeb.

This transfer facility can be extremely useful for many purposes, allowing existing 380Z programs to be downloaded onto the Beeb. It should be possible to use the principles involved with other machines, especially those that use the CP/M operating system.

The following routine was supplied by Peter Andrews of the "Chiltern Region Advisory Unit for Computer Based Education". The Director is W. Tagg Ph.D. M.B.C.S. and the unit is based at Endymion Road, Hatfield, Herts, AL10 8AU.

How to transfer files between the BBC Micro and the RML 380Z

Hardware needs

A special cable is needed to link the two computers. This link is between the SIO-4 port of the 380Z and the RS423 port of the BBC (Model B). Refer to the BBC User Guide for pin connections on the RS423 port. Make the connections as shown in Table 1. It is also necessary to connect a 1 kohm resistor between the CTS and RTS pins. It may be easier to house this in the 25-pin plug at the 380Z end rather than in the small DIN plug at the BBC end.

BBC	380Z
Data in	Pin 3
Data out	Pin 2
RTS	Pin 4
0 volts	Pin 7

Table 1

Software needs

1. Special software has been prepared for use in the BBC machine, please save this and refer to it as "BBCXFR", it is listed after these notes. The program enables transfer in either direction by use of four function keys (f6-f9). When the program is run it gives a brief description of how to use the keys. Once run, any number of program transfers may be undertaken without the need to run the program again; providing the function keys are not reprogrammed.
2. No special software is needed in the 380Z to send programs to the BBC machine. All that is necessary is to assume that the BBC machine is a printer operating at 300 baud (PRINTER 4,1 command in Basic on the 380Z). (Note programs from the 380Z must be sent as ASCII text, not in internal format). If a transfer is aborted, you may experience difficulty in starting a new transfer because of junk data left on the port. This condition can be cleared by switching off the BBC machine and starting again.
3. The easiest way to send to the 380Z is to patch PIP to accept data from the SIO-4 port. Follow exactly the procedure described on page 3.14 of "380Z Disc System Information File". The PIP command is then "filename=INP:". The transfer rate in this direction is 2400 baud (PRINTER 4,4 on the 380Z).
4. Note that the transfer can be made from any 380Z (disc or cassette) to the BBC machine, but if using the above method, transfer is only to a disc based 380Z. [Although RML can supply a configured version of Basic allowing data to be passed to a cassette system from the serial port. Ed.]

Peter Andrews

We have used this program at BEEBUG for over a month now and it works successfully. For those who do not know the specification of the RML 380Z it should be a very simple matter to use this routine to transfer to and from any CP/M based machine. You may need to use the CP/M utility "STAT" to change the 'LST' and 'RDR' (or 'INP') devices. I shall be trying this using a North Star Horizon during August, and will report my findings in a later issue.

```
   1 REM ** TRANSFERRED FROM 380Z **
 2000 REM ********
 2010 REM SET UP FOR RECEIVING
 2020 *KEY 8 "NEW|M| REM ** TRANSFERR
ED FROM 380Z **|M?&0210=&00:?&0211=&0
D:VDU 15:CLS|M"
 2030 *KEY 9 "OLD|M|NLIST|M"
 2040 *FX 7,3
 2050 P%=&0D00
 2060 [
 2070 OPT 0
 2080 .LOOP
 2090 LDA #&01
 2100 BIT &FE08
 2110 BEQ LOOP
 2120 LDA &FE09
 2130 RTS
 2140 ]
 2150 REM ********************
 2160 REM SET UP FOR SENDING
 2170 *KEY 6 "VDU 15:CLS:VDU 21|M*FX6
,10|M*FX5,2|M*FX8,5|MLIST|F|B|M"
 2180 *KEY 7 "|A|Z|M"
 2190 CLS:PRINT'''
 2200 PRINT "  **********************
***********"
 2210 PRINT "  * TRANSFER BETWEEN BBC
 AND 380Z *"
 2220 PRINT "  **********************
***********"
 2230 PRINT:PRINT"To send to 380Z -"
 2240 PRINT"   Set 380Z printer speed
 to 4,4":PRINT"   Use PPIP and specif
y input as INP:"
 2250 PRINT "   To start transfer, pr
ess f6"
 2260 PRINT"   Press f7 when transfer
 complete":PRINT
 2270 PRINT"To receive program from 3
80Z -"
 2280 PRINT"   Press f8"
 2290 PRINT"   Set 380Z printer speed
 to 4,1"
 2300 PRINT"   LLIST or PIP the progr
am to LST:"
 2310 PRINT"   When complete, press B
REAK then f9"'''
>LOA
```

STRING HANDLING TIP

We were sent this tip by a member, but we have regrettably mislaid his/her name. If they would like to write to us we will publish an acknowledgement next issue.

BBC Basic, like other Basics, has operations to manipulate strings. You probably take for granted the fact that string variables in a Basic program may grow and shrink in length as the program executes. If you've ever wondered how the interpreter manages this I can reveal all. It manages it not very well.

To demonstrate this, run the following short program and see what happens:

```
10 CLEAR:B=!2 AND &FFFF
20 A$="":FOR I=1 TO 255:A$=A$+"A":NEXT
30 PRINT"Memory used=";(!2 AND $FFFF)-B
40 CLEAR:B=!2 AND &FFFF
50 A$=STRING$(255,"?")
60 A$="":FOR I=1 TO 255:A$=A$+"A":NEXT
70 PRINT"Memory used=";(!2 AND $FFFF)-B
80 END
```

It will print: 'Memory used=3890', followed by 'Memory used=271'

The FOR loop in line 20 has used more than 14 times as much memory as the identical loop in line 60. The reason for this becomes clear once you understand how the interpreter manages the memory for strings. It does this as follows:

1. The first occurrence of the string variable name will make an entry for the name in the symbol table, but no buffer is allocated.
2. When the string is given a value, a buffer is allocated the length of the string + 8 bytes.
3. Subsequently, when the string is given a new value, if the new length is less than or equal to the old (+8) then the old allocated buffer is used.
4. If the new string is longer, then the previously allocated buffer is discarded, and a new longer buffer is allocated as in (2).

The large amount of memory used by the FOR loop in line 20 is a consequence of (4) above. The string A$ in the program is growing by 1 character every time round the loop, every 8 times around the loop the string becomes too big to fit its allocated buffer. By the end of the loop there are lots of discarded partial strings, wasting valuable memory.

The following programming tip will minimise this waste, and is used in line 50

of the program:

Force the interpreter to allocate enough space for the maximum length that you think your strings will reach. Do this as early in the program as possible. You can do this by saying:

```
A$=STRING$(length,"?"):REM allocate the space
A$=""                 :REM Empty the space
```

(See the 'User Guide' for details of the STRING$ function.)
Provided that you do not overflow 'length' your string will not waste any more memory.

BEEBUG HAMS

Many members are using their micros in connection with their hobbies. One of the many hobbies where computers can be of major help is amateur radio. Here is a list of members who are also 'Hams'.

HAM LIST

John R. Bird	G3LBW	Middlesborough	David Overton	G4EWE	Cheshire
Peter Burden	G3UBX	Wolverhampton	Alan Pemberton	G8ZHG	Sheffield
Pete Cockerell	G6CSZ	Essex	N. Pilling	G4LYE	Yorkshire
Roger Cooke	G3LDI	Wymondham,Norfolk	Mike Potter	G8ZQO	Cambridgeshire
K. Gee	GM4LNN	Orkney Islands	John Ray	G8DZH	Loughton, Essex
S.W. Edwards	G8TMI	Warwick	David Sandy	G6EKV	Norwich, Norfolk
A.J.Hope	G4INL	Gloucestershire	R. C. Sterry	G4BLT	West Yorkshire
P. Knight	G3HGR	Sevenoaks, Kent	R. C. Taylor	GW2HCJ	Wales
J. Lauder	GM3PZG	Aberdeenshire	Roger Tipper	G4KXR	Exeter
J.Mackay	G4LWM	Preston	John Yale	G3ZTY	Dorset.
David Norris	G8HUP	Merseyside			

Space is always at a premium in the newsletter, and because of this we will only be printing the 'ham list' on an irregular basis. If you would like to be included on it please send details on a postcard.

HINTS **HINTS** **HINTS** **HINTS** **HINTS**

CASSETTE LOAD AND SAVE PROBLEMS

Several members, including Ian Robarts, P H Gosmark and H Brook, have experienced particular difficulty using the Ferguson 3T07 cassette recorder (in spite of Ian Robart's comment that Acorn had recommended this very model). The problem, as H Brook discovered, is that the signal level from the Beeb is too great, and overloads the recorder's input circuitry. The solution is a simple one. Mr Brook has wired a resistor in series with the lead from the computer to the recorder (ie the lead going to pins 1 and/or 4). The value, he says should be between 4 and 10 kilohms,'the object being,' he says,'to get a tape that loads at about 60% of full volume setting.' In his case, the best value was found to be 5.6 kilohms.

Brian Carroll confirms these findings with regard to another make of recorder. He has a Sanyo recorder which is severely overloaded with the output from the Beeb. His solution is similar, and uses a 0.2x attenuator. This is achieved by placing a 47 kilohm resistor between the output of the Beeb (pin 4) and the recorder input, and a 12 kilohm resistor across the input of the tape recorder (ie between the input and ground or chassis).

Earth loops can also cause a problem when connecting up separate earth returns for record and playback leads. If this does occur you may notice a considerable hum on the recording. The problem can be resolved by just using a single earth lead, or temporarily disconnecting the playback lead during recording, and vice versa.

POINTS ARISING

Happily we do not appear to have made any gross blunders in the last issue, but there are a couple of points worth raising under this rubric.

Interlacing

On page 16 of the June issue we have the order of the a and b parameters of the *TV a,b command interchanged (though the syntax is OK in the rest of the article).

Screen Map

In the June issue we mentioned the redundancy in the memory mapping on Model A, which meant that if you were writing to the screen in say MODE 7, you could either write to locations between &7C00 and &7FE8, or to those between &3C00 and &3FE8. There is no problem here, except that if you require your programs to work in a 32k machine as well as in 16k machines, then it is essential to use the higher set, which are fully compatible. Another way round this is to use the pseudo-variable HIMEM, since HIMEM is normally &7C00 on a model B and &3C00 on a model A. Thus in mode 7 the command ?(HIMEM+42)=255 will place a solid block (ie character 255) at a position 42 spaces from the top left hand corner of the screen (and since there are 40 characters to a line, the character will appear at the third position from the left in the second line). Note that when you scroll the screen, the locations (unexpectedly) change. We are grateful to Tim Durham for these notes.

*FX 4,2

Some members have used this call to enable the cursor keys to be read with INKEY or GET (their character values are: copy=135, left=136, right=137, down=138, up=139). This appears to work on OS 0.1, but will NOT have the same effect on OS 1.0. The proper call to enable these keys to be read in this way is *FX 4,1 (which works whichever OS you have). *FX 4,0 returns them to normal. The problem arises because the command *FX 4,2 is used in the 1.0 OS to enable the cursor keys to be used as additional user-defined keys. On OS 0.1 this is not implemented, and the machine behaves as if the call were *FX 4,1.

HINTS HINTS HINTS HINTS HINTS

R.T.Russell of Gravesend, Kent has supplied the following: (Mr Russell is the person whose name appears on the Technical Specification of the BBC Micro)

*TV 0,1 which switches the interlacing off will not work in MODE 7, because mode 7 uses "character rounding" (which is why the characters are nicer than in the other modes) which works only because the vertical resolution is effectively doubled (500 TV lines rather than 250) if both fields of an interlaced picture are considered separately.

*CAT will not notice if the header of block zero is not recognised from the first program on a tape.

All model B machines are fully equipped in hardware terms to receive RS423, but the 0.1 O.S. provides no software driver. A GET from the RS423 can be provided by using the following program:

```
100 DIM P% 10          160 LDA &FE09
110 [OPT 2             170 RTS :]
120 .RS423             180 *FX 7,n
130 LDA &FE08          190 REPEAT
140 AND #1             200  VDU USR(RS423) AND 255
150 BEQ RS423          210 UNTIL FALSE
```

The last three lines demonstrate the use of the driver to copy RS423 input to the screen.

CHUNKY INVADERS

(16k)

by

Robin Whitehead

This is a nice rendering of the popular arcade game in Mode 7 (Teletext) graphics. The reason for using mode 7 is to keep the byte count low, and the program will in fact run on a model A without a squeeze. The whole program is written in Basic, and although not as fast as traditional machine code versions, the screen handling is still reasonably rapid. Considerable use is made of procedures, (although not all of these are defined at the end of the program), so that translation of some of the more critical ones might be possible in order to speed things up. Also, use of the envelope control might enable the sound effects to be improved. Full instructions are included in the program.

```
  10 *TV 255
  20 VDU 23;8202;0;0;0;
  30 *FX 11,1
  40 SCORE=0:H%=0:F1=0:F2=0
  50 DIM A$(20):DIM B$(20):DIM C$(20
):DIM D$(20):DIM B1(3):DIM B2(3):DIM
B4(10):DIM AA$(3)
  60 IF X%=89 GOTO 320 ELSE Y%=0:GOT
0 200
  70DEF PROCBASE
  80Z%=INKEY(0)
  90M%=0
 100IF Z%=90 M%=-1:GOTO 140
 110IF Z%=88 M%=1:GOTO 140
 120 IF Z%=32 PROCFIRE:GOTO 180
 130 GOTO 190
 140N=N+M%
 150IF N=36 N=N-1
 160IF N<1 N=N+1
 170PRINTTAB(0,23)CHR$(&96);TAB(N,23
)" ~} "
 180*FX 15,1
 190ENDPROC
 200 MODE 7
 210 REM SPACE INVADERS - R. WHITEHE
AD -          JUNE 1982
 220 PRINTTAB(10,2)CHR$(141);CHR$(&8
3)"BEEB INVADERS"
 230 PRINTTAB(10,3)CHR$(141);CHR$(&8
3)"BEEB INVADERS"
 240 PRINTTAB(9,6)CHR$(&95)"f{w9";CH
R$(&81)"  =  MYSTERY POINTS"
 250 PRINTTAB(10,8)CHR$(&94)"91";CHR
$(&84)"   =  100 POINTS"
 260 PRINTTAB(10,10)CHR$(&93)"f9";CH
R$(&83)"   =   80 POINTS"
 270 PRINTTAB(10,12)CHR$(&92)"ny";CH
R$(&82)"   =   60 POINTS"
 280 PRINTTAB(10,14)CHR$(&91)"~}";CH
R$(&81)"   =   40 POINTS"
 290 PRINTTAB(10,16)"Z KEY = LEFT";T
AB(10,18)"X KEY = RIGHT"
 300 PRINTTAB(10,20)"SPACE BAR TO FI
RE LASER"
 310 ST=TIME:REPEAT UNTIL TIME=ST+50
0
 320 CLS:PRINTTAB(10,0)SCORE:AA$(1)=
"~}":AA$(2)="~}":AA$(3)="~}"
 330 PRINTTAB(0,0)CHR$(&96);AA$(1);"
 "AA$(2);" "AA$(3);CHR$(&87);TAB(22,0
)"HIGH SCORE = ";Y%
 340 ST=TIME:GOTO 1000
 350DEF PROCBOMB
 360 IF B1(1)+B1(2) <> 0 PROCDROP:EN
DPROC
 370 FOR J%=1 TO RND(2)
 380 B1(J%) = RND(10)
 390 IF B4(B1(J%))=4 GOTO 500
 400 IF DL < 10 GOTO 440
 410 IF CL < 10 GOTO 450
 420 IF BL < 10 GOTO 460
 430 GOTO 470
 440 IF D$(B1(J%)) <> "  " B2(J%)=H+
7:GOTO 480
 450 IF C$(B1(J%)) <> "  " B2(J%)=H+
5:GOTO 480
 460 IF B$(B1(J%)) <> "  " B2(J%)=H+
3:GOTO 480
 470B2(J%)=H+1
 480B1(J%)=(B1(J%)*2)+I
 490 B3=0
 500NEXT
 510PROCDROP
 520 ENDPROC
 530DEF PROCDROP
 540 FOR J%=1 TO 2
 550 IF B1(J%)=0 GOTO 660
 560 IF B3=0 B3=B3+1:GOTO 590
 570 IF B3=1 AND B1(2) > 0 B3=B3+1:G
OTO 590
 580 PRINTTAB(B1(J%),B2(J%)-1)" "
 590 X%=FNHIT(B1(J%),B2(J%))
```

```
  600 IF B2(J%) < 23 GOTO 620
  610 IF X% <> 0 AND X% <> 32 PROCGOT
YOU:PROCBASE:GOTO 650 ELSE GOTO 650
  620  IF X%=0 OR X%=32 PRINTTAB(B1(J
%),B2(J%))"¦":B2(J%)=B2(J%)+1:GOTO 66
0
  630 IF X%=94 PRINTTAB(B1(J%),B2(J%)
)"  ":F1=0:GOTO 650
  640 IF X%=255 PRINTTAB(B1(J%),B2(J%
))"a" ELSE PRINTTAB(B1(J%),B2(J%))" "
  650B1(J%)=0
  660NEXT
  670 ENDPROC
  680 DEF PROCGOTYOU
  690 SOUND0,-10,14,15
  700 PRINTTAB(B1(J%)-1,23)CHR$(&91)"
~}"
  710 NT=TIME:REPEAT UNTIL TIME=NT+10
0
  720 LIVE=LIVE+1
  730 IF LIVE < 3 GOTO 810
  740 CLS:PRINTTAB(10,10)"GAME OVER"T
AB(10,12)"YOUR SCORE =  ";SCORE
  750IF SCORE<Y% PRINTTAB(10,14)"HIGH
 SCORE =  ";Y% ELSE PRINTTAB(10,14)"T
HIS IS THE HIGH SCORE":Y%=SCORE
  760 PRINTTAB(10,18)"ANOTHER GAME?":
X%=0:X%=GET:IF X%=89 RUN
  770 IF X% <> 78 THEN 740
  780 CLS
  790 *FX 11,15
  800 END
  810 AA$(LIVE)="  "
  820 PRINTTAB(0,0)CHR$(&96)AA$(1)" "
;AA$(2)" ";AA$(3)
  830 PRINTTAB(N,23)"    "
  840 N=1:PRINTTAB(0,23)CHR$(&96)" ~}
 "
  850 IF F1>0 PRINTTAB(F1,F2)" "
  860 F1=0
  870 ENDPROC
  880 DEF PROCL1
  890 PRINTTAB(0,H)CHR$(&94);TAB(I,H)
"  ";A$(A%+1);" ";A$(A%+2);" ";A$(A%+
3);" ";A$(A%+4);" ";A$(A%+5);" ";A$(A
%+6);" ";A$(A%+7);" ";A$(A%+8);" ";A$
(A%+9);" ";A$(A%+10);" "
  900ENDPROC
  910 DEF PROCL2
  920 PRINTTAB(0,H+2)CHR$(&93);TAB(I,
H+2)"  ";B$(A%+1);" ";B$(A%+2);" ";B$
(A%+3);" ";B$(A%+4);" ";B$(A%+5);" ";
B$(A%+6);" ";B$(A%+7);" ";B$(A%+8);"
";B$(A%+9);" ";B$(A%+10);" "
  930ENDPROC
  940DEF PROCL3
  950 PRINTTAB(0,H+4)CHR$(&92);TAB(I,
H+4)"  ";C$(A%+1);" ";C$(A%+2);" ";C$
(A%+3);" ";C$(A%+4);" ";C$(A%+5);" ";
C$(A%+6);" ";C$(A%+7);" ";C$(A%+8);"
";C$(A%+9);" ";C$(A%+10);" "
  960ENDPROC
  970DEF PROCL4
  980 PRINTTAB(0,H+6)CHR$(&91);TAB(I,
H+6)"  ";D$(A%+1);" ";D$(A%+2);" ";D$
(A%+3);" ";D$(A%+4);" ";D$(A%+5);" ";
D$(A%+6);" ";D$(A%+7);" ";D$(A%+8);"
";D$(A%+9);" ";D$(A%+10);" "
  990ENDPROC
 1000 N=0:AL=0:BL=0:CL=0:DL=0
 1010FOR I%=1 TO 10
 1020A$(I%)="n$"
 1030NEXT
 1040FOR I%=11 TO 20
 1050A$(I%)="91"
 1060NEXT
 1070FOR I%=1 TO 10
 1080B$(I%)="f9"
 1090NEXT
 1100FOR I%=11 TO 20
 1110B$(I%)="."
 1120NEXT
 1130FOR I%=1 TO 10
 1140C$(I%)="ny"
 1150NEXT
 1160FOR I%=11 TO 20
 1170C$(I%)="v="
 1180NEXT
 1190FOR I%=1 TO 10
 1200D$(I%)="~}"
 1210NEXT
 1220FOR I%=11 TO 20
 1230D$(I%)=">m"
 1240NEXT
 1250 A=0:B=0:C=0:A%=0:H=2+C%:F1=0
 1260  PRINTTAB(0,20)CHR$(&92);
 1270 FOR I%=1 TO 4
 1280  PRINT" x"CHR$(&FF)CHR$(&FF)CHR
$(&FF)"t    ";
 1290 NEXTI%
 1300 PRINTTAB(0,21)CHR$(&92);
 1310 FOR I%=1 TO 4
 1320 PRINT" "CHR$(&FF)CHR$(&FF)" "CH
R$(&FF)CHR$(&FF)"    ";
 1330 NEXT I%
 1340 PRINTTAB(0,23)CHR$(&96)" ~} "
 1350 PROCA
 1360FOR I=A TO B STEP C
 1370 IF TIME-ST >= 500 PROCMSHIP
 1380 IF A%=20 A%=0
 1390 IF H%>2 AND H%<33 PROCMSHIP
 1400 IF AL=20 GOTO 1430
 1410 PROCL1
 1420 PROCBASE
 1430 IF BL=20 GOTO 1470
 1440 PROCL2
 1450 PROCBASE
 1460 IF H%>2 AND H%<33 PROCMSHIP
```

```
1470 IF CL=20 GOTO 1500
1480 PROCL3
1490 PROCBASE
1500 IF DL=20 GOTO 1530
1510 PROCL4
1520 PROCBASE
1530 PROCBASE
1540 PROCHIT
1550 NT=TIME+(40-(C*10))
1560 A%=A%+10
1570 SOUND 3,-10,A%+1,4
1580 PROCHIT
1590 PROCBASE
1600 REPEAT PROCBASE
1610 PROCHIT
1620 UNTIL TIME>=NT
1630NEXTI
1640 GOTO 1350
1650 DEF PROCA
1660 IF B=8 A=8:B=1:C=-1 ELSE B=8:A=
1:C=1
1670 FOR J%=0 TO 6 STEP 2
1680 IF J%=0 AND AL=20 GOTO 1730
1690 IF J%=2 AND BL=20 GOTO 1730
1700 IF J%=4 AND CL=20 GOTO 1730
1710 IF J%=6 AND DL=20 GOTO 1730
1720 PRINTTAB(1,H+J%)SPC44
1730 NEXT
1740 H=H+1
1750IF DL < 20 AND H=14 GOTO 1790
1760IF CL < 20 AND H=16 GOTO 1790
1770IF BL < 20 AND H=18 GOTO 1790
1780IF H < 20 GOTO 1800
1790J%=1:B1(J%)=N:PROCGOTYOU:PROCGOT
YOU:PROCGOTYOU
1800 ENDPROC
1810DEF PROCFIRE
1820 IF F1 > 0 GOTO 1850
1830 F1=N+1
1840 F2=22
1850 PROCHIT
1860 IF F1 = 0 GOTO 1880
1870 PRINTTAB(F1,F2)"^"
1880ENDPROC
1890 DEF PROCHIT
1900 PROCBOMB
1910 IF F1=0 GOTO 1970
1920 PRINTTAB(F1,F2)" "
1930 F2=F2-1
1940 PROCHITA
1950 IF F2 < 1 F1=0:GOTO 1970
1960 IF F1 > 0 PRINTTAB(F1,F2)"^"
1970 ENDPROC
1980DEF PROCSCORE
1990 SCORE = SCORE+((10-(F2-H))*10)
2000 PRINTTAB(10,0)SCORE
2010 F3=(F1-I)+1
2020 F3=INT(F3/3)
2030 B4(F3)=B4(F3)+1
2040 L%=F2-H
2050 FOR J%=1 TO 2
2060 IF L%=6 GOTO 2120
2070 IF L%=4 GOTO 2110
2080 IF L%=2 GOTO 2100
2090 A$(F3)="  ":AL=AL+1:GOTO 2130
2100 B$(F3)="  ":BL=BL+1:GOTO 2130
2110 C$(F3)="  ":CL=CL+1:GOTO 2130
2120 D$(F3)="  ":DL=DL+1
2130 F3=F3+10
2140 NEXT J%
2150 IF AL+BL+CL+DL=80 C%=C%+1:CLS:P
RINTTAB(10,10)CHR$(141);CHR$(&81)C%*1
00;" POINTS":PRINTTAB(10,11)CHR$(141)
;CHR$(&81)C%*100;" POINTS":SCORE=SCOR
E+C%*100:NT=TIME:REPEAT UNTIL TIME=NT
+350:X%=SCORE
2160 IF AL+BL+CL+DL=80 CLEAR:SCORE=X
%:X%=89:GOTO 50
2170 F1=0
2180 ENDPROC
2190 DEF FNHIT(ch,l)
2200 =?(&7C00+(l*40)+ch)
2210 DEF PROCHITA
2220 X%=FNHIT(F1,F2)
2230 IF X%=0 OR X%=32 ENDPROC
2240 IF F2 < 20 GOTO 2270
2250 SOUND0,-10,0,1
2260 IF X% = 255 PRINTTAB(0,F2)CHR$(
&92)TAB(F1,F2)"2":F1=0:ENDPROC ELSE P
RINTTAB(F1,F2)" ":F1=0:ENDPROC
2270 IF X%=124 PRINTTAB(F1,F2)" ":F1
=0:ENDPROC
2280 IF F2=1 PROCMHIT:GOTO 2330
2290 IF F2=0 F1=0:GOTO 2330
2300 PRINTTAB(F1,F2)"   "
2310 SOUND0,-10,F2-H,5
2320 PROCSCORE
2330ENDPROC
2340DEF PROCMSHIP
2350IF H%>1 AND H%<33 GOTO 2380 ELSE
I%=RND(2)
2360IF I%=1 I%=-2:H%=33 ELSE H%=1
2370 ST=0
2380 PRINTTAB(0,1)CHR$(&95)TAB(H%,1)
"  f{w9  "
2390 H%=H%+I%
2400 IF H%=1 OR H%>= 33 ST=TIME:PRIN
TTAB(0,1)"
          "
2410 ENDPROC
2420DEF PROCMHIT
2430 FORI%=0 TO 555  STEP 7:SOUND1,-
15,I%,0:NEXT
2440 SCORE=SCORE+(RND(3)*100)
2450 PRINTTAB(9,0)CHR$(&86)SCORE;CHR
$(&87)
2460 PRINTTAB(H%,1)"      "
2470 ENDPROC
```

DISCOUNTS

BEEBUG have arranged discounts for members at a number of retail outlets who supply computer books, software, hardware and services. we are continually negotiating further discounts.

We have heard that some suppliers are selling 120ns 4816 memory chips for Beeb upgrades. The Acorn specification is for a 100ns device (eg 4816-3), though we have not heard of anyone having problems with 120ns chips.

Active Components Ltd (Incorporates Midwich)
Hewitt House
Northgate Street
Bury St Edmunds
Suffolk (Tel: 0284 701321)
Hardware 10% discount on all items

Electronics Applied
4 Dromore Road
Carrickfergus
Co Antrim
BT38 7PJ (Tel: 65973)
Cassette leads for the BBC machine 5% discount

Happy Memories
Gladestry
Kington
Herefordshire
HR5 3NY (Tel: 054422 618)
Computer Hardware 10% off all 'one-off' prices. Quantity prices may be negotiable.

Mine of Information
(Mail Order)
1 Francis Avenue
St Albans
Herts (Tel: 0727 52801)
Computer Books 5% discount

Opus Supplies
Birchtrees
10 Beckenham Grove
Shortlands
Bromley Bh2 OJU
Computer furniture 20% discount. Send for brochure.

Technomatics Ltd
15 Burnley Road
LONDON Nw10
(Tel: 01-452 1500)
Hardware, software and books, 5% discou

Wallace J & S
9 Barn Close
Crewkerne
TA18 8BL
Flexible plastic dus covers. £3.95. 25% discount to members.

Watford Electronics
33, Cardiff Road
watford
Herts.
(Tel: watford 40588)
5% off all items, except special offer mentioned elsewhere

It is advisable to telephone before placing an order, to check availability. Members should simply quote their membership number with their order, though members taking discount will not necessarily be given credit card facilities, (you must check this). we are not acting for these companies, nor receiving payment from them, and cannot be held responsible for their services.

LOCAL USER GROUP INDEX

BEEBUG is happy to act as an information point for local groups. If you would like to be in our index, just drop a line to us and mark the envelope "Local user groups". The following is an update of our index. Please see previous issues for the full list.

Preston
D. Coulter — 8, Briar Grove, Ingol, Preston, Lancashire. PR2 3UR

EDUCATIONAL SUB-GROUPS

Cheltenham
Stroud Teachers' Centre (P.J.Heslip)
Chalford Hill C.P. School
Chalford, Gloucestershire.
(after Aug)- Arthur Dye C.P. School
Springbank Rd
Cheltenham, Glos.

BEEBUGSOFT

BEEBUG SOFTWARE LIBRARY

The introductory offer on the first four cassettes in the program library ends on the 31st August 1982. Prices after that date are detailed below. For prices before that date, see the BEEBUGSOFT news sheet distributed with the June issue. From 1st September 1982 we are adding 2 new cassettes to the library: Games 4 and Applications 1.

Games 1 (£3.00)	Starfire (32k)
Games 2 (£3.00)	Moon Lander (16k): 3D Noughts & Crosses (32k)
Games 3 (£3.00)	Shape Match (16k): Mindbender (16k): Rat-Splat (16k)
Utilities 1 (£3.50)	Disassembler (16k): Redefine (16k): Mini Text Editor (32k)

Available from 1st September 1982:

Games 4 (£3.00)	Electric Eel (32k) Fast moving arcade-type game. Guide the Electric Eel around the screen gobbling up everything it comes across, and getting longer with every bite. Various skill levels.
Applications 1 (£3.50)	SuperPlot (32k) Produces tailored screen representations of any function entered. This can be achieved in any of the three major coordinate systems: Cartesian, Polar, or Parametric. Explore the world of graphic representation.

Please add 40p postage and packing to all orders regardless of size, then add 15% VAT, and post to BEEBUG Software, 374 Wandsworth Road, London SW8 4TE.

SOFTWARE COMPETITION

Programs of all types (eg games, educational, business etc) are eligible. They should be submitted on a cassette with explanation, instructions, and documentation on paper (typed if possible); and accompanied by the application form below (or a copy of it). Members may submit more than one program, but each entry must be sent under separate cover. If you require the tape to be returned, please enclose a suitable stamped addressed package.
Prizes range from £10 to £50 with a £100 reserved prize for programs of special merit.

Entry Form

Program Title:..... Membership no (essential):.....

Programmer's name:..... Address:.....

Category:..... .....
(Games, Educational etc)

.....

Will run on Model A.... B....

The program submitted here is my own work, and has not been submitted to another organisation.
I understand that if I am a prize winner my program may be used by BEEBUG for its program library or for publication. In either case this would be with acknowledgement but without further payment.
Judging will be carried out by the editors, and their decision is final.

Signed:..... Date:.....

IF YOU WRITE TO US

Membership applications, subscriptions queries, Magazine back issues and software library.
Send to:

BEEBUG
Dept 1
374 Wandsworth Road
LONDON
SW8 4TE

If in difficulty
tel: 01-720 9314

For back copies, please give your membership number, and send an SAE plus 80p per issue required.

Membership costs £4.90 for 6 months (5 issues)
£8.90 for 1 year (10 issues)

The magazine is published each month except August and December.

Our editorial address for newsletter contributions is:

The Editor
PO Box 50
St Albans
Herts
AL1 2AR

Please do not send subscription letters to the editorial address, this will delay things somewhat.

NEWSLETTER CONTRIBUTIONS

BEEBUG aims to act as an information exchange on all subjects related to the BBC micro. To do this effectively we need contributions of all kinds from members. Please send us your ideas, whether they be in the form of useful hints, program ideas, or longer articles. Programs more than a line or two long should be submitted on cassette.
More substantial articles are particularly welcome and we will pay for these! But in this case, please give us warning of anything that you intend to write. In the case of material longer than a page, we would prefer this to be submitted on cassette in a format similar to that produced by our Mini Text editor (June 82 newsletter or Utilities 1 cassette) or as a Beeb datafile providing that you use the bugs fix on page 21 of the July issue, or you have the 1.0 operating system.

BEEBUG NEWSLETTER is edited and produced by Sheridan Williams and Dr David Graham, and its contents are subject to copyright.
Thanks are due to Phyllida Vanstone, Frances Donovan, Rob Pickering, and Anthony Camacho for assistance with this issue.