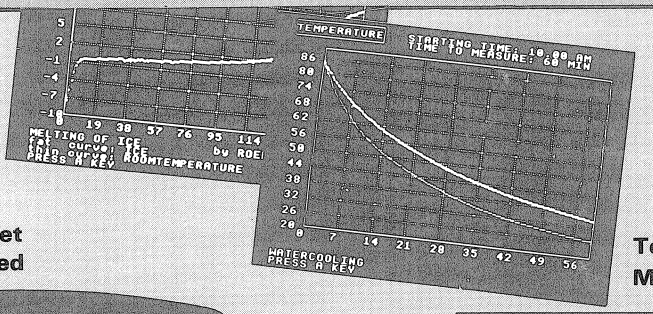
£1.00

BEEBUG

FOR THE BBC MICRO

Vol 3 No 4 AUG/SEPT 1984



Micronet Revisited

Temperature Measurement

- Temperature measurement
- Acorn Z80 second processor reviewed
- Automatic disc menu utility
- Micronet revisited
- Screen driven music synthesizer
- Acorn Bitstik reviewed
- Interstellar raider game
- Pontoon card game
- And much more

BRITAIN'S LARGEST COMPUTER USER GROUP
MEMBERSHIP EXCEEDS 25,000

EDITORIAL

THIS MONTH'S MAGAZINE

If this month's issue of BEEBUG seems fatter and heavier than usual then you will not be surprised to learn that we have included an extra four pages. At the moment this is not a permanent change in the magazine but reflects the fact that this is a two month issue. We felt that it was necessary to devote rather more space than usual to reviews in order that we might adequately cover the new second processors and other hardware recently released by Acorn. The additional pages have allowed us to both review Acorn's Z80 Second Processor and the recently launched Bitstik system in depth, and still reasonably maintain the balance of articles in the magazine. The Z80 Second processor review covers the hardware, operating system and programming languages included as part of the package. In the next issue we shall be turning our attention to the extensive range of applications software that is provided.

With the growing popularity of communications amongst home users we take a further look at Micronet and some of the latest developments of this highly successful enterprise. We expect to follow this in the next issue with a review of Acorn's new Prestel Adaptor and a look at the world of modems and bulletin boards.

With a longer period to the next issue we have included an interesting hardware project complete with comprehensive software that enables your Beeb to act as a temperature probe. Two sensors are included in the design for added interest and flexibilty. The hardware is simple to construct, cheap to buy and has many applications. This excellent article was contributed by one of our Dutch members, Roel Grit, who was the author of the mathematical graphs program that we published in the May issue of BEEBUG.

We have also featured a screen driven music synthesizer which allows all the sound and envelope parameters to be changed graphically on the screen so that you can synthesize music from the keyboard using a wide variety of manufactured sounds. The screen design is a particular feature of this program, and this adds considerably to the flexibilty of the synthesizer.

Mike Williams

TICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOAI

ZOOM COMPETITION RESULTS

We had a very large number of entries to our Zoom Treasure Hunt Competition, which we set in conjunction with the magazine cassette for the June issue, and the two winners of Computer Concepts Graphics ROMs are D.A.Dobbin of Essex and the joint entry submitted by Terry Burden and Nigel Knight from Cornwall. Full details of the solution are included in this month's Advertising Supplement.

HINT WINNERS

Out of this month's good selection of hints we have picked three winners. The £10 prize goes to J.Martins of Norway, and £5 prizes go to Richard Sterry, on behalf of the Wakefield BBC Micro Users Group, and to Michael Quinion. Our thanks to all those who take the time to send their hints in to us.

MAGAZINE CASSETTE/DISC

All the programs in this month's magazine are available on both cassette and disc as detailed on the back cover.

SOFTWARE HELP LINE

We would like to remind members that the Help Line is available primarily for queries relating to subscriptions and software orders. It is not normally practical to answer over the telephone detailed technical queries related to programs published in the magazine. Please write with all the details and mark your letter 'Technical Query'. Some queries can only be dealt with satisfactorily by reference to the original author.

BEEBUG MAGAZINE

GENERAL CONTENTS

HINTS, TIPS & INFO

PROGRAMS

ACORN'S Z80 SECOND PROCESSOR

Reviewed by W.D.S.C. Freeman

With the launch of their Z80 Second Processor, Acorn are clearly aiming at the small business user by the adoption of CP/M as the operating system (widely used on other business machines) and the wealth of business software packages bundled in the price of £299 (inc VAT). This month, in the first of two reviews covering the entire Z80 package, we look at the processor itself, the operating system (CP/M) and the languages supplied with this new system.

Product : Acorn Z80 Second Processor System
Price : £299 (inc. VAT)
Supplier: Vector marketing and Acorn Dealers

Minimum configuration required:
BBC model B with standard disc interface (one using the 8271 controller) and dual double-sided 80 track disc drives. A good monitor and a printer are also useful, though not essential.

The BBC Micro has grown up. With the much delayed arrival of the promised second processors, it is now beginning to fulfill some of its potential. If you have a BBC micro, a pair of 80 track disc drives, a printer and the need for good business software, then the Z80 must be on your shortlist at the very least. There is so much software included with the processor that a review like this cannot hope to be exhaustive. I shall start by describing the second processor system, the operating system (CP/M) and the programming languages that are part of the package. Part 2 of this review will appear in the next issue, and will concentrate on the applications software.

The Z80 is packed in two boxes. The first contains the processor itself plus a set of seven master discs with the software, and a DNFS ROM which replaces your DFS and any NFS fitted (see last months review of the 6502 second processor for more details of this new chip). A voucher for a 1.2 MOS is included should this not already be fitted to your Beeb. An introductory manual provides instructions for setting up the second processor and a set of special function key overlays is also included. The second box contains, eleven (yes eleven) manuals. This comes to a couple of thousand pages but you may well find that you only need a few of these for your own applications.

Setting up is simple. Replace the DFS (and the NFS ROM, if you've got one) with the DNFS ROM. Attach the Z80 second processor to the Tube connector of your Beeb, plug it into the mains and you are ready to go. The cable that joins the Beeb to the Z80 is very short to preserve the integrity of communications between the two processors.

As with the 6502 second processor reviewed last month, there is no absolute restriction on only using the new Acorn DNFS filing systems. The system functioned correctly when tested with a Watford DFS, and we can see no reason why there should be any problems with this alternative DFS. In general, if a filing system works with the 6502 second processor, it should work equally well with the Z80 second processor. The system will not, however, function correctly with ROMs such as Disc Doctor fitted. In general it is best to assume that any product not made or explicitly supported by Acorn could cause problems, although this does not mean that these products will not work.

Moving between applications that either require, or won't work with the Z80 is quite simple; if the Z80 is needed, simply make sure it is turned on, and perform a Control-Break. Likewise, if the Z80 is not needed, turn it off and perform a Control-Break. A normal Break with the Z80 turned on places the user into a mode 0 No Language Environment (NLE) very similar to that discussed in the 6502 review.

You are supplied with 7 discs when you purchase the Acorn Z80 Second Processor, and these are fairly well packed with programs and data. Before using these programs, a specially written program guides you through the preparation of working copies, so that the originals may be stored safely as backups. This is necessary before proceeding further, and the program provides extensive instructions and help messages as you continue. These steps are also covered very well in the Z80 User Guide; a short but comprehensive introduction to the system and its software.

To boot up one of the application discs it must be placed in drive 0 (CP/M refers to both sides of the disc as a single drive with 400K available - very sensible - and thus drives 0 and 2 are now called 'A' by CP/M). Pressing Control-Break will load and run CP/M. A start up message with a new prompt appears. The prompt is A> which tells you that the default drive is A. When the Z80 is in use under CP/M the Beeb is an entirely different machine. *BASIC does not work and if Break is pressed data, programs and CP/M itself will normally be lost - the whole lot then has to be reloaded from disc.

THE CP/M OPERATING SYSTEM

CP/M (Control Program for Microcomputers) is an industry standard disc operating system for eight bit micros (usually based on the Zilog Z80, though originally for the Intel 8080). Like the Acorn MOS/DFS combination it handles disc filing, printing, screen handling etc. It consists of four main components. BIOS (Basic Input Output System) deals with all the devices attached to the computer (screens, discs, printers etc.) and this is specific to the computer. In the case of the Beeb it controls the Tube. CCP (Console Command Processor) interprets and acts upon commands typed in. On top of these, disc management is controlled by BDOS (Basic Disc Operating System), which handles such things as placement of files on the disc etc. Finally there are applic- ations programs that you load and run. Because the Beeb handles input and output devices the Z80 is free to concentrate on data crunching. This makes it a lot more efficient than many other CP/M machines. Once CP/M is loaded about 55K is left for application programs or other languages. For instance with the Z80 version of BBC Basic loaded from disc and running, 40K is available to the user.

Although CP/M is a well established operating system for many micros, there are still many differences between machines. Disc formats are affected in this way and discs are not usually transferrable between different CP/M systems, though the software itself is portable. The conversion of CP/M software to run on the Beeb will usually need to be undertaken by specialist suppliers, and in most cases the task will be beyond that of many end-users.

CP/M recognizes a small number of operating system commands (mostly different from the Acorn DFS commands). For instance to list the files on a disc DIR is used instead of *CAT (directory rather than catalogue). DIR is a built in command; others are ERA (erase a file), TYPE (identical to *TYPE in DFS) etc. As well as these there are utilities on disc like FORMAT and PIP. FORMAT is like the DFS utility of the same name, whilst PIP can copy files (though it is more powerful than *COPY). Utilities are invoked by typing the name of the utility as though it were a command, and this may be followed by various parameters. Two non-standard utilities are STAR which passes the rest of the line entered to the command line interpreter in the Beeb (STAR TEXT is thus equivalent to *TEXT), and DIP which allows files to be moved from DFS to CP/M discs and vice versa.

The CP/M manual is a vast improvement on past issues but is still heavy going. Fortunately Acorn realised this and have done a very good job of providing most of the essential information necessary to use CP/M in the Z80 User Guide. It is quite possible that you may never need to look at the CP/M manual, which is really of more use to systems programmers.

THE LANGUAGES.

Three languages are supplied with

the Z80: BBC Basic, Professional Basic and CIS Cobol. An assembler is also supplied but this is for the 8080 chip which is compatible with, but less powerful than, the Z80. Users who are familiar with BBC Basic will probably carry on using the Z80 version, but the other two languages are arguably more suited for the development of serious business applications.

BBC BASIC

This comes with the Welcome package of programs on disc, and is a faithful reproduction of Issue 2 Basic apart from the assembler which is now for the Z80. The manual supplied is very small for this reason, and just outlines the differences, though the second processor can significntly upgrade performance. The less I/O (access to the Tube) required then the faster the program runs compared with Basic on the Beeb alone. I did not carry out proper benchmarks but a FOR-NEXT loop wrapped around a fairly complex calculation was twice as fast on the Z80. A similar loop that continuously plotted a triangle on the screen was only 5% faster on the Z80.

PROFESSIONAL BASIC

This is very similar to Microsoft Basic as used on the IBM-PC, ACT Apricot etc. It is supplied to allow access to programs written in Basic such as Nucleus (this is part of the supplied. software to be reviewed next issue) but it can be used for applications programming. It is a sophisticated package giving more control over screen formatting and print layouts than BBC Basic, but it is still Basic. The use of a proper run-time package also speeds up execution.

CIS COBOL

COBOL (COmmon Business Oriented Language) has been around for over twenty years, having originated on mainframes. This version has been produced for microcomputers by Micro-Focus. Cobol is a commercial language that pays especial attention to the formatting of data. Printed reports, files and screen layouts can be defined and manipulated much more easily than with Basic. Because it was originally conceived as a language that would be used by untrained programmers it is very wordy, for instance, a Basic statement like:

```
A=B+C
```

becomes the COBOL statement:

```
ADD B,C GIVING A.
```

CIS Cobol is compiled and, like BCPL, it produces a compact code that is interpreted at runtime. Like most other compiled languages all variables must be defined before use.

Overlaying is possible allowing programs larger than the memory size of the machine to be written.

Two other programs are supplied for use with Cobol: Animator and Forms-2. Animator is a very sophisticated Cobol debugging tool. Forms-2 allows the screen to be formatted for data entry and validation, an important aspect of commercial programs. It is very easy to use and could save a lot of programming effort. Not only can a screen layout be designed but if it is the input screen for records in a file then Forms-2 can also create a program that will produce the file itself.

COMPARISON WITH TORCH

The obvious alternative to the Acorn Z80 system is the Torch ZEP100 which has been available for some time now. In BEEBUG Vol.2 No.1 we reviewed the similar Torch Z80 Disc Pack and this system now has a number of successors one of which is the ZEP100 at £343.85. It is possible to buy M-Tec Basic (equivalent to BBC Basic) for £126.50. This was reviewed in BEEBUG Vol.2 No.6. [Torch is now wholly owned by Acorn and we believe the price of the ZEP100 will be reduced by about £80 - Ed.]

The Torch system is not a genuine CP/M system, but uses a similar look-alike operating system called MCP, and also comes with some networking software. The Torch is now supplied with a range of software known as the Perfect (!) range, and this is roughly equivalent to Acorn's Z80 software to be reviewed next issue. No languages are included with the ZEP100 system (M-Tec Basic is extra), and the overall quality is probably not as good as the Acorn product.

TEMPERATURE MEASUREMENTS USING THE BEEB (32K)

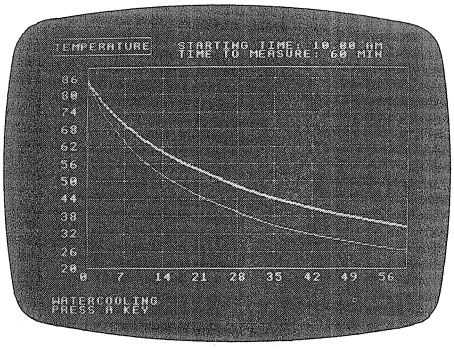
by Roel Grit

Using the ingenious idea in this simple constructional article you can turn your Beeb in to an electronic thermometer with both digital and graphical displays.

The analogue input of the BBC micro is popularly associated with its use as the joystick port. This job is done by accurately reading the voltage at the output of the joystick itself. The elementary electronic circuit described here supplies a voltage in place of this which is related to temperature. The program converts the voltage at the analogue input into a temperature reading which can then be displayed on the screen. The program allows the micro to act as a digital thermometer, with a continuously updated display. In addition, temperatures over a period of time can be displayed in graph form on the screen and the display saved for future reference. The circuit to be described provides two temperature sensors, thus allowing comparative measurements to be made as well.

The analogue port is equipped with four analogue to digital converters, while the program below is only written to deal with the input from two temperature sensitive devices. The program could easily be adapted to cater for any number between one and four of them.

The active device in each thermometer is a thermistor. This is used, in the form of circuit known as a 'potential divider', to provide a variable voltage for input to the analogue port in the same way that a joystick does. In common with all useful temperature measuring devices, this must be calibrated before use, for the temperature range over which you would expect to be using it. The program provides an option to calibrate each temperature measuring device, by making some comparative readings with a conventional thermometer. This calibration can then be saved on tape or disc for further use.



HARDWARE REQUIREMENTS

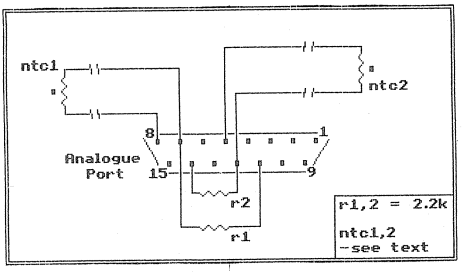
Below is a list of the components required to build the two thermometers for use with the program.

2.2k ohm thermistors (2 off)
(Siemens part K164-2.2k from Electrovalue, Ø784-336Ø3)
1k ohm resistors (2 off)
15 pin D-type plug+cover
some two core wire

Apart from the thermistors, the other components should be readily available from any electronic supplies shop. The thermistors cost around 18p each and are a small, negative temperature coefficient, disc type with a nominal resistance of 2.2k at 25 deg.C (this is not as complicated as it sounds; it simply means that as the temperature of the device rises, its resistance decreases). Though the values are not critical, the 1k resistor must be changed according to the resistance of the thermistor at the centre of its operational temperature range. For example, if the thermistor chosen was a 22k ohm, NTC type then you would have to use a 1Øk ohm resistor in each temperature sensor. It is not recommended that a lower value thermistor be used than the one listed above.

CONSTRUCTION

Construction demands the minimum of



soldering and proceeds as follows. Solder one of the 1k ohm resistors between pins 7 and 11 and the other between pins 12 and 14 of the D-type plug (you should be able to fit these inside the plug cover). Solder the ends of a twin core lead to pins 7 and 8 and another twin core lead to pins 12 and 5. Across the ends of each of these leads, solder a thermistor; it doesn't matter which way round. These are the temperature sensing probes; one connected to channel 1 and the other to channel 2 of the analogue port. The length of the connecting cable is not significant; it simply depends on how far away from the micro you may wish to place your temperature sensors.

USING THE PROGRAM

When you have typed in and saved the program, you will find that when it is run you will be presented with a menu providing a choice of 7 options. The first one that must be taken is option number 1, to make a calibration curve.

This will require you to raise the temperature of the probes to a known point and record it. You will therefore need another, reasonably accurate, thermometer at this stage. Some care must be taken as the accuracy of the computer thermometer will only be as good as your readings from the other thermometer. As you are prompted on screen, enter the reading of the temperature and the computer will record this against the value that it reads from each probe. As many as a hundred different temperatures may be recorded and a simple linear interpolation on these is used by the program to produce an accurately calibrated scale for the thermometer.

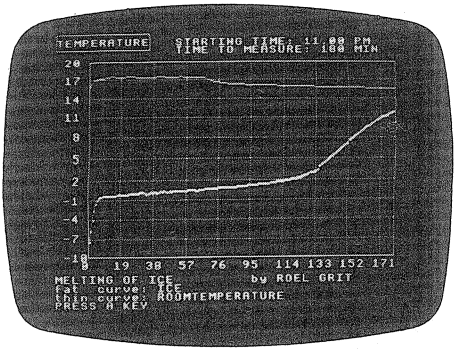
You could place some ice in a saucepan, along with your conventional thermometer, and let it warm up until the ice begins to melt. At this point the temperature will be around 0 deg.C. Warm up the saucepan and as the (by now) water heats up, take intermediate readings from the thermometer and enter them at the keyboard. At the boiling point of the water the temperature will be close to 100 deg.C.

Some points of caution, are firstly, that you should not let the sensors touch the sides or bottom of the saucepan as these are likely to be at significantly different temperatures to the water. Seal the sensors inside a balloon or something watertight so that bared wire connections are not in contact with the water. Both of these can affect the validity of the readings obtained from the sensors.

An increased temperature range may be measured (provided it is within the specified working range of the thermistors used, which may be to an upper limit of 125 deg.C) by calibrating the sensors during immersion in hot oil or cold alcohol in a similar fashion to the method used with water. Alcohol is particularly useful for calibrating below 0 deg.C. Note that all readings during the calibration phase must be entered in ascending order, so you must start at the lowest temperature you wish to measure.

It must be realised that thermistors are not linear devices and accuracy of measurement may be increased by calibrating over narrower intervals in the operating temperature range.

After the last reading has been given, press Return and go back to the main menu. Select option 2 and save the calibration that you have made. To measure any temperature, this calibration data will always be needed by the program and should have been loaded from file prior to any temperature reading from the probes. Option 3 of the main menu is used to load it. (You could have several different calibration files available for different temperature ranges).



A simple digital display of the measured temperatures on each probe is provided by option 4. The display is updated in real time and you will be able to see from this how quickly or slowly your thermistor probes respond to temperature change. You can try this, and identify which probe is which, by holding the probe tight in your fingers and watching the corresponding temperature rise. Release the probe, and watch the temperature fall again.

Options 5 and 6 are for constructing and saving graphs of temperature against time, and this facility is flexible enough to cover a wide variety of applications (provided you've got a printer and screendump routine, it will do away with the need for pen chart recorders too!). You may plot two continuous temperature functions over a chosen period of time and over a chosen temperature range. The graph may be saved to the current filing system and reloaded at any time. A caption setting facility is also provided for you to title your graphs with.

Menu option 7 provides for termination of the program, while pressing Escape at any time always returns you to the main menu.

IDEAS FOR FURTHER USE

If you wish to use a screen dumping routine to output a temperature graph on to your printer, then append your routine to the program as a procedure call and insert a penultimate line to call it in DEF PROCtempdiag and do the same to DEF PROCdiagfromfile.

Unusually, for the addition of a new piece of hardware to the Beeb, this idea does not require the use of machine code; it simply makes use of the ADVAL instruction from within Basic to do the required inputting of data. This means that the resulting program can be easily adapted for other applications.

If you connect a suitable relay across the cassette relay output (pins 6 and 7 of the cassette socket), then by testing for specific temperature levels from within DEF PROCmeasuretemp and using the *MOTOR 0 and *MOTOR 1 commands, it's possible to maintain control of a heating system with the Beeb.

The basic idea here need not be limited to temperature measurement. You might replace the NTC thermistors with LDR's (light dependent resistors) and with some slight changes to the program run a simple control system for your photographic dark-room (all the better if you've got a colour monitor and change all text output on the screen to red for safety lighting!)

TECHNICAL NOTES

On the program:

The variable MAXN is the maximum number of calibration values that are used. These are held in array W1 for the probe on channel 1 and W2 for channel 3 while the corresponding temperatures are held in array T. The function FNtemperature(CH) obtains a temperature reading by comparing the value at the analogue input with a recorded value and temperature in the calibration arrays.

On the hardware:

Positive temperature coefficient thermistors are also available (PTC's) and there is no reason why one of these should not be used instead of the NTC's specified here. However, their inverse performance with temperature means that they must be interchanged with the position of the resistors in the current circuit so that the voltage at the input to the analogue channel continues to move in the same direction as the temperature.

```
10 REM TEMPS
20 REM Author ROEL GRIT
30 REM Version B1.0
```

→

```
   40 REM BEEBUG AUG/SEPT 1984
   50 REM Program Subject To Copyright
   60 :
  100 VDU23,240,192,192,0,0,0,0,0,0
  110 ON ERROR GOTO 130
  120 MAXN=100:DIM T(MAXN):DIM W1(MAXN)
,W2(MAXN),X(200),Y1(200),Y2(200)
  130 MODE7:PROCtitle("TEMPERATURE MEAS
UREMENT")
  140 PRINTTAB(0,3)CHR$136;
  150 PRINT"TYPING <ESCAPE> RETURNS TO
THIS MENU"''
  160 PRINT''"1";SPC4;"MAKE A CALIBRATIO
N CURVE"
  170 PRINT'"2";SPC4;"SAVE CALIBRATION
CURVE"
  180 PRINT'"3";SPC4;"LOAD CALIBRATION
CURVE"
  190 PRINT'"4";SPC4;"MEASURE TEMPERATU
RE"
  200 PRINT'"5";SPC4;"MAKE DIAGRAM TEMP
/TIME"
  210 PRINT'"6";SPC4;"LOAD DIAGRAM"
  220 PRINT'"7";SPC4;"STOP THE PROGRAM"
  230 PRINT''TAB(10)"YOUR CHOICE ";:X$=
GET$:X=VAL(X$)
  240 IF X=1 THEN PROCmakecalib
  250 IF X=2 THEN PROCsavecalib
  260 IF X=3 THEN PROCcalibfromfile
  270 IF X=4 THEN PROCmeasuretemp
  280 IF X=5 THEN PROCdiagdata:MODE 4:P
ROCdiagdecor:PROCdiagscale:PROCtempdiag
  290 IF X=6 THEN MODE4:PROCdiagfromfile
  300 IF X=7 THEN MODE7:END
  310 GOTO 130
  320 :
 1000 DEF PROCmakecalib:PROCtitle("MAKE
 A CALIBRATION CURVE")
 1010 N=0:PRINT'"PRESS <RETURN> AFTER T
YPING TEMPERATURE."
 1020 PRINT"MAX.NUMBER CAL.TEMPERATURES
 IS ";MAXN'"PRESS <RETURN> TO STOP."
 1030 PRINT"THE CALIBRATION TEMPERATURE
S MUST BE IN ASCENDING ORDER"
 1040 PRINT;'TAB(4);"TEMP";TAB(13);"ADV
AL(1)";TAB(25);"ADVAL(2)"'
 1050 REPEAT
 1060 N=N+1
 1070 PRINTTAB(0);N;:INPUT TAB(4) T$:IF
N>MAXN THEN 1120
 1080 IF T$<>""THEN T(N)=VAL(T$):W1(N)=
ADVAL(2):W2(N)=ADVAL(3)
 1090 IF N>1 AND T$<>""THEN:IFW1(N)>=W1
(N-1) OR W2(N)>=W2(N-1) THEN PRINT"TEMP
. MUST BE HIGHER":GOTO1070
 1100 PRINTTAB(14,VPOS-1);W1(N);SPC(7);
W2(N)
 1110 UNTIL T$=""
 1120 W1(0)=N-1:W2(0)=N-1
 1130 ENDPROC
 1140 :
 1150 DEF PROCsavecalib:PROCtitle("SAVE
 CALIBRATION")
 1160  INPUT'''"TYPE NAME OF FILE TO SAV
E CALIBRATON"'N$:Q=OPENOUT(N$)
 1170 PRINT''TAB(1);"MEAS.";TAB(10);"TE
MP.";TAB(20);"ADV 1";TAB(30);"ADV 2"
 1180 PRINTSTRING$(37,"_")
 1190 FOR N=0 TO W1(0)
 1200 PRINT#Q,T(N),W1(N),W2(N)
 1210 PRINT TAB(3);N;TAB(11);T(N);TAB(2
0);W1(N);TAB(30);W2(N)
 1220 NEXTN
 1230 CLOSE#Q:PRINT''TAB(12)"PRESS A KE
Y":X$=GET$
 1240 ENDPROC
 1250 :
 1260 DEF PROCcalibfromfile:PROCtitle("
READ CALIBRATION FROM FILE ")
 1270 INPUT'''"TYPE THE NAME OF THE CALI
BRATION FILE"'N$
 1280 PRINT'''CHR$(136);TAB(7);"LOADING
 - please wait"
 1290 Q=OPENIN(N$)
 1300 CLS:PROCtitle("CALIBRATION CURVE"
)
 1310 PRINT'''TAB(8);"TEMP.";TAB(15);"C
AL.1";TAB(25);"CAL.2"'STRING$(36,"_")
 1320 INPUT#Q,T(0),W1(0),W2(0)
 1330 FOR N=1 TO W1(0)
 1340 INPUT#Q,T(N),W1(N),W2(N)
 1350 IF W1(N)<>0 THEN PRINT T(N),W1(N)
,W2(N)
 1360 NEXTN
 1370 CLOSE#Q:PRINT''TAB(12)"PRESS A KE
Y":X$=GET$
 1380 ENDPROC
 1390 :
 1400 DEF PROCmeasuretemp:PROCtitle("ME
ASURING TEMPERATURE")
 1410 VDU136:PRINT TAB(6,20);"PRESS <ES
CAPE> TO STOP"
 1420 VDU 23;8202;0;0;0;:T1=0:T2=0:NM=1
0:REM NM=NUMBER MEASUREMENTS
 1430 FORP=1 TO NM
 1440 IF W1(0)<2 THEN PROCnocalib:ENDPR
OC
 1450 T1=T1+FNtemperature(2):T2=T2+FNte
mperature(3)
 1460 NEXTP:T1=INT(T1/NM*10+0.5)/10:T2=
INT(T2/NM*10+0.5)/10
 1470 FOR X=1 TO 2:PRINT TAB(4,9+X);CHR
$141"TEMPERATURE 1:"SPC4;T1:NEXT
 1480 FOR X=1 TO 2:PRINT TAB(4,12+X);CH
R$141"TEMPERATURE 2:"SPC4;T2:NEXT
 1490 GOTO 1420
 1500 ENDPROC
 1510 :
 1520 DEF PROCdiagdata:PROCtitle("TEMPE
RATURE/TIME-DIAGRAM")
```

→

```
1530 PRINT''""FOR HOW LONG DO YOU WANT
TO MEASURE""(MINUTES)";
1540 INPUT MTIME
1550 NMEAS=MTIME*30:IF NMEAS>200 THEN
NMEAS=200
1560 PRINT''"MINIMUM AND MAXIMUM TEMPER
ATURES"
1570 PRINT"don't go beyond the calib t
emperatures"
1580 PRINT"MIN.TEMPERATURE TO MEASURE:
";TAB(36);"dgr"
1590 PRINT"  not below ";T(1);" dgr"
1600 INPUTTAB(29,VPOS-2) MINTEMP:IF W1(
0)=0 THEN MAXSTD=0 ELSE MAXSTD=T(W1(0))
1610 PRINT TAB(0,VPOS+1);"MAX.TEMPERAT
URE TO MEASURE:";TAB(36);"dgr"
1620 PRINT"  not above ";MAXSTD;" dgr"
1630 INPUT TAB(29,VPOS-2) MAXTEMP
1640 INPUT''"STARTING TIME " STARTTIME$
1650 VDU136:INPUT''""PRESS <Return> TO
START   "X$:IF X$<>""THEN 1650
1660 ENDPROC
1670 :
1680 DEF PROCdiagdecor
1690 PRINT''"TEMPERATURE";SPC(3);"START
ING TIME: ";STARTTIME$;
1700 PROCcadre(0,944,352,1008)
1710 PRINT TAB(14,2);"TIME TO MEASURE:
";MTIME;" MIN";
1720 PRINT TAB(6,28);"TEMP.1"
1730 PROCcadre(184,16,392,80):PRINT TA
B(11,30);"C";
1740 PRINT TAB(19,28);"TIME";
1750 PROCcadre(504,16,840,80):PRINT TA
B(16,30);"       SEC";
1760 PRINTTAB(31,28);"TEMP.2"
1770 PROCcadre(984,16,1192,80):PRINT T
AB(36,30)"C";
1780 ENDPROC
1790 :
1800 DEF PROCdiagscale
1810 VDU5:PROCcadre(128,224,1279,896)
1820 FOR STRP=0 TO (MAXTEMP-MINTEMP) S
TEP INT((MAXTEMP-MINTEMP+12)/12)
1830 STP=224+STRP*672/(MAXTEMP-MINTEMP)
1840 MOVE 128,STP:PLOT 21,1279,STP
1850 NEXT STRP
1860 FORTW=MINTEMP TO MAXTEMP STEP INT
((MAXTEMP-MINTEMP+12)/12)
1870 MOVE 32,238 +(TW-MINTEMP)*672/(MA
XTEMP-MINTEMP)
1880 IF TW<0 OR TW>9 THEN PRINT;TW,ELS
E PRINT" ";TW,
1890 NEXT TW
1900 STP=INT((MTIME+10)/10)
1910 IF MTIME<2 THEN STP=STP/6
1920 IF MTIME>=2 AND MTIME<=4 THEN STP
=STP/2
1930 FOR MT=0 TO MTIME STEP STP
1940 MOVE 128+MT*1150/MTIME,224:PLOT 2
1,128+MT*1150/MTIME,896
1950 MOVE 100+MT*1150/MTIME,210:PRINT;
INT(MT*100+.5)/100;
1960 NEXT MT
1970 VDU4
1980 ENDPROC
1990 :
2000 DEF PROCtempdiag
2010 IF W1(0)<2 THEN PROCnocalib:ENDPR
OC
2020 VDU 23;8202;0;0;0;:VDU29,128;224;
2030 TIME=0:STP=1:TEMR=MAXTEMP-MINTEMP
:ENDTIME=MTIME*6000:X(0)=0
2040 Y1(0)=(FNtemperature(2)-MINTEMP)*
672/TEMR
2050 Y2(0)=(FNtemperature(3)-MINTEMP)*
672/TEMR
2060 STPTIME=MTIME*6000/NMEAS:ENDTIME=
MTIME*6000
2070 REPEAT
2080 T1=0:T2=0:P=0
2090 REPEAT
2100 T1=T1+FNtemperature(2):T2=T2+FNte
mperature(3)
2110 P=P+1:T1M=T1/P:T2M=T2/P
2120 PRINTTAB(6,30);(INT(T1M*10+0.5))/
10
2130 PRINTTAB(31,30);(INT(T2M*10+0.5))
/10
2140 PRINTTAB(18,30);INT((TIME+50)/100)
2150 UNTIL TIME>=STP*STPTIME
2160 Y1(STP)=(T1M-MINTEMP)*672/TEMR
2170 Y2(STP)=(T2M-MINTEMP)*672/TEMR
2180 X(STP)=STP*1151/NMEAS
2190 MOVE X(STP-1),Y1(STP-1):DRAW X(ST
P),Y1(STP)
2200 MOVE X(STP-1),Y2(STP-1):PLOT 21,X
(STP),Y2(STP)
2210 VDU5:MOVEX(STP),Y2(STP):PRINTCHR$
(240):VDU4
2220 STP=STP+1
2230 UNTIL TIME>=ENDTIME
2240 VDU28,0,31,39,27:VDU29,0;0;
2250 CLS:PRINT"DO YOU WANT TO SAVE THI
S DIAGRAM? "
2260 PROCdiagtofile
2270 ENDPROC
2280 :
2290 DEF FNtemperature(CH)
2300 N=0:WX=ADVAL(CH)
2310 IFWX>W1(1) OR WX>W2(1) THEN TEMP=
T(1):GOTO2390
2320 IFWX<W1(W1(0)) OR WX<W2(W2(0)) TH
EN TEMP=T(W1(0)):GOTO2390
2330 REPEAT
2340 N=N+1
2350 IF CH=2 THEN UNTIL WX>W1(N+1) AND
WX<=W1(N) OR WX<W1(N+1) AND WX>=W1(N)
```

→

```
 2360 IF CH=3 THEN UNTIL WX>W2(N+1) AND
 WX<=W2(N) OR WX<W2(N+1) AND WX>=W2(N)
 2370 IF CH=2 THEN TEMP=T(N)+(WX-W1(N))
*((T(N+1)-T(N))/(W1(N+1)-W1(N)))
 2380 IF CH=3 THEN TEMP=T(N)+(WX-W2(N))
*((T(N+1)-T(N))/(W2(N+1)-W2(N)))
 2390 =TEMP
 2400 :
 2410 DEF PROCcadre(SX,SY,X,Y)
 2420 MOVE SX,SY:DRAW X,SY:DRAW X,Y:DRA
W SX,Y:DRAW SX,SY
 2430 ENDPROC
 2440 :
 2450 DEF PROCnocalib
 2460 PRINT TAB(5,6);CHR$(136);"MAKE A
CALIBRATION CURVE OR"
 2470 PRINT TAB(3);CHR$(136);"-if possi
ble- LOAD FROM FILE "
 2480 PRINT''''TAB(12);"PRESS A KEY ":X
$=GET$
 2490 ENDPROC
 2500 :
 2510 DEF PROCtitle(TITLE$)
 2520 CLS:SP=INT((40-LEN(TITLE$))/2)
 2530 VDU157,129:PRINTTAB(SP-5);CHR$(14
1);TITLE$
 2540 VDU157,129:PRINTTAB(SP-5);CHR$(14
1);TITLE$
 2550 ENDPROC
 2560 :
 2570 DEF PROCdiagtofile
 2580 CLS:PRINT"DO YOU WANT TO SAVE THI
S DIAGRAM? "
 2590 X$=GET$:IF X$<>"Y" AND X$<>"y" TH
EN ENDPROC
 2600 CLS:INPUT"WHAT CAPTION WOULD YOU
LIKE?"'TEXT$
 2610 CLS:INPUT''"TYPE THE NAME OF THE
FILE TO SAVE"'N$
 2620 R=OPENOUT(N$)
 2630 PRINT#R,STARTTIME$,MTIME,NMEAS,MI
NTEMP,MAXTEMP,TEXT$
 2640 FOR N=0 TO NMEAS
 2650 PRINT#R,X(N),Y1(N),Y2(N)
 2660 NEXT N
 2670 CLOSE#R:CLS:PRINT"READY"''"PRESS A
KEY":X$=GET$
 2680 ENDPROC
 2690 :
 2700 DEF PROCdiagfromfile
 2710 PRINT STRING$(39,"*")
 2720 PRINT TAB(10);"LOAD DIAGRAM FROM
FILE "
 2730 PRINT STRING$(39,"*")
 2740 INPUT''"TYPE THE NAME OF THE DIAG
RAM TO LOAD"'N$
 2750 R=OPENIN(N$)
 2760 INPUT#R,STARTTIME$,MTIME,NMEAS,MI
NTEMP,MAXTEMP,TEXT$
 2770 FOR N=0 TO NMEAS
 2780 INPUT#R,X(N),Y1(N),Y2(N)
 2790 NEXTN
 2800 CLOSE#R:CLS:PROCdiagdecor:PROCdia
gscale
 2810 VDU29,128;224;
 2820 FOR N=0 TO NMEAS-1
 2830 MOVE X(N),Y1(N):DRAW X(N+1),Y1(N+
1):MOVE X(N),Y2(N):PLOT21,X(N+1),Y2(N+1
):VDU5:PRINTCHR$(240):VDU4
 2840 NEXTN
 2850 MOVE0,0:VDU29,0;0;:VDU28,0,31,39,
27:CLS:PRINT'TEXT$
 2860 PRINT"PRESS A KEY":X$=GET$
 2870 PROCdiagtofile
 2880 ENDPROC
```

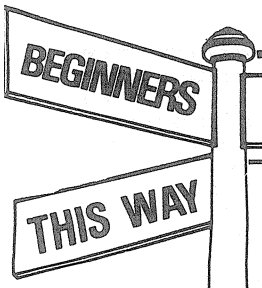
HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

POSITION OF SYSTEM VARIABLES IN MEMORY - P.S.Ganney

The system varibales @% and A% to Z% are all directly accessible at memory locations &400 to &468. As each is an integer, each has four bytes designated to it such that the value of A% is !&404 etc. So it is easy to access these system variables from within machine code programs without passing parameters using the CALL instruction (see pages 214 and 446 of the User Guide), but is not of course Tube compatible.

ERROR DETECTION WHEN OPENING FILES - Wakefield BBC Micro Users Group

When opening data files using the commands OPENOUT, OPENIN and OPENUP, the syntax is basically chan%=OPENOUT("myfile"). The first file to be opened is allocated a channel number. Open another before closing the first and it is allocated the next consecutive channel number. If however you try to open a non-existent file with OPENIN or OPENUP on disc, instead of generating an error, the channel allocated is 0. Only when you use INPUT# or PRINT# will the DFS produce an error with code 222. The file will of course not been opened and so there is no need to use CLOSE#. So by detecting when chan%=0 it is possible to intercept mistakes at an early stage.



CHANGING COLOUR

BEGINNERS START HERE – CHANGING COLOUR

by John Wellsman

This month we turn our attention away from functions and procedures as John Wellsman takes you on a guided tour of the world of colour on the Beeb.

The BBC micro does give the user a great deal of control over which colours are used at any time, and one of the very powerful commands at our disposal is the "VDU 19" statement. This allows us to control exactly which colours we use in the various modes and to change them selectively during a program. However, to the beginner the explanations in the User Guide can be rather off-putting because of the unfamiliar terms used in explaining the use of VDU19.

Describing a colour as "logical" to a beginner in computing is about as meaningful as calling a cart-horse transcendental! And if a colour is not actual, what can it be? It would have been better, perhaps, to have devoted an entire chapter in the Guide to the control and manipulation of colour rather than mixing it with graphics.

The so-called "actual colours" are the colours which are available to the user in any of the modes. There are sixteen colours available though eight of these are two colour "flashers". Each colour has a reference number used in the VDU19 statement. These numbers are listed in several places in the User Guide, especially on page 165. They are quite independent of any mode and they can, therefore, be meaningfully called 'Actual Colour Values' because when used in the VDU19 statement they always refer to the same colour irrespective of mode. So 2 will ALWAYS signify green and so on.

Table of Actual Colour Values

Value	Actual Colour
0	black
1	red
2	green
3	yellow
4	blue
5	magenta (blue-red)
6	cyan (blue-green)
7	white
8	flashing black-white
9	flashing red-cyan
10	flashing green-magenta
11	flashing yellow-blue
12	flashing blue-yellow
13	flashing magenta-green
14	flashing cyan-red
15	flashing white-black

What is the logical colour? Depending on the mode (except mode 7 in which VDU19 does not apply) either two, four or sixteen colours can be used. They are not really colours at all, but "slots" to which we can assign colours. On power-up, the slots in the two colour modes contain black & white, in the four colour modes black, red, yellow and white. Mode 2 has the full sixteen colours in its sixteen slots. These colour slots can be defined and referred to by a number. The default background slot is always zero but this can be changed. So in a two colour mode the slots are 0 and 1, in the four colour modes they are 0 to 3, and in mode 2 with sixteen colours, 0 to 15.

All these numbers have default colour values according to the mode, but they can all be changed by the user. They can in some ways be thought of as colour variables but only in the sense that the actual colour values can be assigned to them by the VDU19 statement. They are what the User Guide refers to as "logical colours" but if the expression "colour" has to be used to describe them it would seem more

useful to call them 'Relative Colour Values' as they are strictly relative to the mode and any subsequent user alterations.

We can abbreviate 'Actual Colour Value' to ACV and 'Relative Colour Value' to RCV.

Now let us look at the statement itself without bothering about what the values mean for the moment. It can be written in two ways, for example:

```
VDU19,1,2,0,0,0
```

or

```
VDU19,1,2;0;
```

These say exactly the same thing but because it is shorter, we shall be using the second form. It is important to note the position of the two commas and the two semi-colons. Using our abbreviations, we can re-write it as:

```
VDU19,RCV,ACV;0;
```

This means that in the first position after VDU19, we insert the relative colour value that we wish to change, and in the next position, the actual colour that we want to assign to it. The final zero is unused by the computer. If we look at our first example, VDU19,1,2;0; its effect would depend on what mode we were in. If we use it in mode 0, a two colour mode, then it would change colour 1, normally the foreground colour, to green, 2 being the ACV of green. Type in this little program, then RUN it and press the space-bar.

```
10 MODE 4:REM two colour mode
20 PRINTTAB(10,10)"TEST"
30 A=GET
40 VDU19,1,2;0;
```

(In case you are quite a beginner, and dont understand line 30, this simply stops the program until you press any key and then carries on.)

Note that the change to green is immediate, you do not have to print "TEST" again, and that everything, including the cursor is printed in green until you change mode, press Break or execute another VDU19 statement changing relative colour value 1 to another colour. Add another GET line to the above program and then a second VDU19 statement changing the 2 to some other value (less than 16), and you will see the text change colour again on pressing the space-bar. In the same way you can change the background by replacing the 1 by a zero in the RCV position. You may know that some computers, the SPECTRUM for example, normally use black letters on a white background, which you may think is better. If so, try this:

```
10 MODE 4
20 VDU19,0,7;0;
30 VDU19,1,0;0;
40 P.TAB(5,10)"This is black on white"
```

Line 20 changes the background colour, usually RCV zero, from black (the default colour) to white, ACV 7. Line 30 changes RCV 1, in this mode the only other RCV that is possible, to black, ACV 0.

Try the same effect using mode 6. You may find this useful sometimes. Changing the background (RCV 0) to blue (ACV 4) in this mode gives the effect that you may have seen in some of the BBC TV programmes on computing.

With a four colour mode, the same rules apply, only now you have three colours plus background colour which can be used. Type this in:-

```
10 MODE 5:REM 4 colour mode
20 FOR X = 1 TO 3
30 COLOUR X:P.TAB(10,9+X)"COLOUR ";X
40 NEXT
50 FOR X= 1 TO 3
60 A=GET
70 VDU19,X,4;0;
80 NEXT
```

When you run it, the program will first display on the screen the three default colours of the mode with their relative colour values. Now look at line 70 and check what is going to happen when you press the space-bar. The loop will successively change RCV 1-3 to ACV 4 which is blue. So when you have pressed the space-bar three times you have converted all the mode 5 colours into blue and changed, temporarily, mode 5 to a two colour mode. To

experiment a bit further, change line 50 to:

```
50 FOR X=0 TO 3
```

This will include the background colour in the change to blue, which will happen the first time that the space-bar is pressed. But as you continue, the lines of text will. apparently disappear, though in reality they are also being changed to blue which makes them invisible.

Now change line 50 to:

```
50 FOR X= 1 TO 15
```

and in line 70 substitute either 1, 2 or 3 for X and substitute X for 4.

If you now run the program and repeatedly press the space-bar, you will change whatever relative colour value you have used in line 70 through the whole pallette of available colours.

The only mode which has the full range of colours available is mode 2. Only in this mode do the default values of RCVs correspond to the ACVs but they can all be changed, if necessary, in exactly the same way as the other modes.

Just for fun, the following will change mode 2 to a two colour mode using only black and white.

```
10 MODE 2
20 FOR X=1 TO 15
30 COLOUR X:PRINT"COLOUR"
40 NEXT
50 PRINT"PRESS SPACE-BAR"
60 A=GET
70 FOR X= 1 TO 15
80 VDU19,X,7;0;
90 NEXT
```

Lines 20 to 30 will display all the default colours of mode 2. Lines 70 to 90 will change them all to ACV 7, which is the value for white. Remember that these changes made by VDU19 will remain in force until you change the mode or press Break. Neither Escape nor NEW will affect the changes.

One more point, if you use a value relating to a colour, which is too large in the context, the computer will always modify the value (in the same way as the MOD operator) accordingly. For instance, if you are in mode 5 (a four colour mode) and you execute VDU19,5,2;0; the computer will divide the "5" by the number of colour slots in the mode, in this case "4" and will regard the remainder "1" as the proper value in this position. In the instruction VDU19,1,25;0; in mode 2 the "25" would be divided by "16" and would put actual colour "9" into slot 1. This will apply generally with colour values in such statements as COLOUR and GCOL as well.

Now you can really ring the colour changes in your programs for the very best effects.

POINTS ARISING

MUSIC WHILE YOU WORK (BEEBUG VOL.3 NO.2)

Some confusion arose in the preparation of the text accompanying this program between the different needs of cassette and disc users. The following notes apply to the program MUSIC as listed in the magazine. The program will work as listed for disc users but cassette users should change the hexadecimal value '&900' in line 120 to '&D00'. No other changes are required.

The confusion also affected the magazine cassette and disc. Because of the additional lines included in these versions, The following line should be added:

```
1415 P%=P%+15+LEN$(ENV+14)
```

and cassette users should change the value of B01 in line 100 to A01 and the value of B00 at line 1410 to A00. We apologise for any inconvenience caused by these errors.

FRAK!

An exciting new game reviewed by David A. Fell

```
Name     : Frak
Supplier : Aardvark Software
Price    : £8.90
Rating   : *****
```

The standard of games for the Beeb was set very early in its history by an implementation of a popular arcade game under the name of Planetoid. About nine months ago, a game called Zalaga was first seen. This was based upon an arcade game called Galaga. Since then, mighty forces have been at work devising a game that combines total originality with masterful programming; the result - Frak!

The basic concept in Frak is for you, who play the part of Trogg, to negotiate an ingenious complex of ladders, ropes, chains, rock ledges, logs and iron girders in the endless pursuit of that unattainable goal - the end of the game. There are three basic screen layouts, each of which features a different monster, and you have to collect all of the keys in order to complete the level. Each key collected gives a time bonus, but the available time is slowly ticking away. If your time reaches zero in the early stages you are not penalised, but later stages inflict a variety of mean effects upon poor Trogg.

There are gems to be collected for bonus points, and light bulbs for a 10 second time bonus. Your plight is also further hindered by daggers and balloons, and your only weapon against these, and the three monsters Scrubbly, Poglet and Hooter, is your faithful yoyo (yeah, a yoyo as a weapon!). With this amazingly versatile weapon you can achieve some astounding feats of self defence - even catching a Scrubbly and a dagger at the same time if you aim correctly. [This does require a considerable degree of dexterity - a pained Ed!]

The graphical depiction of the characters in the game is very detailed (the game's in Mode 1), and movement is on a pixel basis - not a byte basis. As you move left and right, the screen scrolls with a delightfully smooth movement to accommodate your actions.

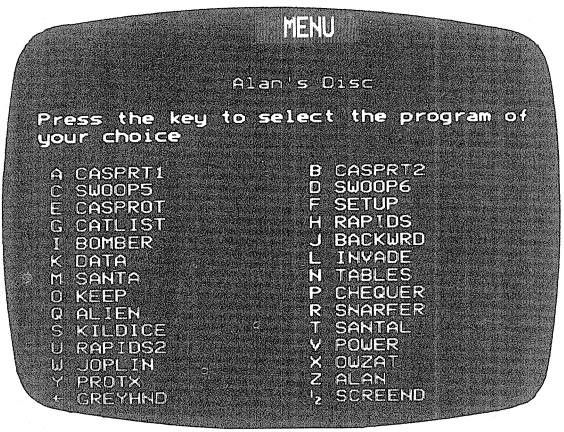
At first, Frak is merely a question of not being hasty, but as the game develops, it reveals itself to be a combination of both action and thought games. The later screen layouts require some advance planning before being attempted as there are a significant number of one-way traps, precise jumps and loop backs. Perhaps the first thing that can trap the unwary is finding that his yoyo will not reach the Scrubbly that he is attempting to 'knock out' of play. When flinging (how else do you describe 'firing' a yoyo?) your yoyo, you need to make sure that you utilise its full potential, or you will find a number of apparently insoluble problems.

As the game progresses, there are some delightful 'twists' and variations that will keep you in confusion for a long time. The high score table features three 'check characters', and these are for verification of the scores in the high score competition. Frak sets new standards for games on the Beeb, and I would recommend that any games player should definitely purchase this masterpiece of programming.

AUTOMATIC DISC MENU UTILITY (DFS)

by Peter Hunter

This month's utility is an extension of the disc menu concept that we introduced in BEEBUG Vol.2 No.4. This most useful variation on that theme will automatically list any appropriate files that are present on the disc when the program is run. New files added to the disc will be accommodated automatically.



The menu program listed here displays a list of the programs present on the chosen disc drive(s) every time it is run. This is accomplished by a little-known call to the DFS that allows for the filenames in the current directory to be read by virtually any program. The program listed here reads these filenames and then forms them into a menu for you to select the program to run. The menu is not restricted to a single drive, but can be readily extended to present a single menu using up to all 4 drives if required.

To use the program, first type it in and save it away safely on disc. Note that you should save the menu program explicitly as "MENU" for the program to work correctly and to avoid "MENU" itself appearing as one of the options in the menu list. The program should also be saved onto each disc with which you wish to use the menu utility. To use the program, put the appropriate disc into drive 0, and then CHAIN "MENU". The program will then scan the disc drive(s) present, and display a list of programs that may be run. To select a program, all you have to do is press the highlighted letter alongside it on the screen, and the menu program will then execute that program for you.

The current version of the menu program can cater for two types of files: Basic programs (to be CHAINed), and machine code programs (to be *RUN). Because the ways in which these two types of file are loaded and executed are different, the program needs to be able to distinguish between them. This is accomplished by using different directories for each type of program. A Basic program is indicated by saving it in directory "B" (see your disc manual if you are uncertain as to how to achieve this), and a machine code program is indicated by saving it in directory "R". The other directories are not used by the menu program, and may contain any other files; these will not appear in the menu display. For example, you may have a large machine code program that has a 'header' program in Basic. The suggested method for accessing this program would be to put the header program into the "B" directory, so that this will be listed in the menu options, and the machine code program in to the "$" directory, so that this will not be listed. You could, as an alternative, put the machine code program into the "R" directory. This would allow for the header program to be run as normal, or for the machine code program to be run directly.

To allow greater flexibility, the program includes a facility to scan more than just one disc surface (i.e. not just DRIVE 0); Line 1140 governs this. You can alter the program to scan all four drives if you wish. You should note that although the program works quite correctly if a large number of

files are present on the disc (say due to the 62 file catalogue option of the Watford DFS), the screen display will scroll and will not appear as attractive if more than 40 files are found. This could be avoided at the expense of extra code, but is unlikely to be a problem for most users.

In practice, the most convenient way of using the menu program is for it to be run automatically, by booting the disc with Shift-Break. This is done by creating a !BOOT file on the disc, and this might be created as follows (if you are uncertain on !BOOT files, please consult your disc manual):

```
*OPT 4,3
*BUILD !BOOT
*BASIC
*TV <whatever parameters you want>
CHAIN"MENU"
<Escape>
```

The *TV command listed above is optional. If you normally use !BOOT files to set your preferred parameters, then these could be included as well.

The menu program has a couple of minor limitations, which are most obvious when it is used in extreme circumstances.

1) The program does not incorporate facilities for programs that need to be moved down in memory (although it has been designed for easy updating if you so desire). You could possibly amend the program to incorporate (or CHAIN) a 'movedown' routine, or to alter the value of PAGE to &1200 if necessary. (This could be included as part of the !BOOT file).

2) The order in which the various disc drives are scanned prevents the use of some combinations of drives (e.g. 0, 2, 3).

PROGRAM NOTES

The program is well structured, and has been specifically designed with an "open" structure to facilitate easy addition of any extra code required. Line 1140 is where the selection of the drives is made. By altering this to '0 TO 1', you can scan drives 0 and 1, while '0 TO 2 STEP 2' scans both sides of the first disc drive. The program, as listed here, will scan just drive 0. The program will work on either 40 or 80 track drives with no problems.

Although only directories "B" and "R" are used by this version, the program has been designed to allow other directories to be used, and to have different meanings attached to them. Line 2010 is a DATA line holding the number of directories to be scanned, then the letter for each directory. If you add any more directories, make sure you update the number of directories as well.

In order that the program can remember in which directory each program is found, it assigns a number to each of the directories it scans. For example, the current version allocates 0 to "B" and 1 to "R", and this is tested for at lines 160 and 170. If you add another directory option to the DATA statement at line 2010 (add it to the end, and update the 1 to a 2), then a new line 175 could check for type%(A%) being 2, and select your own option as a result. The

→

```
  10 REM PROGRAM MENU
  20 REM AUTHOR PETER HUNTER
  30 REM VERSION B1.2
  40 REM BEEBUG AUG/SEPT 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 MODE 7
 110 PROCreadnames
 120 PROCdisplay
 130 A%=FNprogram
 140 PROCoscli("DRIVE "+MID$(name$(A%)
,2,1))
 150 PROCoscli("DIR $")
 160 IF type%(A%)=0 CHAIN name$(A%)
 170 IF type%(A%)=1 PROCoscli("RUN "+n
ame$(A%))
 180 END
 190 :
1000 DEF PROCreadnames
1010 num%=100
1020 RESTORE
1030 READ N:N=N-1
1040 DIM type$(N)
1050 FORI%=0TON
1060 READ type$(I%)
1070 NEXT
```

→

```
1080 DIM name$(num%)
1090 DIM type%(num%)
1100 DIM cb% 16
1110 DIM os% 30
1120 DIM buf% num%*8
1130 N%=0
1140 FOR drive=0 TO 0
1150 FOR type=0 TO N
1160 PROCread(type$(type),type,drive)
1170 NEXT
1180 cb%!1=buf%
1190 PROCosgbpb(5,cb%)
1200 buf%?13=13
1210 TITLE$=TITLE$+CHR$131+$(buf%+1)
1220 NEXT
1230 ENDPROC
1240 :
1250 DEF PROCread(A$,type%,drive)
1260 CLS
1270 PRINTCHR$130"Scanning Drive ";dri
ve
1280 PROCoscli("DRIVE "+STR$drive)
1290 PROCoscli("DIR "+A$)
1300 FOR I% = buf% TO buf%+num%*8-3 ST
EP 4
1310 !I%=0
1320 NEXT
1330 cb%!1=buf%
1340 cb%!5=num%
1350 cb%!9=0
1360 PROCosgbpb(8,cb%)
1370 Z%=buf%
1380 IF ?Z%<>7 ENDPROC
1390 REPEAT
1400 ?Z%=13
1410 Z%=Z%+8
1420 UNTIL ?Z%<>7
1430 ?Z%=13
1440 M%=buf%+1
1450 REPEAT
1460 IF $M%<>"MENU"+STRING$(3,CHR$32)
AND $M%<>"!BOOT"+STRING$(2,CHR$32) name
$(N%)=":"+STR$drive+"."+A$+"."+$M%:type
%(N%)=type%:N%=N%+1
1470 M%=M%+8
1480 UNTIL M%?7<>13
1490 ENDPROC
1500 :
1510 DEF PROCoscli(A$)
1520 $os%=A$
1530 LOCAL X%,Y%
1540 X%=os%
1550 Y%=X% DIV 256
1560 CALL &FFF7
1570 ENDPROC
1580 :
1590 DEF PROCosgbpb(A%,X%)
1600 LOCAL Y%
1610 Y%=X% DIV 256
1620 CALL &FFD1 : REM OSGBPB
1630 ENDPROC
1640 :
1650 DEF PROCdisplay
1660 VDU 23,1,0;0;0;0;12
1670 FOR Y%= 1 TO 2
1680 PROCcntr(CHR$141+CHR$129+CHR$157+
CHR$131+"MENU  "+CHR$156)
1690 NEXT
1700 A$=TITLE$:IFA$="" GOTO 1680 ELSE
TITLE$=""
1710 FORI%=1TOLENA$
1720 IFASCMID$(A$,I%,1)>31 TITLE$=TITL
E$+MID$(A$,I%,1)
1730 NEXT
1740 PRINT'
1750 PROCcntr(TITLE$)
1760 IF N%=0 PRINT'CHR$129"No suitable
 files on disc.":END
1770 PRINT'"Press the key to select th
e program of""your choice"'
1780 N%=0
1790 REPEAT
1800 PRINT TAB(-20*(N%MOD2=1),N%DIV2+9
)CHR$133CHR$(N%+65)CHR$131MID$(name$(N%
),6)
1810 N%=N%+1
1820 UNTIL name$(N%)=""
1830 ENDPROC
1840 :
1850 DEF FNprogram
1860 *FX15,1
1870 REPEAT
1880 G%=GET
1890 UNTIL G%<(N%+65) AND G%>64
1900 =G%-65
1910 DEF PROCdefkey(A$)
1920 *FX18
1930 PROCoscli("KEY 0 "+A$)
1940 *FX138 0 128
1950 ENDPROC
1960 :
1970 DEF PROCcntr(A$)
1980 PRINTTAB(20-LENA$DIV2)A$
1990 ENDPROC
2000 :
2010 DATA 2,"B","R"
```

procedures PROCoscli (to perform an OS (or *) call of the string passed), and PROCdefkey (which defines a key to the string passed, and then inserts the code for this key into the keyboard buffer) are both available to help make any additions easier. For example, MRUN (from Watford DFS) is easily added by combining MRUN and the file specification, and passing this via PROCoscli at line 175 say. Overall, this is both a flexible and most useful utility for disc users.

MICRONET - ONE MILLION FRAMES ON

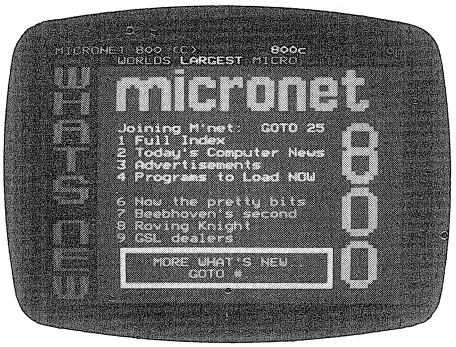
by James Fletcher

Micronet gives the home user access to huge volumes of information, and software, but at a price. Now that Micronet is being claimed a major success, James Fletcher evaluates the service by recounting his own experiences, and describes some of the most recent developments.

The advertisements were compulsive reading, promising a whole new world of opportunities for microcomputer users, access to a huge database of information, hundreds of free software programs, and communication with other like-minded computer nuts. Micronet 800 really did seem to offer everything. "It's fun, friendly, and inexpensive to run", continued the beguiling prose, which went on to describe how a low-cost subscription could turn the BBC micro into the hub of the brave new world of information technology, offering a whole range of services that never before would have been possible in the home.

It all sounded to be just what I needed to bring back the sparkle to my rather jaded love affair with my micro, which had been more than sated with a continuous diet of space-invaders and adventures where I could never get out of the cold, dank cave, whichever way I tried. Returning the coupon in the magazine brought an even more impressive set of leaflets and the knowledge that as a special introductory offer all this technology could be mine for about fifty pounds. The cheque went off by the next post, and surprisingly for an industry that has gained a terrible reputation for delays on orders, a well-packed parcel arrived within just a few days, containing everything needed to turn my BBC into a Micronet terminal.

The main part of the hardware is a box about 10"x3"x2" with holes into which your telephone handset fits, the acoustic coupler. Professional couplers, which normally cost at least three times the price of the Micronet coupler, usually have the holes for microphone and earpiece thick with foam rubber so as to ensure a good seal. The Micronet coupler is rather less well padded, and the seals around the telephone mouthpiece and earpiece are of a fairly stiff plastic which makes it less easy to ensure a perfect fit. I found it necessary on several occasions to push the handset very hard into the coupler to avoid problems with data errors which showed up on the computer screen as typing mistakes. A colleague with a Trimphone found it very difficult to get satisfactory results, so if you have this type of telephone you will be well-advised to use one of the directly wired 'modem' units which are now available.

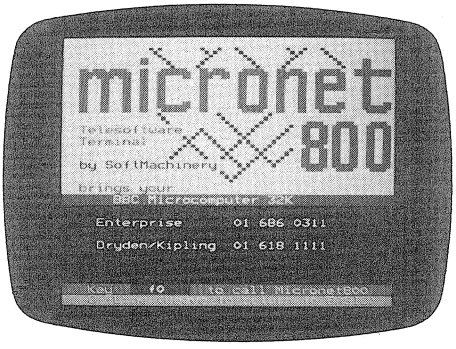


The initial offer price on the acoustic coupler type of modem has now ended, but it is now available with ROM-based software for around £75, which still seems reasonably good value for money when you consider how little proprietry software you could buy for that amount of money. There are two other modems available from Micronet, both of which need to be directly connected to a telephone line (By B.T., not you, of course!). Both of these modems allow 1200/75 baud transmission rates for Prestel/Micronet and also 1200/1200 baud for micro-to-micro communication, and both come complete with ROM-based software. The Modem 1000

costs £93.65 while the Modem 2000 costs £108.65 but offers auto-switching when used for communication between users and a few other features which make the system more user-friendly.

The acoustic coupler that I use came complete with a cable suitable for plugging directly into the RS423 socket at the back of the BBC micro, so it was the work of only a few moments to connect up the hardware and be ready for communicating with the world. The software provided on tape was soon transferred to disc, and worked well although it didn't take long to realise that the system could be made more user-friendly with a more "automatic" type of operation. I presume that the ROM-based software provided with the latest models will overcome some of the initial difficulties that I found. David Graham gave Beebug readers a good account of the various facilities that Micronet provides in his article in Vol.2 No.4 so I'll concentrate on the Telesoftware side of Micronet, which allows a wide range of computer programs to be downloaded directly into the BBC micro.

The Micronet information that is currently available off-screen says



that over 200 programs are available for the BBC, but I was able to find only a fraction of this number, and some of these seemed too simple to be worthwhile, although they would be suitable for beginners looking for something simple to download. Once you have chosen your program, downloading is simple; all you have to do is to follow the instructions, which in my case said "Press F5".

Having done this the pages start to appear, just like pages of teletext. Each program uses the first frame of its program as a header which gives the program name and its length. There is also a check number which serves to ensure that the program has been received correctly. If any errors have been picked up during transmission or reception of the program, the check value will not agree with the value transmitted with the header page, and an appropriate warning appears on the screen. The system automatically tries to reload an incorrectly-received program, but if it receives a frame incorrectly after three attempts it asks you "try again ? -Y/N ".

I found that although excellent reception was obtained most of the time it was possible to get a poor telephone line, just as sometimes happens when the phone is being used for ordinary talk. Under these conditions it proved impossible to download a program correctly no matter how many times I tried, and the only solution was to ring off and re-dial.

Micronet telesoftware programs are in Basic, but have been crammed together to improve transmission efficiency. The programs are divided into blocks, one per Micronet frame. Spaces in the program are replaced by the 3/4 sign and if it proves necessary to actually use a 3/4 sign in a program it has to be preceded by | (ASCII character 124) to prevent confusion. If you have to make use of | in a program then this must be entered as |E. ASCII 124 also has other uses in Micronet telesoftware, |L indicating the end of a line and |F showing the end of a program or of a file.

I managed to download a good deal of the free software on Micronet, but refrained from buying the commercially produced offerings using a credit card for payment. In these days when the postal services can take for ever to deliver a packet it may very well be a good idea to buy your software over the telephone line, (and when the BEEBUG editor offers to pay the bills I will be glad to give you a complete run-down of all the programs I can buy via the Micronet telesoftware service!). One big advantage of a live downloading

service is that an up-to-the-minute catalogue of software can be offered, and I noticed that a BBC Basicode converter program was available on Micronet in time to be used with their Chip-Shop radio programmes whereas there was a considerable delay if the converter was ordered by post from the BBC.

Several weeks of playing convinced me that Micronet is great but, and it's a big but, my telephone bill at the end of the quarter was horrendous. Not only do you pay to subscribe to Micronet, about £1 per week, but you pay the cost of the telephone call all the time that you are connected, as well as a further computer connection charge if you should dare to use the system during business hours or before lunch on a Saturday. The free software is obviously useful, as are all the other facilities that Micronet and Prestel offer, but no father can afford to risk bankruptcy by giving his children access to this most modern of the new technologies. I think that the old expression 'money down the drain' may well be replaced by 'money down the phone' if Micronet catches on!

MICRONET - RECENT DEVELOPMENTS

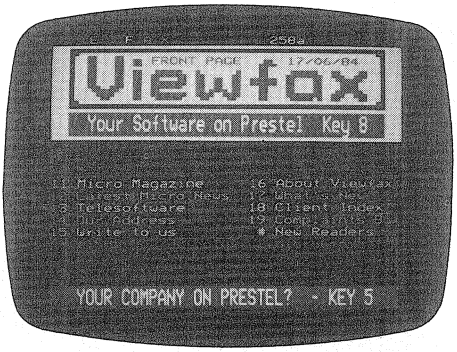
Recognising the success of Micronet, B.T. has launched a new service under the ' Prestel Microcomputing ' banner, and Micronet continues as just one part of this new service, which brings together all the services connected with home computing, making them much easier for the user to find.

Existing Micronet members automatically become paid-up members of 'Prestel Microcomputing', whereas new recruits will not join Micronet as such, but will subscribe to 'Prestel Microcomputing' and receive Micronet as just one part of the package, which costs £8 each quarter. Micronet currently provides all the features of a printed computer magazine, which is not too surprising when you learn that its owners are in fact members of a company that publishes several of the computer magazines that grace the newsagents' shelves. Computer news can be far more up-to-date than in printed magazines and Micronet proudly boasts that it had news of the QL launch on Micronet within an hour of the first announcement. (Micronet cannot, however, work miracles - they couldn't actually get their hands on a QL!)

The last few weeks have seen a considerable increase in the amount of software available via Micronet, and Acorn Computers now have quite a few pages on the system. One of these is called ' Talking back to Acorn', which should help to improve customer communications; there is no guarantee that they will actually reply, of course!

Other services available on the Prestel Microcomputing database include 'Clubspot' and 'Viewfax'. 'Clubspot' is a service operated by the Association of Computer Clubs, which provides individual databases for computer clubs around the country where they can make information available about any aspect of their clubs' work. Not only can such a database be useful for the members of each particular club, but non-members can get a broader idea of what is happening in other clubs around the country. There is no charge to clubs for these pages.

Viewfax claims to be 'the professional micro database', and offers a wide range of pages which should appeal to more advanced home computer users. Viewfax was one of the pioneers of using Prestel for computer software and its entry into the Prestel Microcomputing stable should bring its products to the notice of far more computer users. Viewfax intends to sell computer peripherals as well as software, and claims that its prices will be cheaper than the shops; most of its current offerings seem to be



utility packages suitable for home finance or small businesses.

Other databases within the Prestel Microcomputing fold are provided by 'Prestel Education', 'The National Computing Centre', 'Spectrum UK', and 'VNU Publishers', and other companies such as 'Transam' and 'DRG Business Machines' are soon to join them.

There is no doubt that the hundreds of pages available from Prestel Microcomputing are interesting and great fun to access, but having spent many hours 'playing' with the Prestel services I have come to the conclusion that most of the pages contain fairly trivial information and lack the 'meat' that longer, written articles in magazines such as Beebug can provide. Prestel's many screenfuls, each of which can only carry about 120 words, seem a poor substitute for a chunky monthly magazine, even though the printed version can never be as up-to-date as its electronic companion.

Prestel Microcomputing offers some great communications features. Using 'Mailbox' you can contact many other Prestel users, sending either your own message or one of a wide range of pre-formatted on-screen greetings cards which make imaginative use of Prestel's graphics capabilities. Messages can be left on various 'bulletin-boards' and these allow you to make contact with other enthusiasts around the country.

If you have problems with a computer company, and you want to attract the immediate attention of the managing director, what better than to send him an official-looking Telex message? Until recently Telex has been limited to those businesses that could afford to hire expensive equipment from B.T. Prestel now allows Microcomputing subscribers to send Telex messages of up to 100 words to any UK Telex subscriber for only 50p. I tried it, and it works! Within minutes of the message being despatched from the keyboard of your home micro it is being printed out in the office of the MD's secretary.

Another fascinating extension of the Prestel network is available by using the 'Gateway' service. This allows you to gain access to a computer belonging, not to B.T., but to an information provider who has his own private database. When you pass through the Gateway, and you can only do this if you hold the correct electronic key, the Prestel computer sends instructions to the information provider's computer, which will then send the required information, via the Prestel computer, into your home. Business users such as doctors, travel agents and mail-order companies are already finding the system invaluable, and this type of service could be readily extended to the home user as well.

Prestel Microcomputing looks to be set for expansion. The newly-organised service makes it easy for home computer users to see all the information that is available about their hobby from a wide range of different information providers. With something like two million home computers in the UK, Prestel believes that many other groups will be seeking to come under the 'Microcomputing' umbrella. I can certainly say that if you do join the club, you will open up many new avenues for you and your computer to explore, and you will have the satisfaction of knowing that you are right at the forefront of the information-technology revolution.

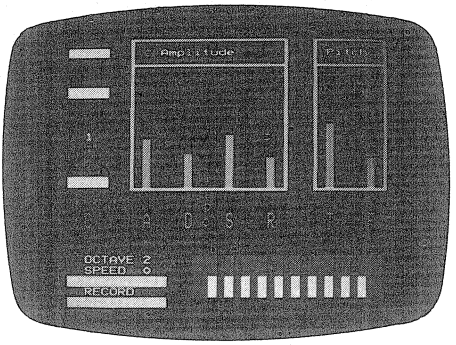
[For further information on Prestel Microcomputing contact Micronet at Scriptor Court, 115 Faringdon Road, london EC1R 3AD. Tel. 01-278-3143]

A SCREEN DRIVEN MUSIC SYNTHESIZER (32K)

by D.J. Westbrook

The sound generating capabilities of the BBC Micro are pretty sophisticated when compared with other home micros. Using these, this program will turn your Beeb into a very practical music synthesizer, by the use of a well designed screen display.

A modern music synthesizer popularly offers, amongst its standard features, the facility to build a particular sound by shaping its envelope, a keyboard to play the sounds as notes, and memory to record a sequence of notes played. This program makes use of the BBC micro's own resources to provide all these features and thereby lets you program the machine to play music very easily, as though it were a synthesizer. To support this, there is a very useful, graphic colour display of the synthesizer's configuration; the envelope selected, the envelope's shape, mode of operation, etc.



HOW TO USE THE SYNTHESIZER

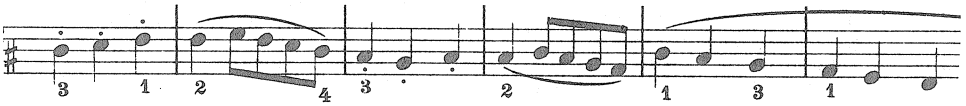
Run the program and immediately, the default status of the synthesizer will be displayed. Across the middle of the screen are the parameters that may be adjusted to contruct a sound. Each is denoted by the initial letter of its operational effect and selection is made by use of the left and right cursor keys.

The options are 'A', 'D', 'S', and 'R' which respectively, are used to set an envelope's Attack, Decay, Sustain and Release times (see the User Guide page 247 for a graphical explanation of these terms); then 'T', 'F', shown separately, for Tremolo depth and Frequency of the tremolo pitch variation. Above each, a bar chart indicates the setting of these, which is adjusted by the use of the up and down cursor keys. On the far left of the screen there is a similar option for selecting one of the four envelopes that are available for programming (envelope Ø is fixed to percussive sounds however).

The bottom two rows of keys on the computer keyboard form the notes of the synthesizer keyboard. They are duplicated on the screen and identify themselves when pressed. The white notes are represented by 'Z' to '/', and 'S', 'D', 'G' etc., are the black notes.

When you want to record a tune, press function key fØ to place the synthesizer in record mode. This mode is shown in the status display at the bottom left of the screen. Anything that you play after that will be recorded until function key f1, replay, is pressed; or the record key is pressed again (which simply toggles recording mode off); or the end of the recording memory is reached (which automatically switches off recording mode).

There are two more facilities that need mentioning. Numeric keys '1' to '4' set the octave for the keyboard, and function keys f8 and f9 change the speed of replay of notes - slower or



faster. (The chosen speed is shown on the display at the lower left of the screen).

```
Synthesizer key functions:
            f0 - select record mode
            f1 - select replay mode
            f8 - slower
            f9 - faster
  1,2,3 or 4 - select octave
 Cursor right - move cursor right
  Cursor left - move cursor left
    Cursor up - increase
  Cursor down - decrease
```

Though we have published programs before to help the user experiment with the Beeb's sound and envelope facilities, this program makes it very easy to produce a variety of musical results. The keen user might add a simple editor for amending music stored in the synthesizer, and an associated filing mechanism so that pieces of music generated in this way could be saved for future performances. He might also make it possible for the playing of a stored piece, accompanied by the user at the keyboard whilst simultaneously recording the duet - a kind of computerised multi-track recording but without a tape recorder.

PROGRAM NOTES

The synthesizer is initially configured by PROCinit which sets up the keys for various functions and also the default envelopes. One variable here called 'limit%', claims the area of memory available to the recording function. You might extend the size of this area simply by increasing its value (the maximum is about 2300 with PAGE at &1900).

The main part of the program, as you might expect, is structured around a continuous loop, responsible for sampling the keyboard for the player's response, playing the required sounds and maintaining an up-to-date screen display. The pitch of a note is interpreted by calculating its distance away from bottom 'B' in terms of semitones. Sounds generated in direct response to keyboard input in this way, are performed by PROCsound. If record mode is selected, then this instigates the action of PROCstore which pokes data about pitch and envelope into memory. A note's duration is stored in an array (durn%). Recorded music is replayed from memory by PROCplayback.

LIST OF PROCEDURES

PROCinit	- keyboard, envelope initialisation
PROCscreen	- display setup
PROCchangevalue	- alters an envelope
PROCbeep	- beeps and pauses
PROCsetchannel	- sets up the current channel
PROCprintcolumn()	- draws a bar on the chart
PROCcursor()	- draws or deletes the cursor
PROCcursorshift	- moves the cursor
PROCupdate	- refreshes the current envelope
PROCoctave	- sets the octave
PROCrecordswitch	- toggles the record mode
PROCkeyon()	- displays the note played
PROCkeyoff	- erases the note played
PROCsound()	- plays a note from the keyboard
PROCplayback	- replays a tune from memory
PROCspeed()	- changes the speed of replay
PROCwait()	- delay loop
FNoffset()	- calculates screen position for column

```
   10 REM Program SYNTH
   20 REM Author  DAVID WESTBROOK
   30 REM Version B1.0
   40 REM BEEBUG  Aug/Sept 1984
   50 REM Program Subject to Copyright
   60 :
  100 ON ERROR GOTO 380
  120 MODE 7:PROCinit:PROCscreen
  150 PROCsetchannel
  160 REPEAT
  161 in=INKEY(0)
  170 key=INSTR(note$,CHR$(in))
  180 IF key THEN PROCkeyon(key):PROCso
und(key):PROCkeyoff
  190 IF in=136 OR in=137 THEN PROCcurs
orshift
  200 IF in=138 OR in=139 THEN PROCchan
gevalue
  210 IF in>48 AND in<53 THEN PROCoctave
  220 IF in=200 THEN PROCrecordswitch
  230 IF in=201 THEN PROCplayback
  240 IF INKEY(-119) THEN PROCspeed(-1)
```

```
  250 IF INKEY(-120) THEN PROCspeed(1)
  260 UNTIL forever
  270 END
  280 :
  380  REM------ Error handling -------
  390 ON ERROR OFF
  400  MODE7
  410 *FX4
  420 *FX12
  430 *FX21
  440 *FX225,1
  450 IF ERR<>17 THEN REPORT:PRINT" at
";ERL
  460 END
  470 :
 1000 DEF PROCinit
 1010 LOCAL I%,J%
 1020 *FX4,1
 1030 *FX11,1
 1040 *FX12,1
 1050 *FX202,32
 1060 *FX225,200
 1070 limit%=500
 1080 DIM notename$(11),P%(6),maxP%(6),
minP%(6),minF%(25),E%(4,14),env% limit%
,pitch% limit%,durn%(limit%)
 1090 note$="ZSXDCVGBHNJM,L.;/"
 1100 black$="SDGHJL;"
 1110 up$=CHR$11+CHR$8
 1120 FOR I%=0 TO 11:READ notename$(I%)
:NEXT
 1130 DATA "B ","C ","C#","D ","D#","E
","F ","F#","G ","G#","A ","A#"
 1140 FOR I%=0 TO 6:READ maxP%(I%),minP
%(I%):NEXT
 1150 DATA 3,0, 125,1, 124,1, 124,1, 12
4,1, 25,0, 13,1
 1160 FOR I%=1 TO 4
 1170 FOR J%=2 TO 14
 1180 READ E%(I%,J%)
 1190 NEXT
 1200 ENVELOPE I%,E%(I%,2),E%(I%,3),E%(
I%,4),E%(I%,5),E%(I%,6),E%(I%,7),E%(I%,
8),E%(I%,9),E%(I%,10),E%(I%,11),E%(I%,1
2),E%(I%,13),E%(I%,14)
 1210 NEXT
 1220 DATA 1, 0,0,0,25,50,25, 125,-2,0,
-1, 125,53
 1230 DATA 1, 0,0,0,25,50,25, 125,-2,0,
-1, 125,53
 1240 DATA 1, 1,-1,1,3,6,3, 1,-2,0,-1,
125,100
 1250 DATA 1, 0,0,0,25,50,25, 7,-2,0,-1
, 125,53
 1260 FOR I%=0 TO 25:READ minF%(I%):NEXT
 1270 DATA 1,13,9,7,6,5,4,4,3,3,3,3,2,2
,2,2,2,2,2,2,2,2,2,2,1
 1280 r$=CHR$145:g$=CHR$146:y$=CHR$147:
b$=CHR$148:m$=CHR$149:c$=CHR$150:w$=CHR
$151
 1290 pn=0:P%(pn)=1:en=pn+1:basenote=48
 1300 event%=0:recording=FALSE:tempo%=5
 1310 forever=FALSE
 1320 ENDPROC
 1330 :
 1340 DEF PROCscreen
 1350 LOCAL I%,J%,cx%
 1360 CLS:VDU 23,1,0;0;0;0;
 1370 REM   ----- Frames -----
 1380 lside$="j"+y$:rside$=c$+"5"
 1390 FOR I%=1 TO 13
 1400 PRINTTAB(7,I%)c$;lside$
 1410 PRINTTAB(26,I%)rside$
 1420 PRINTTAB(30,I%)lside$
 1430 PRINTTAB(38,I%)rside$
 1440 NEXT
 1450 PRINTTAB(7,0)c$;"j";STRING$(18,"£
");"5  j";STRING$(8,"£");"5"
 1460 PRINTTAB(7,2)c$;"j";STRING$(18,"p
");"5  j";STRING$(8,"p");"5"
 1470 PRINTTAB(7,14)c$;CHR$162;STRING$(
18,"£");"!  ";CHR$162;STRING$(8,"£");"!"
 1480 REM   ----- Labels -----
 1490 PRINTTAB(11,1)CHR$135"Amplitude";
TAB(31,1)CHR$135"Pitch"
 1500 FOR I%=1 TO 13 STEP 4
 1510 PRINTTAB(1,I%)CHR$157;CHR$135;3-I
%DIV4;"  "CHR$156
 1520 NEXT
 1530 FOR I%=20 TO 24
 1540 PRINTTAB(1,I%)CHR$157;CHR$135;STR
ING$(10," ")CHR$156;
 1550 NEXT
 1560 PRINTTAB(3,20)"OCTAVE ";(basenote
DIV 48)+1
 1570 PRINTTAB(3,21)"SPEED  ";tempo%-5
 1580 PRINTTAB(3,22)"ENV/PLAY"
 1590 PRINTTAB(3,23)"RECORD"
 1600 PRINTTAB(3,24)"REPLAY";
 1610 PRINTTAB(0,20)b$
 1620 PRINTTAB(0,21)c$
 1630 PRINTTAB(0,22)g$
 1640 FOR I%=16 TO 17
 1650 PRINTTAB(1,I%)CHR$141;y$;"C";SPC(
6);"A";SPC(4);"D";SPC(4);"S";SPC(4);"R"
;SPC(6);"T";SPC(4);"F"
 1660 NEXT
 1670 REM  ----- Cursor -----
 1680 PRINTTAB(0,15)r$CHR$136;TAB(0,18)
r$CHR$136
 1690 PROCcursor(-1)
 1700 REM  ----- Keyboard ---
 1710 FOR I%=20 TO 23
 1720 PRINTTAB(15,I%)g$;CHR$157;CHR$(14
8-3*(I%>21));TAB(39,I%)CHR$156;
 1730 NEXT
 1740 FOR I%=20 TO 21
 1750 PRINTTAB(19,I%)CHR$255;SPC(1);CHR
$255;SPC(3);CHR$255;SPC(1);CHR$255;SPC(
1);CHR$255;SPC(3);CHR$255;SPC(1);CHR$25
5;
```

```
1760 NEXT
1770 FOR I%=0 TO 9:FOR J%=22 TO 23
1780 PRINTTAB(18+2*I%,J%)CHR$255;
1790 NEXT,
1800 ENDPROC
1810 :
1820 DEF PROCchangevalue
1830 LOCAL inc
1840 IF pn>4 OR pn<1 THEN inc=1 ELSE i
nc=4
1850 IF (P%(pn)>(maxP%(pn)-inc) AND in
=139) OR (P%(pn)<(minP%(pn)+inc) AND in
=138) THEN PROCbeep:ENDPROC
1860 P%(pn)=P%(pn)-inc*(in=139)+inc*(i
n=138)
1870 IF pn=0 THEN PROCsetchannel ELSE
PROCupdate(pn)
1880 *FX21
1890 ENDPROC
1900 :
1910 DEF PROCbeep
1920 SOUND 1,-10,200,1:*FX21
1930 PROCwait(5):*FX21,5
1940 ENDPROC
1950 :
1960 DEF PROCsetchannel
1970 LOCAL I%
1980 ch=P%(0)
1990 PRINTTAB(0,13-4*(en-1))w$
2000 PRINTTAB(0,13-4*ch)r$
2010 en=ch+1
2020 P%(1)=125/E%(en,9)
2030 P%(2)=-(125-E%(en,14))/E%(en,10)
2040 P%(3)=E%(en,14)
2050 P%(4)=-E%(en,14)/E%(en,12)
2060 P%(5)=E%(en,3)*E%(en,6)
2070 P%(6)=100/(4*E%(en,2)*E%(en,6))
2080 FOR I%=1 TO 6
2090 PROCprintcolumn(I%)
2100 NEXT
2110 ENDPROC
2120 :
2130 DEF PROCprintcolumn(par)
2140 LOCAL inc,colx,blocknum,rem,top
2150 IF par>4 OR par<1 THEN inc=1 ELSE
inc=4
2160 colx=FNoffset(par)+1
2170 blocknum=(P%(par)-1)DIV(3*inc)
2180 rem=(P%(par)-1)MOD(3*inc)
2190 top=112-12*(rem>=inc AND rem<2*in
c)-143*(rem>=2*inc)
2200 PRINTTAB(colx,13);STRING$(blocknu
m,CHR$255+up$)+CHR$(top)+STRING$(10-blo
cknum,up$+" ")
2210 ENDPROC
2220 :
2230 DEF PROCcursor(on)
2240 LOCAL cx%,c1$,c2$
2250 cx%=FNoffset(pn)
2260 IF on THEN c1$="7£k":c2$="upz" EL
SE c1$=STRING$(3," "):c2$=c1$
2270 PRINTTAB(cx%,15)c1$;TAB(cx%,18)c2$
2280 ENDPROC
2290 :
2300 DEF PROCupdate(N)
2310 PROCprintcolumn(N)
2320 E%(en,9)=125/P%(1)
2330 E%(en,10)=-(125-P%(3))/P%(2)
2340 E%(en,12)=-P%(3)/P%(4)
2350 E%(en,14)=P%(3)
2360 E%(en,6)=100/(4*E%(en,2)*P%(6))
2370 E%(en,7)=2*E%(en,6)
2380 E%(en,8)=E%(en,6)
2390 E%(en,3)=P%(5)/E%(en,6)
2400 E%(en,4)=-E%(en,3)
2410 E%(en,5)=E%(en,3)
2420 ENVELOPE en,E%(en,2),E%(en,3),E%(
en,4),E%(en,5),E%(en,6),E%(en,7),E%(en,
8),E%(en,9),E%(en,10),E%(en,11),E%(en,1
2),E%(en,13),E%(en,14)
2430 maxP%(2)=E%(en,13)-E%(en,14)
2440 maxP%(3)=125-P%(2)
2450 minP%(3)=P%(4)
2460 maxP%(4)=E%(en,14)
2470 IF N=5 THEN minP%(6)=minF%(P%(5))
:P%(6)=minP%(6):PROCupdate(6)
2480 ENDPROC
2490 :
2500 DEF PROCcursorshift
2510 PROCcursor(0)
2520 pn=pn+(in=136 AND pn>0)-(in=137 A
ND pn<6)
2530 PROCcursor(-1)
2540 PROCwait(10):*FX21
2550 ENDPROC
2560 :
2570 DEF FNoffset(col%)
2580  =2+5*col%-2*(col%>0)-2*(col%>4)
2590 :
2600 DEF PROCoctave
2610 basenote=48*(in-49)
2620 PRINTTAB(10,20);in-48
2630 ENDPROC
2640 :
2650 DEF PROCrecordswitch
2660 recording=NOT recording
2670 IF recording THEN PRINTTAB(0,23)r
$;TAB(0,22)w$:TIME=0:event%=0
2680 IF NOT recording THEN PRINTTAB(0,
23)w$;TAB(0,22)g$
2690 PROCwait(20):*FX21
2700 ENDPROC
2710 :
2720 DEF PROCkeyon(keynum%)
2730 IF INSTR(black$,MID$(note$,keynum
%,1)) THEN colour$=b$ ELSE colour$=w$
2740 keyX%=16+keynum%-(keynum%>5)-(key
num%>12):keyY%=22+2*(colour$=b$)
2750 symbol$=notename$(keynum% MOD 12)
```

→

```
 2760 PRINTTAB(keyX%,keyY%)w$;LEFT$(sym
bol$,1);colour$
 2770 PRINTTAB(keyX%,keyY%+1)w$;RIGHT$(
symbol$,1);colour$
 2780 ENDPROC
 2790 :
 2800 DEF PROCkeyoff
 2810 PRINTTAB(keyX%,keyY%)colour$;CHR$
255;TAB(keyX%,keyY%+1)colour$;CHR$255
 2820 ENDPROC
 2830 :
 2840 DEF PROCsound(semitones)
 2850 LOCAL p,new
 2860 IF ch=0 THEN p=6-semitones MOD 3
ELSE p=4*semitones
 2870 IF recording THEN PROCstore(0,0)
 2880 SOUND &10+ch,en,basenote+p,-1
 2890 REPEAT
 2900 new=INSTR(note$,CHR$(INKEY(2)))
 2910 UNTIL new<>semitones
 2920 SOUND &1010+ch,0,0,-1
 2930 IF recording THEN PROCstore(en,p)
 2940 ENDPROC
 2950 :
 2960 DEF PROCstore(envelope,pitch)
 2970 IF event%=limit% THEN PROCrecords
witch:ENDPROC
 2980 event%=event%+1
 2990 durn%(event%)=TIME*tempo%
 3000 pitch%?event%=pitch
 3010 env%?event%=envelope
 3020 TIME=0
 3030 ENDPROC
 3040 :
 3050 DEF PROCplayback
 3060 LOCAL hsfc%,e%,I%
 3070 IF event%=0 THEN PROCbeep:ENDPROC
 3080 PRINTTAB(0,24)m$;
 3090 PRINTTAB(0,22)w$;TAB(0,23)w$:reco
rding=FALSE
 3100 PROCwait(15):*FX21
 3110 FOR I%=2 TO event%
 3120 TIME=0
 3130 IF env%?I% THEN hsfc%=&10+ch:e%=e
n:PROCkeyon((pitch%?I%)/4) ELSE hsfc%=&
1010+ch:e%=0:PROCwait(2)
 3140 SOUND hsfc%,e%,basenote+pitch%?I%
,-1
 3150 REPEAT UNTIL TIME*tempo%+6>durn%(
I%)
 3160 IF env%?I% THEN PROCkeyoff ELSE P
ROCwait(1)
 3170 IF INKEY(0)=201 THEN I%=event%
 3180 IF INKEY(-119) THEN PROCspeed(-1)
 3190 IF INKEY(-120) THEN PROCspeed(1)
 3200 NEXT
 3210 SOUND &1010+ch,0,0,0
 3220 PRINTTAB(0,24)w$;TAB(0,22)g$
 3230 PROCwait(10):*FX21
 3240 ENDPROC
 3250 :
 3260 DEF PROCspeed(change)
 3270 IF change=1 AND tempo%>12 OR chan
ge=-1 AND tempo%<2 THEN ENDPROC
 3280 tempo%=tempo%+change
 3290 PRINTTAB(10,21);tempo%-5;" "
 3300 ENDPROC
 3310 :
 3320 DEF PROCwait(delay)
 3330 LOCAL now
 3340 now=TIME
 3350 REPEAT UNTIL TIME-now>=delay
 3360 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

WORDWISE AND MCP40 PRINT SIZE - N.J.Gill

Printing on the Tandy MCP40 printer is done at the size last selected. So the required print size for Wordwise text can easily be set before entering Wordwise using Basic. If portions of the text require different sized text, then the function keys can be used in the following way *KEY0 VDU2,1,18:P."S0":P."A":VDU3:|M*W.|M This will print at size 0. Any other size can be selected by changing the "S0" to "Sx", where x is the required size. Remember to reset the line length (LLnn) in Wordwise after this operation to ensure that the text is printed out correctly.

REM STATEMENTS IN DATA STATEMENTS - B.Woods

Mrs. Sellick claimed in BEEBUG Vol.1, No.5 that a REM statement could not be placed at the end of a DATA statement. However, it may be inserted at the beginning of a DATA line provided that DATA is followed by a comma and the line is RESTORE'd before use. Here is an example:

```
10 REM staff DATA,NIGE,ALAN,DAVE,****
20 RESTORE 10
30 REPEAT:READ name$:PRINT name$:UNTIL name$="****"
40 END
```

BEEBUG Workshop

DELAY LOOPS

by Surac

This month we'll look at some of the various methods of implementing delay loops on the Beeb, and the relative merits of each. When writing large programs, it is very useful to have a set of standard procedures and functions to perform commonly used tasks. The routines presented here can easily have procedure and function definitions built around them to provide flexibility.

Perhaps the first type of delay that many programmers use is one based around a loop that repeatedly performs a calculation or an operation that achieves nothing but takes some time to process; the most common example of this being a FOR-NEXT loop:

```
FOR I = 0 TO 1000:NEXT I
```

What this will achieve is a delay of approximately half a second. Note that the value 1000 does not convey much meaningful information as to how long the loop will take. Further, the speed which this takes can vary dramatically, depending upon the configuration of your system. Running this on an Electron, a standard Beeb, and a Beeb using a second processor will result in different delays. Even using just a standard Beeb running another task under events (say the event driven music program published in BEEBUG Vol.3 No.2) could result in different timings. This method does not have any particular advantages, and has the two major disadvantages mentioned above. I therefore suggest that you avoid the use of this method.

The next three methods to be covered all have their good and bad points, and the choice of which one to use is very much dependent upon the application. The first of these is vulnerable to a key press while the second is not. The third represents a combination of the virtues of both routines.

The following method of implementing a delay is based upon the use of the Basic function INKEY. The version of this that we will use takes a given number of centiseconds (hundredths of a second), and waits for up to this length of time, but will exit early if a key is pressed, or there is a character waiting in the keyboard buffer. Note that any key pressed to exit from the delay will no longer remain in the keyboard buffer when the routine is left. Thus you couldn't use a single key press to terminate the delay and select, say, a menu option unless you specifically allocate a permanent (global) variable just to hold the value of the last key pressed. The sort of delay routine that might be written around this built-in function is as follows:

```
*FX15,1
dummy=INKEY(delay)
```

Note the *FX15,1 is to ensure that the keyboard buffer is empty before the delay starts. Without this, any character remaining in the keyboard buffer will cause an immediate exit from the delay. The variable "delay" must have been previously set to the maximum number of centiseconds that the delay should last for. This sort of delay is convenient where it is necessary to exit from the delay before it has waited the full length of time; for example a game might cycle around alternately printing the high scores and running a demonstration game. Between each of these it might pause for a certain length of time, when it would be possible for the user to start playing.

The next delay loop we will look at cannot be terminated prematurely by the user pressing a key (except Escape). This uses the in-built pseudo variable TIME to count a specific number of centiseconds. Again, if "delay" has been set to the duration of the pause,

the following code would achieve the required delay:

```
TIME=0
REPEAT
UNTIL TIME>delay
```

This form of delay loop might be used where readings are to be sampled from the analogue port every minute, and it is essential to avoid an accidental key press terminating the delay prematurely.

The third form of delay is really a hybrid between the last two, and was first used in BEEBUG in the extensions to ASTAAD. The code is:

```
*FX15,1
TIME=0
REPEAT
UNTIL TIME>delay OR ADVAL(-1)
```

As in the two previous examples, "delay" is the variable holding the maximum number of centiseconds for which the delay should last. The ADVAL (-1) tests the keyboard buffer to see if there are any characters present. If there are, then the delay loop will be immediately terminated, but the character will not be taken from the buffer. This is useful in situations where there is no easy way for one section of a program to determine the last key pressed and read in by another section; i.e. there is no easy way to implement a "global" variable for the last key press, as suggested for the "INKEY" delay above. By not actually removing the value from the keyboard buffer, we can be certain that any key pressed before the delay loop is finished will not be lost, and also that the delay will exit as soon as there is some more data for the program to act upon.

These last three routines will cater for most of the situations in which some form of delay is required within a Basic program.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

WATFORD-ACORN DFS DELETE

The Watford DFS up to and including version 1.3, generates an error when trying to delete a file that does not exist using an OSFILE instruction (&FFDD); the Acorn DFS 0.90 does not. (Which is the most useful?).

INVERSE CHARACTER DEFINITIONS - M.Quinion

Using the routine listed below, it is possible to generate characters in inverse video (reversing foreground and background colours) for highlighted text on screen for instance.

```
  10 OSWORD=&FFF1:DIM param 9
1000 DEF PROCrevvid(a%)
1010 ?paramblock=a%:A%=10:X%=paramblock MOD 256:Y%=paramblock DIV 256
1020 CALL OSWORD:VDU23,224:FOR B%=1 TO 8:VDU?(paramblock+B%) EOR &FF:NEXT
1030 VDU224
1040 ENDPROC
```

The routine looks up the definition of the character whose ASCII code is held in a% and its inverse definition is put into character 224. If this one is now printed (ie. PRINT CHR$224 or VDU224) then the reversed image of the original appears on screen.

SELF DELETING MOVE DOWN ROUTINE - C.A.M.Timmerman

This is a move down routine that after having moved the program in which it is included, deletes itself. It does however, make use of a function key.

```
  1 *K.0DEL.1,3|MF.I%=0 TO TOP-PA.S.4:I%!&E00=I%!1900:N.|M*T.|MPA.=&E00|ME.|M
RUN|M
  2 *FX138,0,128
  3 END
```

The routine should be placed at the start of any program in which it is used. (Remember to save it before running it!-Ed.).

ACORN'S BITSTIK GRAPHICS SYSTEM

Reviewed by Terry Hallard

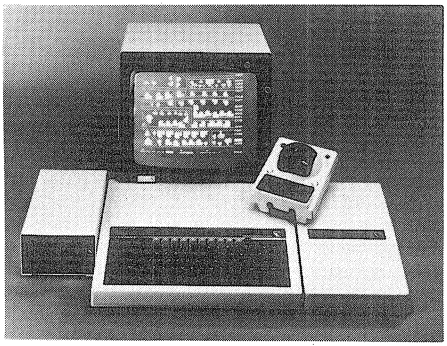
Following last month's report on Acorn's 6502 Second Processor, Terry Hallard now casts his expert eye over the Bitstik graphics system, which relies on the same second processor for its computing power. After all the publicity, does this come up to expectation?

Product : Acorn Robocom Bitstik
Price : £375 (inc. VAT)
Supplier: Vector Marketing and Acorn Dealers
Minimum configuration required:
BBC Model B; 6502 2nd. processor; dual 80-track double-sided disc drives and colour monitor.

The first time that I saw the Robocom Bitstik on the Apple micro, I was absolutely staggered by the graphics that appeared on the screen so easily and were manipulated so incredibly. At the time it was the only computer aided draughting package on the market for a personal micro, and with its superb range of facilities and screen manipulation, it was - well - just fantastic. This was in 1981, back in the very dim and distant past.

Now there are many micros on the market with graphic capabilities equal to or better than the Apple, and of course, the standard micro of today is the Beeb, for which a number of drawing packages have now been produced. So it was with some slight trepidation that I approached the job of delving into the latest version of the Bitstik, wondering if some of the shine has been taken off the one Beeb add-on that I have been looking forward to for so long. Inevitably I suppose, I have to say that for me it has a bit - principally by the great cost of the complete system. However, make no mistake, it is a very clever, impressive peripheral to have and I wish I had one of my own.

Bitstik comes in a neatly packaged cubical box which disgorges three special utilities discs, a ROM, a well written and very comprehensive handbook and the Bitstik itself. This is a plastic, cream coloured rectangular box which incorporates three buttons which, when pressed together in various combinations, send appropriate command messages to the computer. The box is moulded around a very large ball socket marked with graduations on which is set a three-way joystick - the ball movement axes being the usual X and Y movements and, by twisting the stick, a Z axis is obtained. This piece of equipment is ergonomically well designed and produced to a very high standard, with rubber feet to stop slip and also a moulded rubber handrest pad conveniently placed so that control of all movement and buttons is fairly easy. The joystick moves almost too smoothly and exhibits no 'bias'. Installation is simple (this presupposes that the second processor has already been set up) with the Bitstik merely being plugged into the analogue port at the rear of the Beeb.



When up and running, with the system master disc, the first thing to appear on the screen is a very comprehensive menu - one part along the bottom of the screen and the other up the right hand side. The right hand menu gives the system command words, such as 'draw' 'mode', 'paint' (which allows infilling of shapes with a choice from 16 patterns made up from the current four colours - like all fill commands it is prone to leaking all over the screen), 'copy', 'file', 'zoom' (of which more

later), 'erase', 'utilities', etc. The system always defaults to draw mode.

In the lower menu, the line of symbols indicates a range of line types, colours and drawing functions such as straight line, circle, arc and 'nib', the latter being a very interesting method of painting very broad lines or even wedge shaped areas with plain colour or a choice of 'texturing' parallel lines. This is great for the creation of bar charts or even block lettering.

The different line drawing parameters (line type, colour and drawing method) are easily chosen by simply moving the cursor to the relevant symbol. The cursor is easily and quite accurately moved by the joystick in the X Y axes and a constant flashing 'rubber band' ensures that one is always aware of the effect that any line might have. I found it slightly annoying at times that I could not switch this off. The cursor is a '+' symbol and is constantly attached to the last drawn point by the flashing 'rubber-band' line. If the red command button is pressed then a line instantly appears in place of the rubber-band line. If no line is needed then pressing another button deletes the rubber-band line and the position of the cursor at that moment is the new fixed basepoint. Circles are fantastically easy to draw - just touching the cursor to the appropriate symbol causes a flashing circle to appear. It is centred on the cursor and by twisting the Z axis joystick knob it can be made to grow or shrink. Such is the speed of the second processor that the flashing circle constantly follows the cursor with no discernible lag. Rapidly pushing the red button and moving and twisting the joystick causes a 'spray' of circles to appear on the screen. Great fun!

Arc drawing takes a little getting used to because it works by projecting an arc which is tangential to a straight line between the last two points indicated, and if you think that that complicates things you are quite right. The arc indicated is fixed at one end at the last point marked and will grow or shrink with any movement of the joystick, and will even whip backwards if one overdoes it - the effect is rather that of a skipping rope!

I have a personal set of drawing routines and effects that I look for in any drawing package and one yardstick that I apply is ease and accuracy of ellipse generation and manipulation - the ellipse being vitally important in any kind of formal pictorial technical illustration. Other criteria I use are ease of construction of isometric drawings and aids for accurate orthographic drawing. The Bitstik software has a splendid set of grids which can be called up from the vertical right hand menu. These can be set at various angles and the cursor constrained to move in only two pre-set directions and only in multiples of fixed increments, these being recorded as you go in a new menu which appears on the right, replacing the other.

Isometric is one of these preset grids and away one goes - that is until one wants an ellipse! Here we have a splendid, sophisticated drawing package and back we go to the Stone Age, drawing ellipse approximations by the four-arc method! Not only that, but because of the peculiar tangential method of generating the arcs, mentioned earlier, this takes a tediously long time.

With regard to the other grids, admittedly I did not spend a lot of time exploring them, but apart from right angled ones for orthographic projections, I cannot think of a lot of uses for two-way grids at other angles. They would aid the creation of parallelograms I suppose, but otherwise I feel that movement along two lines of direction only is too limiting.

All in all the actual drawing techniques employed appear to me to have been produced by some very clever programmers with minimal reference to actual practicing draughtsmen and their real needs. However, when one has completely absorbed the system techniques and short cuts, one can use the package to produce superb drawings. But it does need a lot of practice - and the manual tends to admit this by giving a very extensive course of

'Teach Yourself Bitstik' in thirteen fairly involved lessons.

Two drawbacks to producing accurate, detailed technical drawings on a micro - lack of screen drawing area and only moderate resolution compared with mainframe CAD systems - are, however, dealt with in breathtaking fashion.

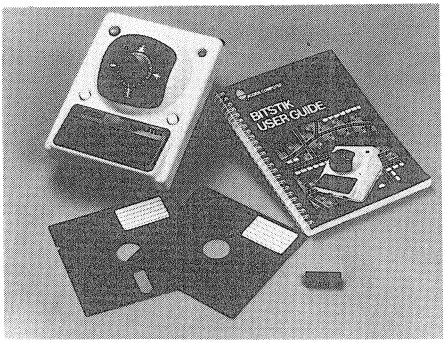
This is by means of the ZOOM command. When it is selected from the menu at the side, the cursor immediately breaks up into a set of four little 'Right Angle' marker symbols, each indicating one of the corners of a square. This square can be enlarged or shrunk by twisting the 'Z' axis knob, and can be moved anywhere on the screen by normal joystick movement. When positioned over part of the drawing, pressing the red button causes the drawing to vanish and the part selected by this framing reappears, blown up to full screen size. Fine work can now be done much more accurately at this larger scale and when complete the original drawing is recalled. Now all the added detail is incorporated. It is possible to zoom, draw, zoom, draw, zoom, etc. many times, so that we can get detail nested into finer and finer proportions. The now classic example of this process is of the view of the Galaxy, then the Solar System, then Earth, and so, in stages, right down to the compound eye of a fly somewhere in London.

ZOOM really is a tremendous asset in producing highly detailed technical drawings and illustrations and some of the possibilities are fantastic. Imagine a motoring map of the country that you use at large scale for motorway journeys and then, when you reach a town, the dot which represents it expands to become a detailed street map, with fine detail and more information available if needed. The effect of ZOOM is as incredible as that.

One cannot store a 'zoomed' screen in the main library, only the full detailed picture. But one can hold in a temporary store two zoom frames at a time, which could prove useful if a lot of 'to and fro' working is to be done. Also, whilst in zoom mode, one can call PAN which allows panning to areas adjacent to the current screen. So if work was needed in detail on a piece that extends over the edge of the screen, one does not have to go all through the selecting and zoom, zoom, zoom again, but merely 'hunts' sideways in the same magnification mode until it is found. Very useful.

Of course with something like this, without megabytes of memory to play with, there is a penalty to be paid. This is the long drawing time for each successive zoom. This means that you could well have fallen asleep by the time the final fly's eye zoom appears in the sequence outlined above. What happens is that each amended drawing is put onto disc in its library space, overwriting and updating the previous one, and is then pulled out again for redrawing at the new scale.

It is the library storage and access facility that really puts the cherry on the top, to make the whole package a mouthwatering treat. FILE allows the current drawing to be put into a massive library from whence it can be withdrawn at will. Imagine! This means that anything, once drawn, is always available to be reproduced straight or to be manipulated in various ways.



The beauty of the recall system, using the COPY command, is that one is shown the complete contents of the disc at one go. The screen divides into a grid of 4 or 16 rectangles, according to initial formatting, each of which displays the stored picture in miniature. Very clever and very helpful. One selects the item one wants and the screen returns to the drawing in progress. Now the 'square frame'

cursor appears and one can shrink it, expand it (in both, or in either axis independently) or, by again twisting the Z axis knob, after an appropriate button command, rotate it to any desired angle. Of course the normal joystick movement allows this modified cursor to be put anywhere, and on pressing the red button the drawing, straight or modified, appears.

The possibilities are tremendous - any item stored can now be adapted to a new purpose. Imagine, house plans could be altered, or even made into terraces by repeated COPY calls. (Regarding my earlier grumble, the manual says that a circle can be called and altered to make any perfect ellipse - but because the cursor indicators only mark the corners of the original screen, it is almost impossible to align these new ellipses with any degree of accuracy).

There are two more serious question marks over the whole package however. In the earlier Apple version the package could accept digitiser tablet input which of course opens up whole new areas of possibilities - such a facility coupled with British Micro's Graphpad seems logical, but there isn't one! Similarly the old Bitstik could output to a Calcomp or Watanabe plotter. To be useful, any CAD system must have a facility for clear, accurate hard copy. At present Bitstik can output to the Acorn Sparkjet printer which is adequate but doesn't really do justice to the full power of the Bitstik system. [We understand that Acorn are working on a driver routine for an, as yet unannounced, graph plotter - Ed.]

So all-in-all the whole package is a veritable Aladdins Cave of graphics delights marred only by the price and the occasional niggle that consultation with a battery of practicising draughtsmen would have avoided. Do I want one? Yes please, and I am now saving up hard - because can I afford one? NO!

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SHINWA CP80 '£' PATCH - D.A.Clover

Change line 70 in the small patch program published in HINTS (Vol.2, No.10, p.17) to read

```
70 LDA #&81
```

run the program, and a pound sign will be correctly printed on the Shinwa CP80. Pressing Break is not required.

PRINTER DUMP WITHOUT CORRUPTING DISPLAYS - P.S.Ganney

This is for use in immediate mode and allows you to call a screen dumping program without corrupting the display. It works by typing to the display at the graphics cursor which is first moved off the visible screen area. It's easiest to define a function key to do this.

```
*KEY0 |Y|D|M|M|M|M|ECALL&A00|M|D
```

The screen dump is assumed to reside at &A00, but this could be altered if required.

PERSONALISED HEADER ON BREAK - J.Martins, Norway.

There are of course more useful ways to use the Break vector, but the short piece of code below will personalize your title banner on Break.

```
10 osasci=&FFE3:osbyte=&FFF4:PROCassemble:CALL init:END
20 DEFPROCassemble
30 FORpass%=0 TO 3 STEP 3:P%=&C00
40 [OPT pass%:.start BCC exit:LDX#11:LDY#0
50 .print LDAmess,Y:JSRosasci:INY:DEX:BNEprint:.exit RTS
60 .init LDY#0:LDA#&F7:LDX#&4C:JSRosbyte:LDA#&F8:LDX#start MOD 256:JSRosbyte
70 LDA#&F9:LDX#start DIV 256:JSRosbyte:RTS:.mess:]
80 $P%="Your name":P%=P%+9
90 ?P%=13:P%?1=13:P%=P%+2
100 NEXT:ENDPROC
```

Replace 'Your name' with a suitable string no longer than nine characters.

TESTING OUT YOUR MICRO (Part 5)
PORTS AND INTERFACES

by Hugh Brown-Smith

This month we continue our series on testing out your micro, by looking at the various ports and interfaces, apart from the cassette interface which was covered in the previous article in this series.

The BBC micro (Model B) has a variety of ports and interfaces as standard. This single program will test all of these (apart from the cassette port), and also includes a test of the sound generator, which is another part of the machine's hardware.

The program is presented in such a way that you need type in only those sections needed for the parts you wish to test. You will need to enter all lines before 1000, and lines 9997 to 11030 in every case. The instructions for each part are headed by a REM statement for identification. Simply include the sections you require.

ANALOGUE PORT

The first part of the program tests the analogue port. It will be necessary to connect a set of linear joysticks to the analogue input to do this. Linear joysticks are the ones with potentiometers as opposed to the digital type which have switches. If you have a switched type of joystick, which switches in different resistor values, it should work but this will depend on the values of resistors used, and this will vary between manufacturers. If you do not wish to test the analogue port, wait until you are prompted, and press the space bar.

Two separate tests are performed with each joystick, one being the analogue test and the other digital (for the fire button). If you do not have a fire button, or it is not working, then the program will automatically move on to the next test after about 10 seconds. This does mean that if you have fire buttons, you will have to press the right one within 10 seconds. There is an option to repeat the joystick tests, as failure in the analogue test could be simply because you have positioned the joystick badly, or used the wrong one.

USER PORT

The second section of the program deals with the user port. It will be necessary to insert a loop-back plug here if you wish to test the user port, (see accompanying table for connection details). The results will be plotted whether the plug is connected or not, and will show red for a fail or green for a pass. No test is made of the control bits as in most applications these are not used. It should be noted that it is possible to confuse the system if a large number of bits are permanently high or low. In such a case, where a large number of fails are displayed (assuming that the plug is fitted correctly), the test can only be taken as an indication of possible failure in this area. Further test equipment would be needed to isolate the fault.

SOUND CHANNEL

The third test is for the sound channel which although not technically an interface, has been included for the sake of completeness. You should listen here for a series of four notes, the first being a noise produced on channel 0 while the following 3 produce a series of rising notes on channels 1, 2 and 3 respectively. The test will loop around until the space-bar is pressed.

RS423 PORT

Finally the RS423 port is tested. This will produce a 'PASS' or 'FAIL' message and requires no user interaction other than the insertion of a serial loop-back plug before the start of the test (see table below). This is only a simple loop-back test, and can therefore only be taken as a guide to the working of the RS423 system (it does not cover different baud rates nor does it test individual bits).

PRINTER PORT AND 1MHZ BUS

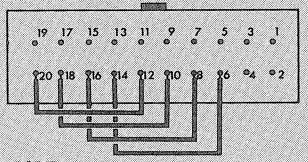
No tests have been included for the printer port or the 1MHz bus. As most printers have a self-test facility, anyone using this interface should be able to determine whether any apparent failure in the use of this port is due to the port itself, or to the device connected to it. The 1MHz bus has also not been included because of the need for additional hardware that would be expensive and of little use to most readers.

Loop-Back Plug Connections.

USER PORT

Connect pin to pin:

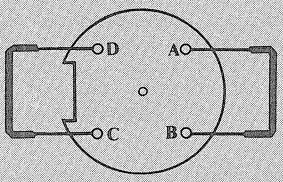
20 - 12
18 - 10
16 - 8
14 - 6



RS423 PORT

Connect pin to pin:

A - B
C - D




```
10 REM Program PORTS
20 REM Author Hugh Brown-Smith
30 REM Version B1.1
40 REM BEEBUG AUG/SEPT 1984
50 REM Program subject to copyright
60 :
100 ON ERROR GOTO11000
110 DIMBIT(8)
120 *TAPE
130 MODE7
140 Q=0
150 :
1000 REM ANALOGUE PORT
1010 JO=1
1020 VDU28,0,24,39,2
1030 CLS
1040 PRINT''CHR$141CHR$130SPC9"The Ana
logue Input"
1050 PRINTCHR$141CHR$130SPC9"The Analo
gue Input"
1060 VDU28,0,24,39,7
1070 @%=&00001
1080 PRINT'CHR$129"Connect Joysticks"
1090 PRINT'CHR$131"Move joystick "JO"
to top left & press spc"
1100 R1=0:R2=0
1110 *FX15,1
1120 REPEAT
1130 UNTILGET=32
1140 IFJO=1:R1=ADVAL(1):R2=ADVAL(2)
1150 IFJO=2:R1=ADVAL(3):R2=ADVAL(4)
1160 SOUND 1,-15,180,2
1170 PRINT'CHR$131"Move joystick "JO"
to bottom right & space"
1180 R3=0:R4=0
1190 *FX15,1
1200 REPEAT
1210 UNTIL GET=32
1220 IFJO=1:R3=ADVAL(1):R4=ADVAL(2)
1230 IFJO=2:R3=ADVAL(3):R4=ADVAL(4)
1240 SOUND 1,-15,180,2
1250 PRINT'CHR$131SPC8"Press joystick
"JO" button"
1260 TIME=0
1270 IFJO=1 REPEAT S1=ADVAL(0) AND1:UN
TILS1=1 OR TIME>1000
1280 IFJO=2 REPEAT S1=ADVAL(0) AND2:UN
TILS1=2 OR TIME>1000
1290 IF S1<1OR S1>2PRINT'CHR$129SPC9"F
AILED JOYSTICK "JO" FIRE"ELSE PRINT'CHR
$130SPC9"PASSED JOYSTICK "JO" FIRE"
1300 IF ABS(R1-R3)<15000 OR ABS(R2-R4)
<20000 PRINT'CHR$129SPC11"FAILED JOYSTI
CK "JO ELSE PRINT'CHR$130SPC11"PASSED J
OYSTICK "JO
1310  PRINT'CHR$134SPC7"Test Joystick
"JO" again (Y/N)"
1320 *FX15,1
1330 REPEAT A=GET:UNTILA=ASC"Y"OR A=AS
C"N"
1340 IF A=ASC"Y" GOTO1020 ELSE IF JO<>
2 JO=2:GOTO1020
1350 :
2000 REM USER PORT
2010 VDU28,0,24,39,2:CLS
2020 PRINT''CHR$141CHR$130SPC12"User P
ort Test"
2030 PRINTCHR$141CHR$130SPC12"User Por
t Test"
2040 VDU28,0,24,39,6
2050 PRINT'CHR$129SPC7"Insert User Loo
p-Back Plug"''
```

```
 2060 PROCget
 2070 HI%=15:LI%=240
 2080 ?&FE62=HI%
 2090 ?&FE60=0
 2100 IF ?&FE60<>0 HFA=?&FE60 ELSE HFA=0
 2110 ?&FE62=LI%
 2120 ?&FE60=0
 2130 IF ?&FE60<>0 LFA=?&FE60 ELSE LFA=0
 2140 IF( LFA AND1)=1 BIT(0)=0 ELSE BIT
(0)=1
 2150 IF(LFA AND2)=2BIT(1)=0 ELSE BIT(1
)=1
 2160 IF(LFA AND4)=4BIT(2)=0 ELSE BIT(2
)=1
 2170 IF(LFA AND8)=8BIT(3)=0 ELSE BIT(3
)=1
 2180 IF(HFA AND16)=16BIT(4)=0 ELSE BIT
(4)=1
 2190 IF(HFA AND32)=32BIT(5)=0 ELSE BIT
(5)=1
 2200 IF(HFA AND64)=64BIT(6)=0 ELSE BIT
(6)=1
 2210 IF(HFA AND128)=128BIT(7)=0 ELSE B
IT(7)=1
 2220 FORX=0TO3
 2230 ?&FE62=HI%
 2240 IF BIT(X) ?&FE60=(2^X):IF (?&FE60
 AND(2^(X+4)))=2^(X+4)PRINTCHR$130SPC12
"PASSED BIT "X" ] "X+4 ELSE PRINTCHR$12
9SPC12"FAILED BIT "X" ] "X+4
 2250 NEXT
 2260 FORX=4TO7
 2270 ?&FE62=LI%
 2280 IF BIT(X) ?&FE60=(2^X):IF (?&FE60
 AND(2^(X-4)))=2^(X-4)PRINTCHR$130SPC12
"PASSED BIT "X" [ "X-4 ELSE PRINTCHR$12
9SPC12"FAILED BIT "X" [ "X-4
 2290 NEXT
 2300 PROCget
 2310 :
 3000 REM SOUND
 3010 VDU28,0,24,39,2:CLS
 3020 PRINT''CHR$130CHR$141SPC9"The Sou
nd Channels"
 3030 PRINTCHR$130CHR$141SPC9"The Sound
 Channels"
 3040 VDU280,24,39,6:CLS
 3050 PRINT''CHR$134SPC10"Press space t
o exit"
 3060 REPEAT
 3070 SOUND0,-15,100,10
 3080 TIME=0:REPEAT UNTIL TIME>30
 3090 SOUND1,-15,120,10
 3100 TIME=0:REPEAT UNTIL TIME>30
 3110 SOUND2,-15,140,10
 3120 TIME=0:REPEAT UNTIL TIME>30
 3130 SOUND3,-15,160,10
 3140 TIME=0:REPEAT UNTIL TIME>30
 3150 UNTILINKEY(-99)
 3160 :
 4000 REM SERIAL PORT
 4010 VDU28,0,24,39,2
 4020 CLS
 4030 PRINT''CHR$130CHR$141SPC12"RS423
Port Test"
 4040 PRINTCHR$130CHR$141SPC12"RS423 Po
rt Test"
 4050 VDU28,0,24,39,6
 4060  PRINT''CHR$129SPC6"Insert RS423
Loop-Back Plug"''
 4070 PROCget
 4080 PRINT'CHR$134SPC15"TESTING"'
 4090 *FX7 1
 4100 *FX8 1
 4110 *FX3 7
 4120 PRINT"RDBSDU!LDRR@FD"
 4130 *FX3 4
 4140 *FX2 1
 4150 A$=""
 4160 FORX=0TO13
 4170 A$=A$+INKEY$(15)
 4180 NEXT
 4190 IF A$="RDBSDU!LDRR@FD" PRINT'CHR$
130SPC13"PASSED RS423"'ELSE PRINT'CHR$1
29SPC13"FAILED RS423"'
 4200 *FX2
 4210 :
 4220 REM EXIT!
 4230 PROCget
 9997 CALL!-4
 9998 END
 9999 :
10000 DEFPROCget
10010 PRINT'CHR$131SPC6"Press spacebar
to continue"
10020 *FX15,1
10030 REPEAT
10040 UNTIL GET=32
10050 ENDPROC
10051 :
11000 ON ERROR OFF
11010 MODE7:REPORT
11020 PRINT" at line "ERL
11030 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

DELETING NULL FILES

To delete a null file from disc, that is one whose presence in the catalogue only shows as spaces, type *DEL."".

INTERSTELLAR RAIDER (32k)

by Alastair Nicol

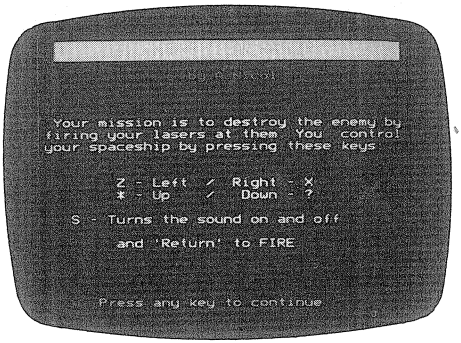
Your spaceship, the Black Falcon, is endlessly locked in battle with Starfighters of the Galactic Empire. Your hands grip the controls of the laser cannon as you wait apprehensively for one of the enemy ships to move into your sights.

Interstellar Raider is a one player space game which runs in mode 2 and uses the BBC micro's graphics very well. The game is both fast and compelling, with a full high score table. The listing is quite long, but the game certainly justifies the length.

Your view is from the cockpit of the fighter, and you are looking out into deep-space. You use the four direction keys, Z, X, * and ? to move left, right, up and down respectively. Remember that when you move left, the enemies ship will move to the right. You fire your laser cannon using the Return key, and you have 25 shots for each enemy ship. Destroy the enemy ship and your firepower is returned to the starting level of 25 shots ready for the next attack. If you use all of your 25 shots, then you die and the game is over. There is also a facility to turn the sound on and off, and this is implemented by pressing the 'S' key.

There are three enemy ships per level, and after each level the enemy moves more erratically, and becomes harder to hit. You score points by shooting three ships before your fuel reaches zero, and you score one point for every unit of fuel remaining.

All of the colours are used in this game, and most of them are used solely to build up the moving background of stars.



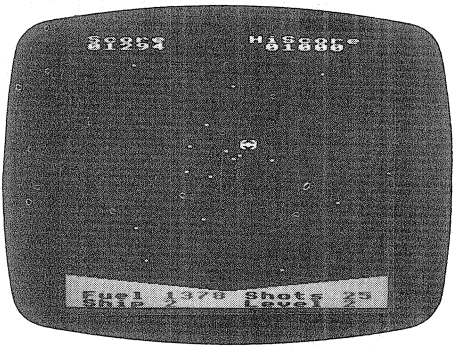
PROGRAM NOTES

As the program is only a moderate length, there is plenty of room for the full procedure names. The most important procedures, and their functions are outlined below.

PROCinstr	Gives instructions and the keys to use in the game.
PROCinit	Initialises the envelopes and high score table.
PROChiscore	Prints out the high score table.
PROCvdus	Defines the characters and sets the colours.
PROCstars	Draws the stars on the screen.
PROCsights	Draws the sight on the screen.
PROCinitl	Initialises and prints the fuel, score and other information.
PROCNman	Resets the fuel, number of shots and alters the level and ships.
PROCmove and PROCprint	Moves and prints the Imperial Starfighter.
PROCupdate	Update the fuel.
PROCgameover	Prints out 'Game Over'
PROContable	Tests to see if you are on the high score table.

```
10 REM Program STELLAR
20 REM Version B0.3
30 REM Author  Alastair Nicol
40 REM BEEBUG  Aug/Sept 1984
50 REM Program subject to copyright
60 :
100 ON ERROR GOTO 570
110 PROCinstr
120 PROCinit
```

```
  130 REPEAT
  140 MODE7
  150 VDU23;11,0;0;0;0
  160 PROChiscore
  170 MODE2
  180 VDU23;11,0;0;0;0
  190 VDU5
  200 PROCvdus
  210 PROCstars
  220 PROCsights
  230 PROCinit1
  240 Ships%=1
  250 COLOUR10
  260 PROCoutput(STR$(Dif%),17,1,1)
  270 REPEAT
  280 PROCNman
  290 I%=0
  300 REPEAT
  310 I%=I%+1
  320 IF I%=7 THEN I%=1
  330 VDU19,I%,7;0;
  340 t%=TIME+10
  350 REPEAT
  360 IF POINT(608,496)<>0 THEN SOUND1,
-15,100,1
  370 PROCmove(Dif%)
  380 IF INKEY-73 PROCprint(MX%,MY%-32)
  390 IF INKEY-105 PROCprint(MX%,MY%+32)
  400 IF INKEY-98 PROCprint(MX%+64,MY%)
  410 IF INKEY-67 PROCprint(MX%-64,MY%)
  420 IF INKEY-74 PROCshoot
  422 IF INKEY-82 SO%=SO%+1:SO%=SO% MOD
 2:PROCsound(SO%)
  430 UNTIL TIME>t%
  440 *FX21,0
  450 PROCupdate
  460 VDU19,I%,0;0;
  470 UNTIL Dead%<>0
  480 Ships%=Ships%+1
  490 UNTIL Ships%=4 OR Dead%=1
  500 IF Dead%=2 THEN PROCnewlev:GOTO240
  510 PROCgameover
  520 MODE7
  530 PROContable
  540 UNTIL FALSE
  550 END
  560 :
  570 ON ERROR OFF
  580 MODE7:IF ERR=17 END
  590 REPORT:PRINT" at line ";ERL
  600 END
  610 :
 1000 DEF PROCinit
 1010 Hi%=0
 1020 DIM Name$(8),Sc%(8)
 1030 @%=1
 1040 FORJ%=1 TO 8
 1050 Name$(J%)="BEEBUG"
 1060 Sc%(J%)=1000
 1070 NEXT
```



```
 1080 ENVELOPE1,1,-10,-5,-2,2,10,3,0,0,
0,-3,126,4
 1090 ENVELOPE2,4,0,0,0,0,0,0,1,0,-4,0,-6
,126,80
 1100 ENDPROC
 1110 :
 1120 DEF PROChiscore
 1130 Hi%=Sc%(1)
 1140 A$=CHR$(131)+CHR$(157)+CHR$(141)+
CHR$(129)
 1150 FORR=0TO1:PRINTTAB(0,R)A$ TAB(6)"
INTERSTELLAR RAIDER HISCORE":NEXT
 1160 FORJ%=1 TO 8
 1170 PRINTTAB(3,(J%*2)+3);J%".."RIGHT$
("00000"+STR$(Sc%(J%)),5),TAB(9)"... "N
ame$(J%)
 1180 NEXT
 1190 PRINTTAB(9,22)CHR$134;CHR$136"Pre
ss any key to play"
 1200 G=GET
 1210 ENDPROC
 1220 :
 1230 DEF PROCvdus
 1240 VDU23,224,192,192,0,0,0,0,0,0
 1250 VDU23,225,102,195,153,255,255,153
,195,102
 1260 VDU19,8,0;0;
 1270 VDU19,9,1;0;
 1280 VDU19,10,2;0;
 1290 VDU19,11,3;0;
 1300 VDU19,12,4;0;
 1310 VDU19,13,5;0;
 1320 VDU19,14,6;0;
 1330 VDU19,15,7;0;
 1340 VDU19,1,7;0;
 1350 FORJ%=2 TO 6
 1360 VDU19,J%,0;0;
 1370 NEXT
 1380 ENDPROC
 1390 :
 1400 DEF PROCstars
 1410 FOR J%=15 TO 360 STEP 15
```

```
1420 C%=RND(6):CS=COS(RAD(J%)):SN=SIN(
RAD(J%))
1430 FOR K%=RND(30)+50 TO 700 STEP RND
(40)+100
1440 C%=C%+1
1450 GCOL0,C% MOD 6+1
1460 PLOT4,640+CS*K%,512+SN*K%:VDU224
1470 NEXT
1480 NEXT
1490 ENDPROC
1500 :
1510 DEF PROCsights
1520 VDU19,10,0;0;
1530 GCOL0,10
1540 S=SIN(2*PI/90):C=COS(2*PI/90)
1550 X1=70:Y1=0
1560 COLOUR13
1570 MOVE678,496
1580 FOR J%=1 TO 90
1590 X=X1*C-Y1*S
1600 Y=X1*S+Y1*C
1610 DRAW608+X,496+Y
1620 X1=X:Y1=Y
1630 NEXT
1640 MOVE508,496:DRAW598,496
1650 MOVE618,496:DRAW708,496
1660 MOVE608,396:DRAW608,486
1670 MOVE608,506:DRAW608,596
1680 VDU19,10,2;0;
1690 GCOL0,11
1700 MOVE0,0:MOVE0,130:PLOT85,1280,0
1710 MOVE1280,0:MOVE1280,130:PLOT85,0,0
1720 ENDPROC
1730 :
1740 DEF PROCinit1
1750 TIME=0
1760 Sc%=0
1770 Dif%=1
1780 Dead%=0
1790 Ships%=0
1800 VDU4
1810 COLOUR13
1820 PRINTTAB(0,0)" Score     HiScore"
1830 PRINTTAB(0,1)" 00000"
1840 PROCoutput(STR$(Hi%),12,1,5)
1850 VDU4
1860 VDU28,0,31,19,30
1870 COLOUR139:COLOUR12
1880 CLS
1890 PRINT" Fuel       Shots"
1900 PRINT" Ship       Level";
1910 VDU5
1920 ENDPROC
1930 :
1940 DEF PROCNman
1950 VDU5
1960 Dead%=0
1970 Fuel%=1500
1980 Shot%=25
1990 MX%=64:MY%=920
2000 GCOL4,7
2010 MOVEMX%,MY%:PRINTCHR$225
2020 PROCoutput(STR$(Shot%),17,0,2)
2030 PROCoutput(STR$(Ships%),6,1,1)
2040 ENDPROC
2050 :
2060 DEF PROCmove(D%)
2070 PROCprint(MX%+(RND(3)-2)*12.8*D%,
MY%+(RND(3)-2)*6.4*D%)
2080 ENDPROC
2090 :
2100 DEF PROCprint(X%,Y%)
2110 IF X%<=0 OR X%>=1216 OR Y%<=162 O
R Y%>=924 ENDPROC
2120 GCOL4,7
2130 MOVEMX%,MY%:PRINTCHR$225:MOVEX%,Y
%:PRINTCHR$225
2140 MX%=X%:MY%=Y%
2150 ENDPROC
2160 :
2170 DEF PROCshoot
2180 Shot%=Shot%-1
2190 PROCoutput(STR$(Shot%),17,0,2)
2200 IF Shot%=0 THEN Dead%=1
2210 GCOL3,9
2220 MOVE160,115:DRAW608,496
2230 DRAW1120,115
2240 SOUND1,1,150,1
2250 MOVE160,115:DRAW608,496
2260 DRAW1120,115
2270 IF POINT(608,496)=0 THEN ENDPROC
2280 VDU19,1%,0;0;
2290 Dead%=2
2300 GCOL4,7
2310 MOVEMX%,MY%:PRINTCHR$225
2320 SOUND0,2,5,10
2330 FORI%=1 TO 6 STEP 2
2340 VDU19,I%,1;0;
2350 VDU19,I%+1,7;0;
2360 PROCwait(20)
2370 VDU19,I%,0;0;
2380 VDU19,I%+1,0;0;
2390 NEXT
2400 ENDPROC
2410 :
2420 DEF PROCupdate
2430 Fu%=Fuel%-TIME DIV 10
2440 IF Fu%<=0 Fu%=0:Dead%=1
2450 PROCoutput(STR$(Fu%),6,0,4)
2460 ENDPROC
2470 DEF PROCnewlev
2480 Sc%=Sc%+Fu%
2490 VDU4,26
2500 COLOUR128:COLOUR13
2510 PROCoutput(STR$(Sc%),1,1,5)
2520 VDU28,0,31,19,30
2530 COLOUR139:COLOUR12
2540 TIME=0
2550 Dif%=Dif%+1
2560 ENDPROC
```

```
2570 :
2580 DEF PROCgameover
2590 FORJ%=0 TO 1
2600 FORK%=1 TO 6
2610 VDU19,K%,7;0;
2620 VDU19,K%,0;0;
2630 NEXT
2640 NEXT
2650 SOUND0,2,5,50
2660 VDU19,3,1;0;
2670 VDU4,26
2680 COLOUR14:COLOUR0
2690 VDU19,1,1;0;
2700 PRINTTAB(5,15)"Game Over"
2710 PROCwait(750)
2720 ENDPROC
2730 :
2740 DEF PROContable
2750 *FX21,0
2760 IF Sc%<=Sc%(8) THEN ENDPROC
2770 PRINTTAB(11,9)CHR$(134)+"YOUR SCO
RE WAS IN"
2780 PRINTTAB(12,10)CHR$(134)+"THE TOP
EIGHT"
2790 PRINTTAB(8,11)CHR$(130)+CHR$(136)
+"PLEASE ENTER YOUR NAME"
2800 Name$=""
2810 PRINTTAB(9,12);
2820 REPEAT
2830 G=GET
2840 IF G=13 THEN 2900
2850 IF G=127 AND LEN(Name$)>=1 THEN N
ame$=LEFT$(Name$,LEN(Name$)-1):GOTO2880
2860 IF (G=127 AND LEN(Name$)=0) OR LE
N(Name$)=15 THEN VDU7:GOTO2890
2870 Name$=Name$+CHR$(G)
2880 VDUG
2890 *FX21,0
2900 UNTIL G=13
2910 Rank%=9
2920 REPEAT
2930 Rank%=Rank%-1
2940 UNTIL Rank%=1 OR Sc%(Rank%-1)>Sc%
2950 FOR J%=8 TO Rank%+1 STEP -1
2960 Name$(J%)=Name$(J%-1):Sc%(J%)=Sc%
(J%-1)
2970 NEXT
2980 Name$(Rank%)=Name$:Sc%(Rank%)=Sc%
2990 ENDPROC
3000 :
3010 DEF PROCoutput(A$,X%,Y%,N%)
3020 VDU4
3030 B$=STRING$(N%-1,"0")
3040 PRINTTAB(X%,Y%)RIGHT$(B$+A$,N%);
3050 VDU5
3060 ENDPROC
3070 :
3080 DEF PROCwait(T%)
3090 Time%=TIME
3100 REPEAT UNTIL TIME>Time%+T%
3110 ENDPROC
3120 :
3130 DEFPROCinstr
3140 VDU22,7
3150 SO%=0
3160 A$=CHR$(134)+CHR$(157)+CHR$(141)+
CHR$(133)
3170 FORR=0 TO1:PRINTTAB(0,R)A$ TAB(11
)"INTERSTELLAR RAIDER":NEXT
3180 PRINT'TAB(15)CHR$130"by A.Nicol"
3190 PRINT''
3200 PRINT" Your mission is to destroy
 the enemy byfiring your lasers at them
. You controlyour spaceship by pressin
g these keys:"
3210 PRINT''TAB(8)"Z - Left / Right
- X"'TAB(8)"* - Up / Down - ?"
3220 PRINT'TAB(3)"S - Turns the sound
on and off"''TAB(8)"and 'Return' to FIR
E"
3230 PRINT''''TAB(5)CHR$134"Press any
key to continue.":G=GET
3240 ENDPROC
3250 :
3260 DEFPROCsound(s2)
3270 IF s2=0 THEN *FX210,0
3280 IF s2=1 THEN *FX210,1
3290 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

DOWNLOADING LONG PROGRAMS FROM TAPE TO DISC - R.Ager

Having got a long program (up to 6E blocks) from tape on to disc (see BEEBUG Vol.2, No.7) how do you then load and run it? This solution uses screen memory, as you might expect, but carefully adjusts HIMEM and calls the O.S. to avoid damaging the screen memory contents.

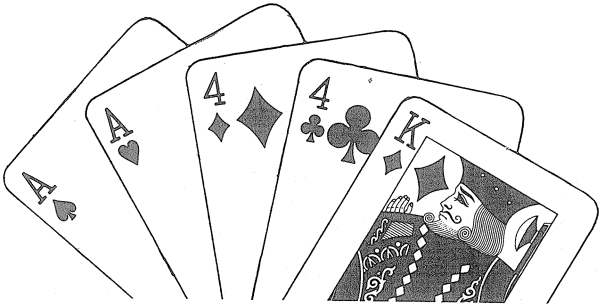
```
    1 *K.0HIMEM=&8000:V.23;8202;0;0;0;:$&C00="LO.<filename>1100":X%=0;
Y%=12:CA.&FFF7:$&C00="TA.":CA.&FFF7:F.I%=0TO&6E00
S.4:I%!&E00=I%!&1100:N.:PAGE=&E00:CLS:END|MRUN|M
    2 *FX138,0,128
    3 END
```

Don't forget to replace <filename> with the name of your file. If it's a machine code program then replace PAGE=.....RUN by CLS:CA.<execution address>

PONTOON (32K)

by Nick Day

Here is an excellent implementation of the card game Pontoon or 21. Card games like Killer Dice in BEEBUG Vol.2 No.7, can be represented very well on the BBC micro with its high resolution of graphics. If you like card games, you will find the effort of typing this program in will be well rewarded.



Having presented you with a nice baize cloth, the computer is ready to preside as banker and adjudicator over a game of Pontoon with you. All the card shuffling, dealing and betting is handled by the program offering no opportunity for you to cheat! So this is your challenge; to beat the computer.

If you do not know how to play Pontoon, the idea of the game is to win as much money as possible from the 'Banker'. Your hand (which may consist of between two and five cards) must total as near to 21 with their face values as possible without actually exceeding 21; an ace counts as 1 or 11 (the choice is yours). The rules of this version of Pontoon are that the best possible hand is a 'Pontoon' which is an ace and a picture card (11 + 10) which will give you a total of 21. The second highest hand is a 'five card trick', and is accomplished by getting five cards whose total is not more than 21, for example ace-4-6-3-4 would give you 18. The next best hands are 21 with an ace and 10, then 21 with more than 2 cards. After that the next best hand is 20, then 19 etc. The number of cards is unimportant other than in a Pontoon itself, and clearly, in five card tricks.

The program starts most amusingly, with the actual sound of the cards being shuffled before play commences. You are asked to place your bet when you are given your first card. You should determine the amount that you bet by examining your first card; an ace would normally attract a larger bet than, say, a 5. After betting on the first card (just press a key between 1 and 9 to indicate how much to bet) you are then given your second card. If your first two cards are good enough (and they must total at least 16) you may decide to 'stick', which means that you do not want any more cards, and the computer is required to play its hand. If you want another card you can 'buy' this by betting any amount from 1 up to the amount of your previous bet on this hand, or you could simply 'twist' the next card which means you can have the card, but you are not allowed to increase your bet any further.

If your cards total over 21 you 'bust' and the banker wins regardless. When you have finished your turn, and have 'stuck' with your cards, the computer will take its turn to play. The computer as banker need only achieve a total equal to or greater than your hand to win. If the computer 'sticks' on a lower total than you, or 'busts', then you win. Remember that although your cards are displayed for you to see on the screen, they are 'invisible' as far as the computer is concerned when playing its hand as banker.

PROGRAM NOTES

To save space the procedure names

have been shortened, but the most important procedures are outlined below.

PROCse Sets up the sound envelopes, variables and characters.
PROCshu Shuffles the 52 cards.
PROCsh Decides which card to display.
PROC2 to PROCk draws the corresponding card, for example:
PROCj Draws the Jack.
PROCq Draws the Queen.
PROCk Draws the King.

Take care when typing in this program, especially with the three procedures PROCj, PROCq and PROCk, otherwise you will be confronted with very strange cards indeed!

```
  10 REM PROGRAM PONTOON
  20 REM VERSION B0.2
  30 REM AUTHOR  N.DAY
  40 REM BEEBUG  AUG/SEPT 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 ON ERROR GOTO 420
 110 MODE1
 120 PROCse
 130 REPEAT
 140 CLG
 150 IFH%<1 VDU4:PRINTTAB(1,30);"you h
ave run out of money!";STRING$(10,CHR$3
2):END
 160 GCOL0,0
 170 P%=0:a=0:b=0:t=0:I%=0:w=0:p=0:F%=
0:bpon=0:bust=0
 180 IFP(1)MOD13=P(3)MOD13 temp=P(3):P
(3)=P(4):P(4)=temp
 190 FORC%=1TO10 :B(C%)=(P(C%)MOD13)+1
:S(C%)=(P(C%)DIV13)+1:IFS(C%)=5 S(C%)=4
 200 NEXT
 210 FORC%=1TO12:N(C%)=0:NEXT
 220 FORD%=1TO12
 230 *FX15,1
 240 PROCki
 250 IFD%<8 ORD%>9 P%=P%+1:L(P%)=P(1)
 260 PROCp
 270 PROCf
 280 N(D%)=N%
 290 IFD%<8:IFD%=1 ORD%=3 ORD%>4:IFN%=
1 a=TRUE
 300 PROCsh
 310 IFN(D%)>10 N(D%)=10
 320 IFD%=2 PROCbet
 330 IFD%>3 ANDD%<8 PROCscore
 340 IFD%>8 ANDNOTbust PROCbank
 350 NEXT:PROCr:PROCi:PROCrep
 360 VDU4:PRINTTAB(1,30);"You have "H%
;" pound";:IFH%>1PRINT;"s":VDU5
 370 PROCm("ANOTHER HAND?")
 380 REPEAT:A%=GET:UNTILA%=89 ORA%=78:
UNTILA%=78
 390 VDU4:PROCm(" "):VDU4:PRINTTAB(1,3
0);"You finished with "H%;" pound";:IFH
%>1 PRINT;"s"
 400 END
 410 :
 420 ON ERROR OFF:MODE 7
 430 *FX4
 440 *FX12
 450 IF ERR=17 END
 460 REPORT:PRINT" at line ";ERL
 470 END
 480 :
1000 DEF PROCse
1010 p=0:bpon=0
1020 ENVELOPE1,1,0,0,100,0,50,2,2,5,100,
0,126,90,100
1030 ENVELOPE2,2,0,0,40,5,-10,5,5,10,10,
0,-100,100,40
1040 ENVELOPE3,3,0,0,0,0,1,1,1,50,-4,0,-
4,126,80
1050 *FX11,0
1060 *FX4,1
1070 *FX202,32
1080 H%=50
1090 DIMB(10),S(10),L(10),N(12)
1100 VDU23;8202;0;0;0;
1110 VDU24,48;128;1232;960;
1120 VDU19,2,2;0;
1130 GCOL0,130
1140 CLG
1150 VDU23,224,24,60,126,126,255,255,2
19,60
1160 VDU23,225,24,60,90,255,255,90,60,
126
1170 VDU23,226,24,60,126,255,255,126,6
0,24
1180 VDU23,227,102,255,255,255,255,126
,60,24
1190 VDU23,228,76,82,82,82,82,82,82,76
1200 VDU23,229,255,255,255,255,255,255
,255,255
1210 G%=0
1220 PROCshu
1230 G%=G%+1
1240 ENDPROC
1250 :
1260 DEF PROCsh
1270 SOUND0,1,4,4
1280 GCOL0,3:MOVEX%,Y%:MOVEX%+160,Y%:P
LOT85,X%,Y%+256:PLOT85,X%+160,Y%+256:VD
U5
1290 IFD%=2 ORD%=4 PROCb:ENDPROC
1300 IFS%<3 GCOL0,1 ELSE GCOL0,0
1310 MOVEX%,Y%+250:PRINTN$:IFN%<11 MOV
EX%+128,Y%+32:PRINTN$
```

→

```
1320 IFN%=1 PROC5
1330 IFN%=2 ORN%=3 PROC2
1340 IFN%>3 ANDN%<11 PROC4
1350 IFN%=11 PROCj
1360 IFN%=12 PROCq
1370 IFN%=13 PROCk
1380 ENDPROC
1390 :
1400 DEFPROCf
1410 IFD%=8 N%=N(2):S%=S(2):GOTO1460
1420 IFD%=9 N%=N(4):S%=S(4):GOTO1460
1430 N%=(P(1)MOD13)+1
1440 S%=(P(1)DIV13)+1:IFS%=5 S%=4
1450 FORC%=1TO51:P(C%)=P(C%+1):NEXT
1460 RESTORE1510
1470 FOR PIP=1TO13
1480 READ NUMBER$
1490 IFN%=PIP N$=NUMBER$
1500 NEXT
1510 DATAA,2,3,4,5,6,7,8,9,10,J,Q,K
1520 IFN$="10" N$=CHR$(228)
1530 IFS%=1 S$=CHR$(227)
1540 IFS%=2 S$=CHR$(226)
1550 IFS%=3 S$=CHR$(225)
1560 IFS%=4 S$=CHR$(224)
1570 ENDPROC
1580 :
1590 DEF PROC2:MOVEX%+64,Y%+192:PRINTS
$:MOVEX%+64,Y%+96:PRINTS$:IFN%=3  MOVEX
%+64,Y%+146:PRINTS$
1600 ENDPROC
1610 :
1620 DEFPROC4:MOVEX%+32,Y%+224:PRINTS$
:MOVEX%+32,Y%+64:PRINTS$:MOVEX%+96,Y%+6
4:PRINTS$:MOVEX%+96,Y%+224:PRINTS$
1630 IFN$=CHR$(228) PROC9:PROC2:ENDPROC
1640 IFN$="5" OR N$="9" PROC5
1650 IFN$="6"OR N$="7" ORN$="8" PROC6
1660 ENDPROC
1670 :
1680 DEF PROC5:MOVEX%+64,Y%+146:PRINTS$
1690 IFN$="9" PROC9
1700 ENDPROC
1710 :
1720 DEFPROC9:MOVEX%+32,Y%+175:PRINTS$
:MOVEX%+32,Y%+116:PRINTS$:MOVEX%+96,Y%+
116:PRINTS$:MOVEX%+96,Y%+175:PRINTS$:EN
DPROC
1730 :
1740 DEFPROC6:MOVEX%+32,Y%+146:PRINTS$
:MOVEX%+96,Y%+146:PRINTS$:IFN$="7" ORN$
="8" PROC7
1750 ENDPROC
1760 :
1770 DEFPROC7
1780 MOVEX%+64,Y%+110:PRINTS$
1790 IFN$="8" MOVEX%+64,Y%+187:PRINTS$
1800 ENDPROC
1810 :
1820 DEFPROCj:GCOL0,0:MOVEX%+32,Y%+32:
MOVEX%+32,Y%+112:PLOT85,X%+88,Y%+32:MOV
EX%+112,Y%+32:MOVEX%+112,Y%+96:PLOT85,X
%+96,Y%+60:PLOT85,X%+96,Y%+128
1830 GCOL0,1:MOVEX%+92,Y%+128:MOVEX%+9
2,Y%+80:PLOT85,X%+64,Y%+128:MOVEX%+32,Y
%+128:MOVEX%+40,Y%+128:PLOT85,X%+96,Y%+
32:PLOT85,X%+112,Y%+32:MOVEX%+40,Y%+32:
MOVEX%+80,Y%+32
1840 PLOT85,X%+40,Y%+80:MOVEX%+120,Y%+
176:MOVEX%+120,Y%+32:PLOT85,X%+128,Y%+1
76:PLOT85,X%+128,Y%+32
1850 GCOL0,0:MOVEX%+64,Y%+128:MOVEX%+4
0,Y%+128:PLOT85,X%+64,Y%+192:PLOT85,X%+
48,Y%+192
1860 GCOL0,1:MOVEX%+112,Y%+192:PLOT85,
X%+128,Y%+224:MOVEX%+48,Y%+192:PLOT85,X
%+32,Y%+224
1870 GCOL0,0:MOVEX%+76,Y%+144:DRAWX%+9
2,Y%+144:MOVEX%+80,Y%+156:DRAWX%+88,Y%+
156:DRAWX%+88,Y%+168:MOVEX%+80,Y%+184:D
RAWX%+72,Y%+176:MOVEX%+102,Y%+184:DRAWX
%+96,Y%+176
1880 GCOL0,1:MOVEX%+104,Y%+202:MOVEX%+
112,Y%+224:PLOT85,X%+56,Y%+202:PLOT85,X
%+48,Y%+224:PROCPIC:ENDPROC
1890 :
1900 DEF PROCPIC
1910 GCOL0,0
1920 MOVEX%+32,Y%+32:DRAWX%+128,Y%+32:
DRAWX%+128,Y%+224:DRAWX%+32,Y%+224:DRAW
X%+32,Y%+32
1930 IFS%<3 GCOL0,1
1940 MOVEX%+132,Y%+64:PRINTN$:MOVEX%,Y
%+218:PRINTS$:MOVEX%+135,Y%+32:PRINTS$
1950 ENDPROC
1960 :
1970 DEFPROCk:GCOL0,1:MOVEX%+32,Y%+224
:MOVEX%+48,Y%+192:PLOT85,X%+128,Y%+224:
PLOT85,X%+112,Y%+192
1980 GCOL0,0:FORH=112TO80STEP-8:MOVEX%
+H,Y%+192:DRAWX%+H+16,Y%+128:NEXT:FORH=
88TO64STEP-8:MOVEX%+H,Y%+144:DRAWX%+H+1
6,Y%+128:NEXT
1990 MOVEX%+80,Y%+152:DRAWX%+60,Y%+152
:MOVEX%+72,Y%+160:DRAWX%+60,Y%+152:MOVE
X%+72,Y%+160:DRAWX%+68,Y%+180:DRAWX%+52
,Y%+180:MOVEX%+80,Y%+180:DRAWX%+64,Y%+1
80:MOVEX%+48,Y%+192:DRAWX%+72,Y%+128:MO
VEX%+32,Y%+112:PLOT85,X%+32,Y%+32
2000 MOVEX%+72,Y%+128:PLOT85,X%+128,Y%
+128
2010 PLOT85,X%+128,Y%+32:PLOT85,X%+32,
Y%+32:GCOL0,1:MOVEX%+32,Y%+64:MOVEX%+12
8,Y%+128:PLOT85,X%+32,Y%+32:PLOT85,X%+1
28,Y%+96:GCOL0,3
2020 MOVEX%+44,Y%+220:PRINT"X":MOVEX%+
68,Y%+220:PRINT"X":MOVEX%+92,Y%+220:PRI
NT"X"
```

→

```
 2030 MOVEX%+48,Y%+32:MOVEX%+64,Y%+32:P
LOT85,X%+48,Y%+128:PLOT85,X%+64,Y%+128:
MOVEX%+112,Y%+128:MOVEX%+112,Y%+32:PLOT
85,X%+96,Y%+128:PLOT85,X%+96,Y%+32:GCOL
0,0:FORH=132TO64STEP-16:MOVEX%+44,Y%+H:
PRINT",":MOVEX%+92,Y%+H:PRINT",":NEXT
 2040 PROCPIC:ENDPROC
 2050 :
 2060 DEFPROCq:GCOL0,0:MOVEX%+32,Y%+224
:MOVEX%+32,Y%+128:PLOT85,X%+96,Y%+224:M
OVEX%+112,Y%+224:PLOT85,X%+112,Y%+176:M
OVEX%+84,Y%+192:DRAWX%+72,Y%+168:MOVEX%
+104,Y%+168:DRAWX%+92,Y%+192:DRAWX%+96,
Y%+148:DRAWX%+84,Y%+148
 2070 GCOL0,1:MOVEX%+92,Y%+140:DRAWX%+7
6,Y%+140:MOVEX%+56,Y%+224:MOVEX%+72,Y%+
224:PLOT85,X%+32,Y%+160:PLOT85,X%+32,Y%
+128:MOVEX%+48,Y%+128:PLOT85,X%+32,Y%+3
2:PLOT85,X%+48,Y%+32
 2080 FORH=48TO64STEP8:MOVEX%+H,Y%+32:D
RAWX%+H+32,Y%+96:NEXT:MOVEX%+96,Y%+128:
MOVEX%+64,Y%+128:PLOT85,X%+128,Y%+112:P
LOT85,X%+128,Y%+32
 2090 GCOL0,0:MOVEX%+48,Y%+112:PRINT"*"
:MOVEX%+80,Y%+64:PRINT"*":MOVEX%+96,Y%+
128:MOVEX%+112,Y%+64:PLOT85,X%+128,Y%+1
12:PROCPIC:ENDPROC
 2100 :
 2110 DEFPROCb:GCOL0,1:FORC%=40TO244STE
P20:FORW%=12TO128STEP20:MOVEX%+W%,Y%+C%
:PRINT"#":NEXT,:ENDPROC
 2120 :
 2130 DEFPROCshu:VDU4:PRINTTAB(1,30);"I
'm shuffling      ":VDU5:IFG%>0 GOTO2170
 2140 DIMP(52)
 2150 FORT%=1TO52
 2160 P(T%)=T%:NEXT
 2170 FORC%=1 TO 52
 2175 IFC% MOD 5=0 SOUND0,2,4,5:T=TIME:
REPEAT:UNTIL TIME>T+(RND(25)+20)
 2180 dummy=P(C%)
 2190 newplace=RND(52)
 2200 P(C%)=P(newplace)
 2210 P(newplace)=dummy
 2220 NEXT
 2230 ENDPROC
 2240 :
 2250 DEFPROCp
 2260 IFD%=1X%=160:Y%=192
 2270 IFD%=2Y%=640:X%=160
 2280 IFD%=3X%=348:Y%=192
 2290 IFD%=4Y%=640:X%=348
 2300 IFD%=5X%=576
 2310 IFD%=6X%=784
 2320 IFD%=7X%=992
 2330 IFD%=8X%=160
 2340 IFD%=9X%=348
 2350 IFD%=10X%=576
 2360 IFD%=11X%=784
 2370 IFD%=12X%=992
 2380 IFD%>4 ANDD%<8 Y%=192
 2390 IFD%>7 Y%=640
 2400 ENDPROC
 2410 :
 2420 DEFPROCscore
 2430 Q%=N(1)+N(3)+N(5)+N(6)+N(7):IFa=T
RUE ANDQ%<12 Q%=Q%+10
 2440 IFQ%>21PROCm("BUST!"):D%=12:Q%=0:
bust=TRUE:ENDPROC
 2450 IFD%=4 ANDN(1)=1 ANDB(3)>10 p=TRUE
 2460 IFD%=4 ANDN(3)=1 ANDB(1)>10 p=TRUE
 2470 IFN(7)>0 PROCm("5 CARD TRICK"):EN
DPROC
 2480 IF w=TRUE AND Q%>15 PROCm("STICK
or TWIST?")ELSE IF w=TRUE PROCm("TWIST?
")ELSE IF Q%>15 PROCm("STICK,TWIST,BET?
") ELSE PROCm("TWIST or BET?")
 2490 REPEAT
 2500 A%=GET
 2510 UNTILA%=83 ORA%=84 ORA%=89 ORA%=66
 2520 IF w=TRUE ANDA%=66 GOTO2490
 2530 IFA%=83 AND Q%>15 D%=7:PROCm("YOU
 STICK"):ENDPROC ELSE IF A%=83 VDU 7:GO
TO2430
 2540 IFA%=84 OR A%=89 w=TRUE:PROCm("TW
IST"):ENDPROC
 2550 IFA%=66 PROCbet:ENDPROC
 2560 :
 2570 DEFPROCbank
 2580 IFN(2)=1 ORN(4)=1 b=TRUE
 2590 IFD%>7 ANDN%=1 b=TRUE
 2600 R%=N(2)+N(4)+N(10)+N(11)+N(12):IF
b=TRUE ANDR%<12 R%=R%+10
 2610 IFD%=9 ANDN(2)=1 ANDB(4)>10 Q%=0:
bpon=TRUE:D%=12:ENDPROC
 2620 IFD%=9ANDN(4)=1 ANDB(2)>10 Q%=0:b
pon=TRUE:D%=12:ENDPROC
 2630 IFp ANDNOTbpon PROCm("PAY PONTOON
"):D%=12:ENDPROC
 2640 IFR%>21PROCm("DEALER'S BUST "):D%
=12:R%=0:ENDPROC
 2650 IFR%<17 ANDD%=11 ANDN(7)>0 ENDPROC
 2660 IFR%<17 ANDRND(1)>.5 AND w=TRUE:I
FN(6)>0 ORN(5)>0 ENDPROC
 2670 IFR%=21 ANDNOTp AND N(7)=0 D%=12:
ENDPROC
 2680 IFR%>15 PROCm("BANK STICKS:"):VDU
4:PRINT;"PAY";R%+1:VDU5:D%=12:ENDPROC
 2690 ENDPROC
 2700 :
 2710 DEFPROCr:PROCw
 2720 IF Q%>21 OR Q%=0 I%=0:PROCl:ENDPR
OC
 2730 IFN(12)>0 AND NOT p I%=0:PROCl:EN
DPROC
 2740 IFN(7)>0 ANDNOTbpon  H%=H%+(I%*3)
:PROCv:ENDPROC
 2750 IFR%=21 ANDNOTp ANDN(7)=0 PROCm("
SORRY"):I%=0:PROCl:ENDPROC
 2760 IFp H%=H%+(I%*4):PROCv:ENDPROC
```

```
2770 IFQ%>R% H%=H%+(I%*2):PROCv:ENDPROC
2780 IFQ%<R%+1 PROC1:ENDPROC
2790 :
2800 DEFPROCrep
2810 FORC%=1TOP%
2820 P(53-C%)=L(C%)
2830 NEXT:ENDPROC
2840 :
2850 DEFPROCbet
2860 PROCm("YOUR BET?")
2870 B%=GET:IFB%<49ORB%>57GOTO2870
2880 IFD%>3 AND B%>F%GOTO2860
2890 IFB%-48>H% PROCm("SILLY!"):PROCw:
GOTO2850
2900 F%=B%
2910 B%=B%-48:I%=I%+B%:PROCi:PROCi2
2920 H%=H%-B%:PROCm(STRING$(15,CHR$32))
2930 ENDPROC
2940 :
2950 DEFPROCm(M$):PROCw:VDU4:PRINTTAB(
22,30);SPC(17):PRINTTAB(22,30);M$;:VDU5
:ENDPROC
2960 :
2970 DEFPROCw:C%=INKEY(200):ENDPROC
2980 :
2990 DEFPROCki
3000 VDU4:PRINTTAB(1,30);"You have "H%;
" pound";:IFH%>1PRINT;"s ":VDU5:ENDPROC
3010 :
3020 DEFPROCi:VDU5:GCOL0,2:FORC%=480TO
800 STEP32:MOVEC%,512:PRINTCHR$(229):NE
XT
3030 ENDPROC
3040 :
3050 DEFPROCi2:GCOL0,0:MOVE480,512:VDU
5:PRINT;I%;" Pound";:IFI%>1 PRINT;"s":V
DU4
3060 ENDPROC
3070 :
3080 DEFPROCv:SOUND1,3,197,4:ENDPROC
3090 :
3100 DEFPROC1:SOUND1,3,33,4:ENDPROC
```


IF YOU WRITE TO US

BACK ISSUES (Members only)
All back issues are kept in print (from April 1982). Send 90p per issue PLUS an A5 SAE to the subscriptions address. This offer is for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

SUBSCRIPTIONS
Send all applications for membership, subscription renewals, and subscription queries to the subscriptions address.

MEMBERSHIP COSTS:

U.K.
£5.40 for 6 months (5 issues)
£9.90 for 1 year (10 issues)
Eire and Europe
Membership £16 for 1 year.
Middle East £19
Americas and Africa £21
Elsewhere £23
Payments in Sterling preferred.

Subscriptions & Software Address	Subscriptions and Software Help Line
BEEBUG PO BOX 109 High Wycombe Bucks	St.Albans (0727) 60263 Manned Mon-Fri 1pm-4pm

PROGRAMS AND ARTICLES

All programs and articles used are paid for at around £25 per page, but please give us warning of anything substantial that you intend to write. In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in machine readable form using "Wordwise", "Minitext Editor" or other means. If you use cassette, please include a backup copy at 300 baud.

HINTS

There are prizes of £5 and £10 for the best hints each month.

Please send all editorial material to the editorial address below. If you require a reply it is essential to quote your membership number and enclose an SAE.

Editorial Address
BEEBUG
PO Box 50
St Albans
Herts

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams.
Production Editor: Phyllida Vanstone.
Technical Assistants: David Fell, Nigel Harris and Alan Webster.
Managing Editor: Lee Calcraft.
Thanks are due to Sheridan Williams, Adrian Calcraft, John Yale, and Tim Powys-Lybbe for assistance with this issue.

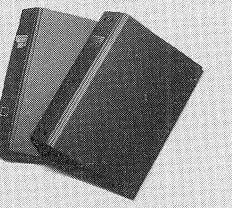
NEW BEEBUG BINDERS

We have produced a new hard-backed binder for the BEEBUG magazine. These binders are dark blue in colour with "BEEBUG" in gold lettering on the spine and allow for the whole of one volume of the magazine to be stored as a single reference book.

The new binders now have a slightly larger capacity to accommodate 10 thicker BEEBUG magazines, and are supplied with 12 wires. This enables the index and the latest copy of the supplement to be included within the binder if required. Individual issues may still be easily added and removed, allowing for the latest volume to be filed as it arrives.

The price of the new BEEBUG binder is £3.90 including VAT. Overseas members please send the same amount, this will cover the extra postage but not VAT.

BEEBUGSOFT, PO BOX 109, High Wycombe, Bucks, HP10 8HQ.



BEEBUG NEW ROM OFFER

1.2 OPERATING SYSTEM

A special arrangement has been agreed between Acorn and BEEBUG whereby BEEBUG members may obtain the 1.2 operating system in ROM at the price of £5.85 including VAT and post and packing. The ROM will be supplied with fitting instructions to enable members to install it in their machine. If the computer does not subsequently operate correctly, members may take their machine to an Acorn dealer for the upgrade to be tested, which will be done at a charge of £6.00 plus VAT. This charge will be waived if the ROM is found to have been defective. If the computer has been damaged during the installation process, the dealer will make a repair charge.

NEW ROMS FOR OLD
EXCHANGE YOUR 1.0 FOR THE 1.2

FREE

We can now exchange your old 1.0 operating system for the new 1.2, free of charge. To take advantage of this offer, please send your 1.0 (supplied on eprom with a carrier board), in good condition to the address below.

£5 FOR YOUR OLD 1.0

If you have the 1.0 operating system and have already bought a 1.2 we will exchange the 1.0 (supplied on eprom with a carrier board) for a £5 voucher. This voucher may be used against any purchase from BEEBUGSOFT.

ADDRESS FOR 1.2. OS:
ROM Offer, BEEBUG.PO Box 109, High Wycombe, Bucks, HP10 8HQ

THE BEEBUG MAGAZINE ON DISC AND CASSETTE

The programs featured each month in the BEEBUG magazine are now available to members on disc and cassette.

Each month we will produce a disc and cassette containing all of the programs included in that month's issue of BEEBUG. Both the disc and the cassette will display a full menu allowing the selection of individual programs and the disc will incorporate a special program allowing it to be read by both 40 and 80 track disc drives. Details of the programs included in this month's magazine cassette and disc are given below.

Magazine cassettes are priced at £3.00 and discs at £4.75.
SEE BELOW FOR FULL ORDERING INFORMATION.

This Month's Programs Include:

This month's cassette (Vol. 3 No. 4): includes: Pontoon Card Game, Interstellar Raider – a fast action game, a program for Temperature Measurement, a very well designed Screen Driven Music Synthesizer, an Automatic Disc Menu Utility, and the latest program for Testing Out Your Micro – Ports and Interfaces.

MAGAZINE DISC/CASSETTE SUBSCRIPTION

Subscription to the magazine cassette and disc is also available to members and offers the added advantage of regularly receiving the programs at the same time as the magazine, but under separate cover.

Subcription is offered either for a period of 6 months (5 issues) or 1 year (10 issues) and may be backdated if required. (The first magazine cassette available is Vol. 1 No. 10; the first disc available is Vol. 3 No. 1.)

MAGAZINE CASSETTE SUBSCRIPTION RATES

6 MONTHS	(5 issues)	UK £17.00 INC... Overseas £20.00 (No VAT payable)
1 YEAR	(10 issues)	UK £33.00 INC... Overseas £39.00 (No VAT payable)

MAGAZINE DISC SUBSCRIPTION RATES

6 MONTHS	(5 discs)	UK £25.50 INC... Overseas £30.00 (No VAT payable)
1 YEAR	(10 discs)	UK £50.00 INC... Overseas £56.00 (No VAT payable)

CASSETTE TO DISC SUBSCRIPTION TRANSFER

If you are currently subscribing to the BEEBUG magazine cassette and would prefer to receive the remainder of your subscription on disc, it is possible to transfer the subscription. Because of the difference between the cassette and disc prices, there will be an extra £1.70 to pay for each remaining issue of the subscription. Please calculate the amount due and enclose with your order.

ORDERING INFORMATION

Please send your order to the address below and include a sterling cheque. Postage is included in subscription rates but please add 50p for the first item and 30p for each subsequent item when ordering individual discs or cassettes in the UK. Overseas orders please send the same amount to include the extra post but not VAT.
SEND TO:

BEEBUGSOFT, PO BOX 109, HIGH WYCOMBE, BUCKS, HP10 8HQ

Printed in England by Staples Printers St Albans Limited at The Priory Press.

ISSN 0263-7561