Vol.11 No.9 March 1993

BEEBUG

FOR THE BBC MICRO & MASTER SERIES

SHAKESPEARE

n is our winter of my countenance
d fitly beauteous by this child from Goblers d

MOORLAND VERSE

LOWLAND CHOTTER SONG

We walked as carefree as some lark
Past streams and plains with thoughtful mind,

The Sideways Poet

And up before the peaks I spied
This heap - oh! knickers, this damn thing doesnt
Commit to dreary paper this thy soul? (Yae!! Ne
Or quit in scorn this base, ignoble ROM?

- SPRITER
- GRAVITY AND ORBITS
- THE FIFTEEN PUZZLE
- HEAVY WEATHER

BEEBUG Vol.11 No.9 March 1993

FEATURES

Spriter (1) 5
Fast ADFS Backup (1) 8
Gravity and Orbits (1) 12
BEEBUG Education 15
The Fifteen Puzzle 18
Heavy Weather 23
Trouble Shooting Guide (3) 25
Workshop: Tree Structures (4) 29
512 Forum 34
The Sideways Poet (1) 38
First Course: Input (2) 45
Public Domain Software 48
Mr Toad's Machine Code Corner 50
Form Designer (2) 53
Competition Results 59

REGULAR ITEMS

Editor's Jottings/News 4
Points Arising 44
RISC User 57
Hints and Tips 59
Personal Ads 60
Postbag 61
Subscriptions & Back Issues 62
Magazine Disc 63

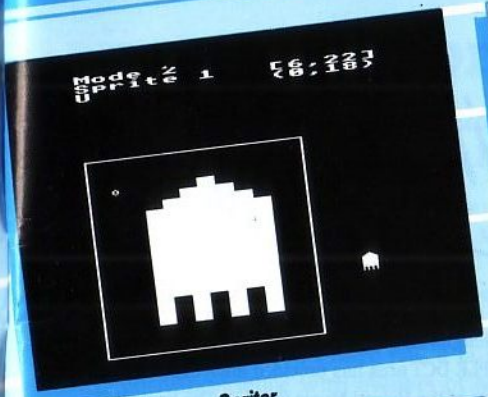
HINTS & TIPS

Control Characters in Function Key Strings
Program Protector

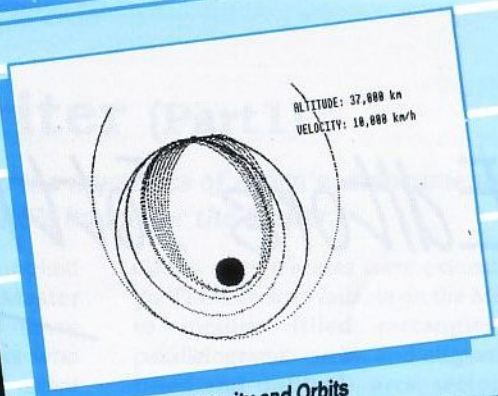
PROGRAM INFORMATION

All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints

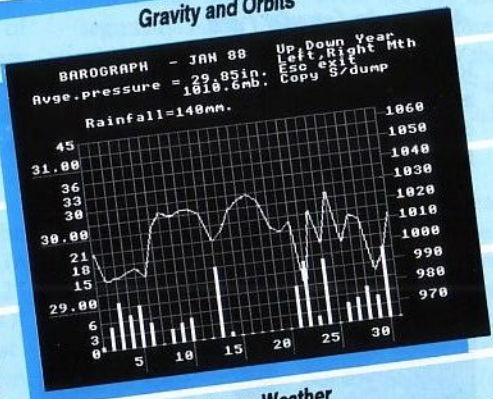


Spriter

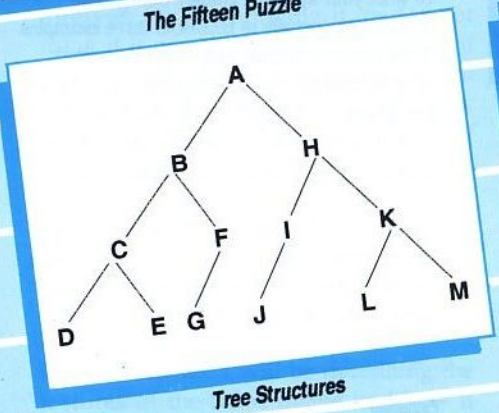


Gravity and Orbits

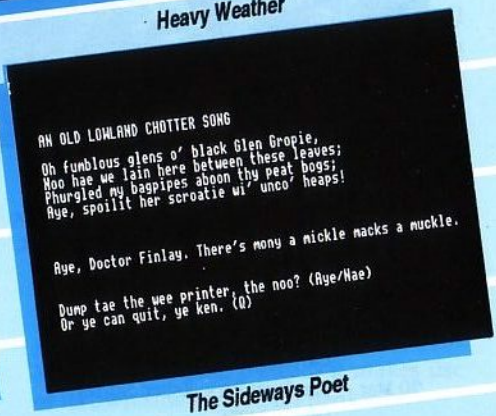
The Fifteen Puzzle



Heavy Weather



Tree Structures



The Sideways Poet

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

All Formats Computer Fairs

Regular readers will know that we frequently feature the dates of All Formats Computer Fairs on the News page. These fairs, as their name implies, are intended to cover all types of machine and can provide a useful source of both hardware and software at rock bottom prices.

With this issue of BEEBUG we are including a ticket which will give you free entry to any of the All Formats Computer Fairs listed on the ticket, so there should be one near you soon. Next month's fairs are listed below:

7 Mar	University Sports Centre, Calverley Street, Leeds.
20 Mar	Sandown Park, Esher, Surrey (J9/10 M25).
21 Mar	National Motorcycle Museum, NEC, Birmingham (J6 M42).
27 Mar	Haydock Park Racecourse (J23 M6).

On-Line Music Library for the BBC Micro

Dudley College in the West Midlands has been operating a successful educational viewdata system for nearly eight years. Within this, the AMPLE DCT area has provided enjoyment for users of the Hybrid Music System. The host DCT Database system has now been implemented on a new hard disc, and to celebrate all the music files have been formed into one large music library. These are the files which have appeared for just two or three weeks over the last six years - quite a few files.

The DCT Database can be contacted via a suitable modem set to Prestel 1200/75 protocols, and by dialling 0384 239944 or 0384 238073 (24 hours a day).

To whet your appetite, Dudley College has provided exclusively to BEEBUG three examples of the music available, and two of these are included on this month's magazine disc. These files require a standard Hybrid Music 5000 or 4000 to play. We hope to include the other and a sample catalogue next month, space permitting.

For further information, or to contribute to the library, write to David King, The DCT Database, Dudley College of Technology, The Broadway, Dudley, West Midlands DY1 4AS.

Goodbye

Mark Moxon, who has admirably fulfilled the position of Technical Editor for the last one and a half years, has deserted us for the glamour of Camden and a similar role at BBC Acorn User. My thanks in particular for all his efforts for BEEBUG and BEEBUG readers.

M.W.

Spriter (Part1)

Alan Blundell delves into the mysteries of Acorn's undocumented graphics ROM image for the Master.

The Sprite ROM version 0.6 is supplied on the Welcome disc with all Master series computers supplied since sometime in late 1986. As readers who have followed the PD Column in recent issues will know, I have had a fair bit of correspondence with owners of Masters and with Acorn about this subject. This article was prompted by suggestions from readers who wanted to know how to use the ROM image for themselves.

The ROM image is supplied on the Welcome disc in the file *Spriter* in directory *$.Library* on the Master 128 Welcome disc and in directory *$.Wimps.R* on the Master Compact Welcome disc. There is no mention of this program in the manuals for either machine. Because it is tucked away like this, you could be forgiven for not noticing it, or for not using it if you did notice it. However, it can be quite useful in programs which use moving graphics as it does away with the need for large chunks of program by simplifying the design and display of sprites. It can be used in any of the graphics modes (0,1,2,4 and 5), although sprites defined in one mode will only be usable in that mode.

ACORN'S GXR

In these articles, I will be describing the features of the ROM image and how it may be used. *Spriter*, as I will call it from now on, is essentially the same as the Graphics Extension ROM which Acorn produced for the Model B in 1985, but without the graphics extensions. If that seems nonsensical, let me explain. The GXR's main features were extensions to the PLOT codes available on the Model B, to include filled rectangles and parallelograms, circles and ellipses (both filled and unfilled), arcs, sectors and segments. Flood fills were also introduced on that ROM. Master users will know that these are standard features of the MOS on their machines, so the GXR itself would needlessly duplicate them.

The remaining parts of the GXR dealt with sprites and, whilst their introduction was welcomed at the time, they must have been a bit limited in use because they used chunks of the limited 32K of memory as sprite storage and workspace. Raising PAGE to &2000 (or even higher) to allow for a few small sprites didn't leave much memory to fit a program in if you used mode 2 to display the sprites. Spriter can be used with a shadow mode and the private ROM workspace of the Master, so would not be quite so limiting. However, it actually does not make use of these areas of RAM. The only difference between the GXR and Spriter sprite facilities is that the GXR's *SSPACE command, which was used to claim pages of RAM for sprites, is omitted. Instead, it makes use of its own sideways RAM bank.

The ROM image is only about 9K long, so there is 7K of unused space in the standard 16K SWR slot. This 7K is used for sprite workspace, which is sensible use of the space when the program is intended to be used from RAM rather than ROM, and provides much more

workspace than could practically have been made available to the older GXR.

Other features of Spriter also relate it to the GXR; its *HELP messages, of which there are two. *HELP SPRITES produces a summary of the sprite commands, together with information about the amount of workspace still free and the number of sprites defined. *HELP GRAPHICS produces a list of all of the available PLOT codes - a carryover from the original ROM.

SPRITER COMMANDS

Apart from the *HELP texts, Spriter offers 10 new star commands and support for a few VDU code sequences. The new commands are:

*SPRITE enables the use of sprites - another hangover from the GXR

*NOSPRITE disables the ROM image

*SLOAD <filename> loads one or more previously defined sprites from a file

*SSAVE <filename> saves the currently defined sprite(s) to a file

*SMERGE <filename> adds the sprites in a file to those already defined

*SDELETE n deletes the definition of sprite number n

*SNEW clears any existing sprite definitions from memory

*SRENUMBER n,m renumbers sprite number n to sprite m

Most of that is fairly self explanatory, and there should be no problems in using these commands to perform sprite filing and organizational actions.

The two main areas which need explaining in more detail are how to define the sprites in the first place, and how to make use of them once you have defined them. There are two remaining '*' commands which I have not yet dealt with, namely *SEDIT and *SCHOOSE. *SCHOOSE is for use in your own programs when displaying and animating sprites. In the rest of this article, I will explain how to use *SEDIT to create or edit sprites, and next issue will deal with displaying and animating sprites from within your own programs.

CREATING AND EDITING SPRITES

The first step you need to take is obviously to load the Spriter ROM image into sideways RAM. This can be done in the usual way, by the use of the *SRLOAD command. Insert and mount your Welcome disc and, on the Master 128, type:

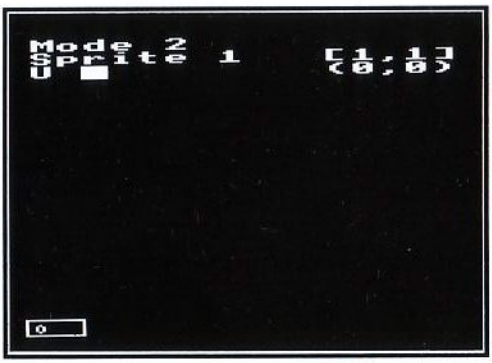
```
*SRLOAD $.LIBRARY.SPRITER 8000 4
```

on the Compact type:

```
*SRLOAD $.Wimps.R.SPRITER 8000 4 Q I
```

(You can substitute the number of any vacant SWR bank on your computer). You will need to press Ctrl-Break on the Master 128 to initialise the ROM image, after which typing *HELP should announce the presence of SPRITE ROM 0.6, together with the availability of the two *HELP parameters, Sprites and Graphics. Before you can use the ROM, you must type *SPRITE, otherwise none of the sprite commands will be enabled.

*SEDIT *n*, where *n* is a sprite number, calls up a sprite definer/editor program, similar to the character definer programs with which you will probably be familiar. There is a second form of the command, *SEDIT *n,m* which is used to copy sprite number *n* to sprite number *m*, editing sprite number *m*. This second variant is very useful when designing animated sprites, since at least two slightly different versions of the same character are normally used to give the impression of animation rather than just movement. Up to 256 sprites can be defined at one time (numbered 0 to 255), although it is unlikely that you would want to use so many or that they would all fit into 7K of workspace if you did.



*The *SEDIT screen display*

The *SEDIT screen display (mode 2 version) is shown here; as you can see, there isn't much to look at as you start. All you are given is an almost blank screen, with a small box at the bottom and a few bits of information at the top. The small box is your sprite (don't worry, we can make it bigger later). The information at the top includes the mode, the number of the sprite being defined or edited, two sets of numbers (all zero to start with), the letter 'U' and a white block. The numbers are in two pairs: The top pair show the size of the sprite being edited, as number of bytes high and wide, the lower pair are the co-ordinates of the small circle cursor within the sprite, as pixel co-ordinates. The white block is the default pixel colour for a filled pixel (empty pixels are black). Pressing a number key selects a logical colour as the current fill colour (1 = Red, 2 = Yellow, and so on). Finally, the letter 'U' stands for 'Pen Up'. Pressing function key f0 toggles the pen state from 'Up' to 'Down' and vice versa. The keys operate as follows:

f0	Pen up/down
f1	Horizontal fill
f2	Vertical fill
f3	Add row at top
Shift-f3	Remove row at top
f4	Add column at right
Shift-f4	Remove column at right
f5	Add row at cursor
Shift-f5	Remove row at cursor
f6	Add column at cursor
Shift-f6	Remove column at cursor
f7	Mirror left/right
f8	Mirror top/bottom
f9	Does nothing
Return	Set pixel
Delete	Unset pixel
1 to 9, A to F	Select logical colour

The first thing you need to do is to set a realistic size for the sprite you want to design. Referring to the table above, which shows the actions of various keys, you can see that pressing f3 makes the sprite taller and f4 makes it wider. f3 is quite straightforward, whereas f4 can be

Continued on page 14

Fast ADFS Backup (Part 1)

Roger Smith speeds up single drive ADFS backups.

Backing up an ADFS disc on a single drive is mind-numbingly boring; it takes 82 disc swaps to backup an L-format disc. The program described here, *FastBackup*, reduces this by using sideways RAM and second processor RAM (if it is present), and by copying only those sectors of the disc which are actually used. On a Master without a second processor, backing up a full L-format disc will take 16 swaps; on my own system (Model B with 8 sideways RAMs and a 6502 second processor) it takes 6 swaps. On a standard Model B with one bank of sideways RAM, FastBackup will reduce a full disc backup to 34 swaps. A partially full disc will take proportionally fewer swaps. FastBackup also ensures that the correct backup disc is used, and prevents the discs from being muddled up during the swapping process.

The Basic program *MakeBackup*, which is listed at the end of this month's article, generates the machine code program FastBackup; this requires at least one bank of sideways RAM for it to work. It always executes in the I/O processor and will overwrite all sideways RAMs, I/O processor main memory above OSHWM and second processor memory. It will only backup a disc to its corresponding backup disc, and disc identity is checked at every disc swap. The coding method is symmetrical, so FastBackup will also copy a backup disc to its source disc if the backup disc is inserted first. A backup disc is a normal ADFS disc except that its title is a bit weird!

USING THE PROGRAM

When FastBackup is run it prompts you to insert source and backup discs alternately until the backup is complete. FastBackup reads the title of the first disc inserted, and checks that the source and backup discs are correctly alternated. It will return to the "Ready to start backup..." prompt if the wrong disc is inserted, Escape is pressed or a disc error is detected. Ctrl-Break should be used to stop FastBackup.

For safety, try out FastBackup on discs which do not matter (or write protect the source disc), especially if MakeBackup has been typed in manually or modified! Note that you will also need the *InitBackup* program, to be listed in the next issue of BEEBUG, to produce backup discs with correctly encoded titles before you can use FastBackup. This InitBackup program is provided on this month's disc (and will be repeated next month).

PROGRAM NOTES

MakeBackup is a fairly straight forward Basic program for generating a machine code program. It is arranged primarily for compactness, not readability. The program it generates, FastBackup, was originally written in 6502 assembler and assembled using Alan Phillips' excellent public domain 6502 Macro Assembler (see BEEBUG Vol.10 No.8). Once it was working and extensively tested, the assembler source code was translated into BBC Basic which produces an exact duplicate of the output from the assembler.

The FastBackup program loads at address &0840 in the I/O processor, overwriting the sound buffers, the RS423/Cassette buffers, the soft key definitions and the start of the redefinable character

definitions. It uses the free space map of the source disc to find out which sectors of the disc are used, and then copies sectors to the corresponding ones on the backup disc. The program reads active sectors into memory until there is no free memory or the whole disc has been read. The data is then written to the destination disc and the process repeats until all active areas of the disc have been copied. The one exception to this is sector 6; this contains the title of the root directory which is encrypted and written to the backup disc after the first batch of sectors has been written to it.

MODIFYING THE PROGRAM

Users with second processors other than the 6502 should change the values in line 3160 of MakeBackup to correspond to the start page and number of pages of second processor memory available - make sure that the second processor O.S. is not overwritten. When FastBackup is run it detects whether the second processor is currently active before using it. The value of Tube_pgcnt should not exceed &FF.

Any other modification to the code of FastBackup should be undertaken with extreme care - a mistake can totally corrupt your discs! The early parts of FastBackup, lines 1040-1270, set up the BRK vector and find which ROM slots have sideways RAM in them; these locations are subsequently overwritten by code using the data areas defined in lines 3180-3270. The program also uses the Econet locations in page zero (&90-&9B).

The second part of the article will present two programs: InitBackup, which makes a matching backup disc for a particular source disc by encoding the title, and CheckADFS, which compares two ADFS discs and shows which sectors are different.

```
  10 REM Program MakeBackup
  20 REM Version B 1.00
  30 REM Author  Roger Smith
  40 REM BEEBUG  March 1993
  50 REM Program Subject to Copyright
  60 :
 100 PROCdefines
 110 DIM mc_area &500
 120 PROCassemble
 130 OSCLI"SAVE FastBackup "+STR$~mc_ar
ea+" "+STR$~O%+" FFFF0840 FFFF0840"
 140 END
 150 :
1000 DEF PROCassemble
1010 FOR pass=4 TO 7 STEP 3
1020 P%=start:O%=mc_area
1030 [OPT pass
1040 SEI:LDA #myBRK MOD 256:STA BRKV
1050 LDA #myBRK DIV 256:STA BRKV+1
1060 LDX #&FF:TXS:CLI:LDA #180:LDX #0
1070 LDY #&FF:JSR OSBYTE:CPX #10
1080 BCS hwm_ok:LDX #10
1090 .hwm_ok INX:INX:STX buf:LDY #&FF
1100 LDX #0:LDA #&EA:JSR OSBYTE:LDY #0
1110 TXA:BEQ noTube:LDA #17
1120 STA memtype,Y:INY
1130 .noTube LDX #0
1140 .SRAM_loop STX crt_ROM:STX ROMsel
1150 LDA &8000:EOR #&FF:STA &8000
1160 CMP &8003:BEQ nextslot:CMP &8000
1170 BNE nextslot:EOR #&FF:STA &8000
1180 TXA:STA memtype,Y:INY:LDA #0
1190 STA ROM_tt,X:.nextslot INX:CPX #16
1200 BNE SRAM_loop:TXA:STA memtype,Y
1210 INY:LDA #&FF:STA memtype,Y:JSR msg
1220 EQUB 22:EQUB 7:EQUB 13:EQUB 134
1230 EQUS"    Fast ADFS Backup - v1-00"
1240 EQUB 13:EQUB 149
1250 EQUS"    wwssssssssssssssssss{{"
1260 EQUB 13:EQUB 28:EQUB 0:EQUB 24
1270 EQUB 39:EQUB 4:EQUB eot
1280 LDX #(n_spare-1):LDA #0
1290 .zlp STA d_spare,X:DEX:BPL zlp
1300 .warmstart JSR msg:EQUB 13
1310 EQUS"Ready to start backup..."
1320 EQUB 13:EQUB eot
1330 .reset LDA #0:STA FSptr
1340 STA r_FSptr:STA w_FSptr:LDY #2
```

```
1350 STY s6_flag
1360 .L_002 STA d_sect,Y:STA r_sect,Y
1370 STA w_sect,Y:DEY:BPL L_002
1380 BMI set_Read
1390 .switch_RW BIT write_flag
1400 BPL set_Write:.set_Read LDA FSptr
1410 STA w_FSptr:LDA r_FSptr:STA FSptr
1420 LDY #2:.l_sr LDA d_sect,Y
1430 STA w_sect,Y:DEY:BPL l_sr
1440 JSR write_ttl:LDY #&FF:STY memarea
1450 INY:STY pgcnt:STY memflag
1460 STY write_flag:BEQ disc_prompt
1470 .set_Write LDA FSptr:STA r_FSptr
1480 LDA w_FSptr:STA FSptr:LDY #2
1490 .l_sw LDA d_sect,Y:STA r_sect,Y
1500 DEY:BPL l_sw:STY memarea
1510 STY write_flag:INY:STY pgcnt
1520 .disc_prompt JSR msg:EQUS"Insert"
1530 EQUB eot:BIT write_flag
1540 BMI write_prompt:JSR msg:EQUB 130
1550 EQUS"SOURCE":EQUB 135:EQUB eot
1560 JMP end_prompt
1570 .write_prompt JSR msg:EQUB 129
1580 EQUS"BACKUP":EQUB 135:EQUB eot
1590 .end_prompt JSR msg
1600 EQUS"disc and hit a key":EQUB eot
1610 JSR read_ch:JSR OSNEWL
1620 BIT write_flag:BMI startXfer
1630 LDA d_sect:ORA d_sect+1
1640 ORA d_sect+2:BNE startXfer
1650 LDA #read_fn:STA d_func:LDA #0
1660 STA s6_flag:LDX #2:LDY #&0E
1670 JSR getblks:LDX FS_end:LDY #0
1680 .dummy_FS LDA disc_size,Y
1690 STA FS_start,X:LDA #0:STA FS_len,X
1700 INX:INY:CPY #3:BNE dummy_FS
1710 STX FS_end:JSR getref_ttl:LDY #0
1720 .tloop LDA (ptr0),Y:STA src_ttl,Y
1730 EOR #15:STA (ptr0),Y
1740 STA dest_ttl,Y:INY:CPY #ttl_len
1750 BNE tloop
1760 .startXfer JSR get_title:LDX #0
1770 BIT write_flag:BPL noadj
1780 LDX #(dest_ttl-src_ttl)
1790 .noadj LDY #0
1800 .tcheck LDA (ptr0),Y:CMP src_ttl,X
1810 BEQ char_ok:BRK:EQUB 0
1820 EQUS"Wrong disc":EQUB 0
1830 .char_ok INX:INY:CPY #ttl_len
1840 BNE tcheck:LDY #2:BIT write_flag
1850 BMI load_w:.load_r LDA r_sect,Y
1860 STA d_sect,Y:DEY:BPL load_r
1870 LDA #read_fn:BNE set_fn
1880 .load_w LDA w_sect,Y:STA d_sect,Y
1890 DEY:BPL load_w:LDA #write_fn
1900 .set_fn STA d_func
1910 .Xfer_loop:LDX FSptr:SEC
1920 LDA FS_start,X:SBC d_sect+2
1930 STA d_count:LDA FS_start+1,X
1940 SBC d_sect+1:STA temp_ws
1950 LDA FS_start+2,X:SBC d_sect
1960 BCC next_FS:ORA temp_ws:BEQ lt256
1970 LDA #255:STA d_count
1980 .lt256 LDA d_count:BNE this_FS
1990 .next_FS CLC:LDY #2
2000 .L_001 LDA FS_start,X:ADC FS_len,X
2010 STA d_sect,Y:INX:DEY:BPL L_001
2020 STX FSptr:CPX FS_end:BCC Xfer_loop
2030 JMP end_Xfer:.this_FS:LDA pgcnt
2040 BNE this_mem:LDX memarea
2050 BMI no_saveRAM:BIT write_flag
2060 BMI no_saveRAM:LDA memtype,X
2070 CMP #16:BCS no_saveRAM
2080 JSR swap_SRAM:LDX memarea
2090 .no_saveRAM INX:STX memarea
2100 LDA memtype,X:CMP #&FF:BNE set_mem
2110 JMP switch_RW:.set_mem LDY #0
2120 STY d_addr:CMP #17:BEQ Tube_mem
2130 DEY:.Tube_mem STY d_addr+2
2140 STY d_addr+3:CMP #16:BCC setROM
2150 BEQ setIOmem:LDX #Tube_start
2160 LDA #Tube_pgcnt:BNE finset
2170 .setROM BIT write_flag
2180 BPL reading_1:BIT memflag
2190 BPL nomem:STA memflag
2200 .nomem JSR swap_SRAM
2210 .reading_1 LDX buf:LDA #&40
2220 BNE finset
2230 .setIOmem BIT write_flag
2240 BPL reading_2:LDA memflag
2250 JSR swap_SRAM:BEQ IO_2
2260 .reading_2 DEC memflag
2270 .IO_2 SEC:LDA #&7C:SBC buf:LDX buf
2280 .finset STX d_addr+1:STA pgcnt
2290 .this_mem CMP d_count:BCS c_ok
2300 STA d_count:.c_ok JSR disc_op:CLC
2310 LDA d_count:ADC d_addr+1
2320 STA d_addr+1:BCC NC_d_addr
```

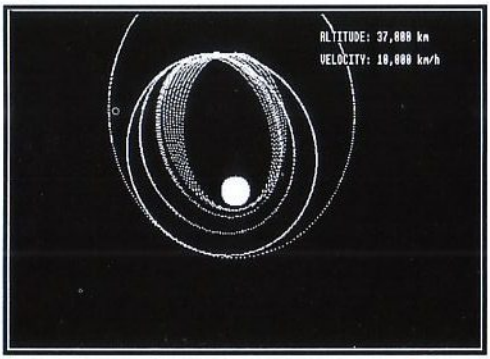
```
2330 INC d_addr+2:BNE NC_d_addr
2340 INC d_addr+3:.NC_d_addr SEC
2350 LDA pgcnt:SBC d_count:STA pgcnt
2360 CLC:LDA d_count:ADC d_sect+2
2370 STA d_sect+2:BCC NC_d_sect
2380 INC d_sect+1:BNE NC_d_sect
2390 INC d_sect:.NC_d_sect
2400 JMP Xfer_loop
2410 .end_Xfer BIT write_flag
2420 BMI complete:LDX memarea
2430 LDA memtype,X:CMP #16
2440 BCS not_SRAM2:JSR swap_SRAM
2450 .not_SRAM2 JMP switch_RW
2460 .complete JSR write_ttl:BRK:EQUB 0
2470 EQUS"Backup":EQUB 131
2480 EQUS"complete":EQUB 0
2490 .write_ttl LDA s6_flag:BNE gt_exit
2500 LDA #write_fn:STA s6_flag
2510 BNE ref_y
2520 .getref_ttl LDA #read_fn
2530 .ref_y LDY buf:DEY:BNE dec_y
2540 .get_title LDA #read_fn:LDY buf
2550 .dec_y DEY:STA d_func:LDA #6
2560 LDX #1:STY ptr0+1:JSR getblks
2570 LDX #&D9:STX ptr0:.gt_exit RTS
2580 .getblks STA d_sect+2:STX d_count
2590 STY d_addr+1:LDY #0:STY d_addr
2600 STY d_sect:STY d_sect+1:DEY
2610 STY d_addr+2:STY d_addr+3
2620 .disc_op LDA #0:STA disc_pb
2630 JSR ck_esc:LDX #disc_pb MOD 256
2640 LDY #disc_pb DIV 256:LDA #&72
2650 JSR OSWORD:LDA disc_pb
2660 BNE disc_flt:RTS
2670 .disc_flt BRK:EQUB 0
2680 EQUS"Disc error":EQUB 0
2690 .swap_SRAM STA crt_ROM:STA ROMsel
2700 LDA #&80:STA ptr0+1:LDA buf
2710 STA ptr1+1:LDA #0:STA ptr0
2720 STA ptr1:TAY
2730 .swap_loop LDA (ptr0),Y:TAX
2740 LDA (ptr1),Y:STA (ptr0),Y:TXA
2750 STA (ptr1),Y:INY:BNE swap_loop
2760 INC ptr0+1:INC ptr1+1:LDA ptr0+1
2770 CMP #&C0:BNE swap_loop:RTS
2780 .msg PLA:STA ptr0:PLA:STA ptr0+1
2790 LDY #0:.msg1 INC ptr0:BNE msg2
2800 INC ptr0+1:.msg2 LDA (ptr0),Y
2810 CMP #eot:BEQ msg3:JSR OSASCI
```

```
2820 JMP msg1:.msg3 LDA ptr0+1:PHA
2830 LDA ptr0:PHA:RTS
2840 .myBRK LDX #&FF:TXS:JSR OSNEWL
2850 LDY #1:.brk_loop LDA (os_brk0),Y
2860 BEQ brk_end:JSR OSWRCH:INY
2870 BNE brk_loop:.brk_end JSR OSNEWL
2880 JMP warmstart
2890 .read_ch LDX #1:LDA #15:JSR OSBYTE
2900 JSR OSRDCH:BCS do_escape:RTS
2910 .ck_esc PHA:LDA esc_flag
2920 BMI do_escape:PLA:RTS
2930 .do_escape LDA #&7E:JSR OSBYTE
2940 LDA #0:STA esc_flag:BRK:EQUB 17
2950 EQUS"Escape":EQUB 0
2960 ]
2970 NEXT
2980 ENDPROC
2990 :
3000 DEF PROCdefines
3010 P%=&90
3020 ptr0=FNds(2):ptr1=FNds(2)
3030 pgcnt=FNds(1):memarea=FNds(1)
3040 FSptr=FNds(1):write_flag=FNds(1)
3050 buf=FNds(1):memflag=FNds(1)
3060 temp_ws=FNds(1):s6_flag=FNds(1)
3070 FS_start=&0E00:disc_size=&0EFC
3080 FS_len=&0F00:FS_end=&0FFE
3090 os_brk0=&FD:eot=&FF:BRKV=&202
3100 crt_ROM=&F4:esc_flag=&FF
3110 ROM_tt=&2A1:ROMsel=&FE30
3120 OSRDCH=&FFE0:OSASCI=&FFE3
3130 OSNEWL=&FFE7:OSWRCH=&FFEE
3140 OSWORD=&FFF1:OSBYTE=&FFF4
3150 read_fn=&08:write_fn=&0A
3160 Tube_start=&04:Tube_pgcnt=&F4
3170 start=&840:P%=start
3180 max_mems=19:ttl_len=19
3190 memtype=FNds(max_mems)
3200 src_ttl=FNds(ttl_len)
3210 dest_ttl=FNds(ttl_len)
3220 r_sect=FNds(3):r_FSptr=FNds(1)
3230 w_sect=FNds(3):w_FSptr=FNds(1)
3240 disc_pb=FNds(1):d_addr=FNds(4)
3250 d_func=FNds(1):d_sect=FNds(3)
3260 d_count=FNds(1)
3270 n_spare=5:d_spare=FNds(n_spare)
3280 ENDPROC
3290 :
3300 DEF FNds(N%):Q%=P%:P%=P%+N%:=Q%
```

Gravity and Orbits (Part 1)

Cliff Blake teaches captain Kirk a thing or two.

Many listings to demonstrate the effect of electric or gravitation fields have been published, but usually with little detail for their use. In this series of articles the accent will be on basic principles and experiment.



Elliptical orbits progressing round the Earth

It was Johannes Kepler who deduced that Mars followed an elliptical path with the Sun at one of its focal points. This could be illustrated by simply drawing an ellipse, but by the same method one could draw an orbit of any shape, even a square. In these listings the gravitational pull and its effect are calculated at each step, so that the orbit shape is the result of a genuine simulation.

Due to the restricted size of the screen, some relaxation of reality has to be accepted. Bodies are oversize in relation to their spacing, and mass ratios are reduced to ensure that each body shown has a gravity field of an influential size.

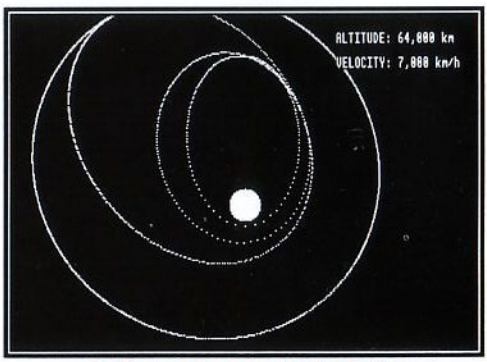
THE PROGRAM

The orbiting body is assumed to be a spaceship going round the Earth. Initial position and velocity are set to produce an elliptical orbit, and the trail of the spaceship is marked to show its track. The velocity and altitude from the centre of the planet are also displayed.

If the program is just left to run for a few orbits, the ellipse will be seen to also slowly swing round the planet. The odds are against an orbiting body following precisely the same path as it did on its previous orbit, so the ellipse swings round. It may move either in the same direction as the body is travelling, or in the opposite rotation.

CHANGING ORBIT

The position where the spaceship is at greatest altitude is the *apogee* (Gk. from Earth), and the point of lowest altitude is the *perigee* (Gk. near Earth).



Achieving a near circular orbit

At apogee the spaceship has minimum velocity. By firing the propulsion unit in this region, the vehicle is accelerated and does not fall in so close to the planet at perigee. Press key 'R' to repeat the program. After an orbit or two, press the

'>' key to accelerate when the vehicle is in the region at the top of the screen. It will be seen that the ship does not approach so close to the planet on the lower side, and repeated adjustment can produce a circular orbit with a larger radius.

At perigee the spaceship reaches maximum velocity. By firing the retro-braking unit in this region, the vehicle is slowed and does not climb so high at apogee. Press key 'R' to repeat the program again. After an orbit or two, press the '<' key to slow the vehicle while it is in the region at the bottom of the screen. Notice that the ship does not climb so high from the planet on the apogee side.

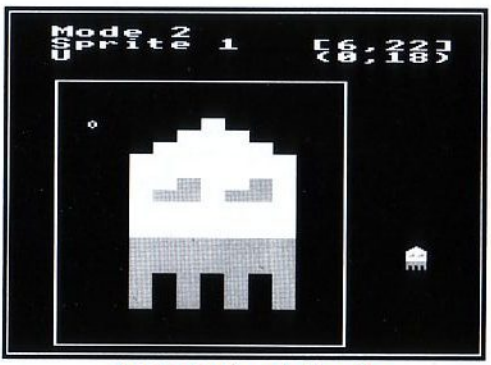
Now practice using both controls to obtain a circular orbit of intermediate radius. If you can do this in a few orbits, then try slowing on the left side, while accelerating on the right side to obtain a more horizontal ellipse. Keep up the training, and next time we'll try a trip to the moon.

```
 10 REM Program ORBIT
 20 REM Version B2.0
 30 REM Author  Cliff Blake
 40 REM BEEBUG  March 1993
 50 REM Program subject to copyright
 60 :
100 MODE7:*FX11
110 PROCinfo:g%=GET
120 MODE0:VDU5
130 REPEAT
140 quit%=FALSE
150 CLS:PROCplanet:PROCspaceship
160 REPEAT
170 rerun%=FALSE
180 PROCmove:PROCaltitude
190 PROCgravity:PROCthrust
200 PROCflags
210 UNTIL rerun%
220 UNTIL quit%
230 VDU4:*FX12
240 CLS:*FX21
250 END
260 :
1000 DEF PROCmove
1010 MOVE Xs,Ys:PRINT CHR$64
1020 ENDPROC
1030 :
1040 DEF PROCaltitude
1050 Xd=640-Xs:Yd=512-Ys
1060 Rds=Xd*Xd+Yd*Yd:Rd=SQR(Rds)
1070 alt%=INT(.2*Rd)
1080 VDU4:PRINT TAB(55,2)"ALTITUDE: ";a
lt%;",000 km";SPC3:VDU5
1090 ENDPROC
1100 :
1110 DEF PROCgravity
1120 Xg=20000*Xd/Rd/Rds:Yg=20000*Yd/Rd/
Rds
1130 Xv=Xv+Xg:Yv=Yv+Yg
1140 Rvs=Xv*Xv+Yv*Yv:Rv=SQR(Rvs)
1150 vel%=INT(.85*Rv)
1160 VDU4:PRINT TAB(55,4)"VELOCITY: ";v
el%;",000 km/h";SPC3:VDU5
1170 ENDPROC
1180 :
1190 DEF PROCthrust
1200 Xt=0.05*Xv/Rv:Yt=0.05*Yv/Rv
1210 IF INKEY(-103) THEN Xv=Xv-Xt:Yv=Yv
-Yt:SOUND0,-7,6,4
1220 IF INKEY(-104) THEN Xv=Xv+Xt:Yv=Yv
+Yt:SOUND0,-7,6,4
1230 Xs=Xs+Xv:Ys=Ys+Yv
1240 ENDPROC
1250 :
1260 DEF PROCflags
1270 IF INKEY(-52) THEN rerun%=TRUE
1280 IF INKEY(-17) THEN rerun%=TRUE:qui
t%=TRUE
1290 ENDPROC
1300 :
1310 DEF PROCplanet
1320 Ca=COS(PI/40):Sa=SIN(PI/40)
```

```
 1330 CA=1:SA=0:MOVE 640+40,512
 1340 FOR A=1 TO 80
 1350 Cp=CA:Sp=SA
 1360 CA=Cp*Ca-Sp*Sa:SA=Sp*Ca+Cp*Sa
 1370 x=40*CA+640:y=40*SA+512
 1380 MOVE 640,512:PLOT 85,x,y
 1390 NEXT A
 1400 ENDPROC
 1410 :
 1420 DEF PROCspaceship
 1430 VDU23,64,192,0,0,0,0,0,0,0
 1440 Xs=640:Ys=900
 1450 Xv=3.5:Yv=0
 1460 ENDPROC
 1470 :
 1480 DEF PROCinfo
 1490 y$=CHR$131:c$=CHR$134:w$=CHR$135
 1500 PRINT TAB(10,2)y$+"SATELLITE ORBIT
"
 1510 PRINT'c$+"A spaceship is in ellipt
ical orbit"
 1520 PRINT c$+"around a planet. Try usi
ng its"
 1530 PRINT c$+"propulsion and retro bra
king units"
 1540 PRINT c$+"to make the orbit circul
ar."
 1550 PRINT'w$+"Gently holding down the
< key will"
 1560 PRINT w$+"fire the retro unit to s
low the ship."
 1570 PRINT'w$+"Gently holding down the
> key will"
 1580 PRINT w$+"fire the drive to accele
rate the ship."'
 1590 PRINT c$+"The smaller the variatio
ns in"
 1600 PRINT c$+"altitude and velocity, t
he more"
 1610 PRINT c$+"circular the orbit."
 1620 PRINT'w$+"Press any key to start."
 1630 PRINT'w$+"Press R to clear the scr
een & Repeat."
 1640 PRINT'w$+"Press Q to Quit."
 1650 ENDPROC
```

Spriter (Continued from page 7)

confusing: The number of pixel columns added is dependent on the screen mode.



A completed mode 2 sprite

In fact, it is quite straightforward. A sprite is a whole number of bytes high and wide. Whichever graphics mode is chosen, a row of pixels is one byte high, whereas one byte horizontally can count as 2, 4 or 8 pixels. So, f4 makes the sprite one byte, rather than 1 pixel, wider. f5 and f6 work in a similar way. The only area where caution is appropriate is with the use of Shift-f4 and Shift-f6, where you could lose some defined part of your sprite which you did not intend to lose if you calculated the number of pixel columns you were about to delete wrongly.

Once you have finished your session designing a given sprite press Escape to leave the editor. You can now use the *SSAVE <filename> command to save your sprite.

That's about it for this article; next time, I will cover sprite display and animation, and the one 'Star' command which has been omitted this month, *SGET. With the information provided so far, you should be able to have a few sprites defined in readiness for the next issue.

BEEBUG Education

By Mark Sealey

This month's Beebug Education is slightly different from usual. Instead of examining a single issue or new products relevant to BBC computers in education, we take a look - after six years of this column - at a possible essential core collection of software which any school or home might aim to have.

No distinction is made between primary and secondary: there is some overlap so target ages have been indicated (by Key Stages). Whatever your interest in education it's worth reading right through.

In other words, this is a sort of educational software 'Top Ten' that could also serve as a subjective checklist for anyone - educationalists and parents - who are unused to picking their way through the thousands of educational software titles now available.

In each case there is a short description of the item chosen and reasons for its choice. 'Utilities' (such as screendumps and admin packages) are excluded. To save space only the publisher's name and a rough indication of current VAT-inclusive price are given in the text (there is usually more than one version). Addresses follow at the end. Where an "A" is shown, versions (which are usually radically enhanced) are available for the Acorn's 32-bit machines: this will further extend the products' usefulness if and when you upgrade or pupils transfer.

Granny's Garden (A)
4Mation, £15 (KS 1 and 2)

One of the classic adventures but one with a purpose. Pupils have to traverse a series of locations to find missing children. This pack - and indeed others by 4Mation such as *Dragon's World*, *Zoopack* and *Worlds Without Words* etc. - leads the users through mathematically and linguistically useful puzzles and activities to achieve an end result. Hence motivation as well as enjoyment are high. The language used to drive the software is well thought out and the graphics are good for their time. Fruitful group work is also sponsored directly and naturally.

Fun School (A)
Europress, £17 (KS 1 to 3)

If you want mathematical, science and language drill-type programs that also represent good value for money (upwards of half a dozen in a pack, in five age ranges), then Fun School's four issues for children under 5 to those over 11 are worth considering. The range of skills practiced in a less than open-ended way is huge, the coding and visual appeal of a more than satisfactory quality, and child-appeal higher than might at first be expected from programs that are often accompanied by the sort of inane 'soundtrack' that has given this type of software a bad name. For those adults who would be happier to start with software of the 'plug-in-and-go' genre, Fun School represents a good starting point.

Advanced Folio (A)
ESM, £50-55 (KS 1 to 4)

This is the enhanced version of the classic *Folio*, more than a word processor yet not as intimidating as some desktop publishers can be. This is one of the best of its type, not just for text processing but also for poster, notice and display production. The first choice for many people for class and school newspapers, it also represents an excellent introduction to this corner of Information Technology in its own right.

One of the additions to the advanced version of Folio is a set of new typefaces (fonts). Indeed the software is also available with Bengali, French, German, Gujarati, Hindi and Punjabi scripts, although each has to be bought separately and is a product in its own right.

Muddles
SPA, £15 (KS 1 to 4)

One of the very best specific language packages which has stood the test of time. There is text of a complexity and length to suit everyone. *Muddles* is suitable for everyone from emergent readers to adult literacy courses. This text is muddled in a wide range of ways and under full and easy teacher control. The user has to unscramble what they can see: maybe only the top half is present, words or phrases are displayed backwards, there are missing letters or clusters. Thus, an unusually wide range of reading skills (phonic, context, cueing decoding etc.) is practiced. An excellent example of the simplest of ideas being executed stylishly and imaginatively.

Albert's House
RESOURCE, £30 (KS 1 and 2)

Another self-contained program, though one which could be adapted to topics on animals, houses, ourselves etc. at the lower primary range. In this adventure the child is taken on an exploration of a house in several parts, each roughly identified with a different linguistic and/or mathematical skill. Although the graphics are simple, program control is excellent, which is important for the younger users (possibly unfamiliar with the micro) who are likely to use the program. Like most RESOURCE software, this one is particularly strong on generating productive pair and group discussion, and on the range of basic linguistic skills and spatial mathematical ones that are addressed.

LOGO (A)
Longman Logotron, £61 (KS 1 to 4)

Perhaps the single most indispensable piece of educational software (really a ROM but with significant extensions: 3D graphics and control of physical devices attached to the computer, £20). There is a body of thought which claims Logo teaches algebra and geometry, which justifies its place in the National Curriculum. Others believe its most valuable function is as a way for a group to explore microworlds.... computer generated environments which are mathematically and/or scientifically consistent. Hence - with more careful and sensitive guidance on the teacher's part than is often realised or admitted - a rich array of concrete problem-solving and investigative skills can be refined and developed at the pupils' own pace.

Numerator (A)
Longman Logotron, £50 (KS 1 to 4)

On the other hand, *Numerator does* give scope for skills in algebra. It is a microworld in its own right but one that both generates and helps pupils to pose and answer questions relating to number and number patterns. The microworld is that of plumbing, where tanks and pipes represent number stores and the connections between them. The tasks that can be set and completed are infinite. Again the collaborative aspects of mathematical problem-solving are particularly well catered for. Also heavily in Numerator's favour are its ease of use and its flexibility - for something so complex.

Giantkiller (A)
Topologika, £35 (KS 2 and 3)

There are several similar mathematical adventures in scope and quality to *Giantkiller.* This one distinguishes itself by the excellent supporting material including clear, usable, documentation and a support disc (£25) to extend the

original adventure in graphic form. This is based very loosely on Jack and the Beanstalk. Also strengthening Giantkiller's claims are an inventive scenario, and the great variety of mathematical content using calculators, and (particularly) problems involving space, shape and topology (e.g. maps).

Timelines (A)
Soft Teach, £70 (KS 2 to 4)

Timelines has a much wider application than might be imagined. In the first place this historical database allows events and sequences of events to be placed, viewed and investigated in a graphical way. You scroll sideways through time.Of course the chronology can be that of an 'A' Level history topic; but just as well that of the first eleven years of a child's life or the life of a relative - or even a teacher or the lifespan of the BBC Micro. The techniques of sideways scrolling and screen control are the same whatever the historical theme. Existing databases are beginning to appear for the National Curriculum History Core Study units - although these are a little expensive (£23.50).

Fletcher's Castle
Fernleaf, £25 (KS 2 and 3)

More than a historically period-specific package, Fletcher's Castle is also a simulation where a castle has to be built on the computer through logical reasoning and group discussion again. There is a time limit and in many ways the scenario is surprisingly realistic. Again, for this type of software, many examples exist: *Mary Rose* (Cambridgeshire Software house), *Titanic* (ESM) and Fernleaf's own *Railway Drama* etc. Fletcher's Castle, though, is manageable in terms of ease of completion. It doesn't have too many phases nor is it too complex for overall sight of the main objective to be lost.

OVER TO YOU

This has been a purely personal choice, though not a haphazard one by any means. You may think differently. Which titles would you have chosen? League tables are much in fashion in education at present. It would seem an interesting exercise to compile a more worthwhile table - of the educational software that is most favoured and most used by BEEBUG readers.

Bear in mind that your preferred titles must be currently available on 40 or 80 track disc for the 8-bit BBC machines. Send your lists to me care of BEEBUG and I'll report back as soon as we have enough information to make any analysis meaningful.

PUBLISHERS

Cambridgeshire Software House, 7 Free Church Passage, St Ives PE17 4AY, tel. 0480 67945
4Mation, 11 Castle Pk Rd., Whiddon Valley, Barnstaple EX32 8PA, tel. 0271 25353
Europress, Ellesmere Port L65 3EB, tel. 051-357 2961
ESM, Abbeygate Ho., East Rd., Cambridge CB1 1BD, tel. 0223 65445
Resource, Exeter Road, Doncaster DN2 4PY, 0302 340331
Longman Logotron, 124 Cambridge Science Park, Milton Rd., Cambridge CB4 4ZS, tel. 0223 425558
Topologika, PO Box 39, Stilton, Peterborough PE7 3RL, tel. 0733 244682
Soft Teach, Longbridge Deverill, Warminster BA12 7EA, tel. 0985 40329
Software Production Associates (SPA), PO Box 59 Tewkesbury GL20 6AB, tel. 0684 81700
Fernleaf, 31 Old Road West, Gravesend DA11 0LH, tel. 0474 359037

Alternatively, all the titles mentioned in Beebug Education this month can be obtained from Rickitt Educational Media, Ilto, Ilminster TA19 9HS, tel. 0460 57152.

The Fifteen Puzzle

Peter Brown takes an in-depth look at the sliding block puzzle.

The Fifteen Puzzle, consisting of fifteen square pieces in a 4 by 4 square tray has to be the most popular of all puzzles; spawning thousands of variations, including the famous Rubik's Cube. The earliest known example, 'The Puzzle of Fifteen' was manufactured by the Embossing Company of New York in October 1865.

MATHEMATICS AND PUZZLES

Apart from trial and error, mathematicians have not yet worked out how to tell whether one state in a sliding block puzzle can be reached from another, nor the most efficient way of doing this. This poses an interesting problem for mathematicians and programmers alike. However, it is known that precisely half of the start positions of a sliding block puzzle can be solved; the other start positions are impossible to solve merely by sliding.

It is a relatively simple matter to find the number of possible start positions, or permutations. This is done by multiplying successive numbers up to and including the total number of pieces. For the Fifteen Puzzle this produces the phenomenal number of 1,307,674,368,000, to ten significant figures!

Now it is a question of determining which can be solved; there is a fairly simple method for doing this. It has been mathematically proved that if an even number of swaps is needed to get from the arrangement under test to the finish arrangement then it will be possible to solve by sliding also. If an odd number of swaps is necessary then it will not be possible to solve merely by sliding. Mathematicians call this an 'even-odd parity check'. So then, here is the method: arrange the pieces in the starting position which you are testing and begin swapping pairs of numbers over (by lifting them) in order to reach the finish position. Depending upon the number of swaps needed to reach the finish state you can tell whether or not that particular arrangement is solvable.

The original puzzles had removable pieces, unlike the modern versions, so it would probably have been pot-luck as to whether the game you were trying to solve could actually be completed, though this was presumably meant to be half the fun. The modern puzzles have interlocking pieces, so, because they are jumbled simply by sliding only, they will always be possible to solve.

One man to make use of this mathematical theory, taking advantage in the fact that it was not widely realised, was Sam Loyd, 'The Prince of Puzzle-Makers'. Seven years after the original 'Puzzle of Fifteen' he devised a new version, with an interesting twist. It consisted of tiles one to thirteen in the correct order and tiles fourteen and fifteen reversed. The task was, simply, to arrange the tiles in the correct order. 'The 14-15 Puzzle', in Loyd's own words, 'drove the world mad'. So confident was Loyd that no-one could solve his puzzle that he offered a $1000 reward to anyone who could find the solution. Loyd had every right to be confident, for he had, of course, merely given people the simplest of odd-parity arrangements: an arrangement with one pair of tiles swapped. The puzzle was impossible! This was not realised for a considerable time after and the puzzle really did 'drive

the world mad'; the promise of a $1000 prize meant that the craze for the 14-15 Puzzle was not seen again until over a century later with the Rubik's Cube.

THE FIFTEEN PUZZLE ON COMPUTER

The program here is simply a version of the old Fifteen Puzzle. The program is totally self-contained, so just type it straight in and save under an appropriate name. On the model B set PAGE equal to &1200 before loading and running the program.

The cursor keys shunt the tiles around the board by indicating which tile should be moved into the gap. Q is used to quit that particular game, with the option of continuing, starting a new random arrangement, or quitting the program all together.

TECHNICAL DETAILS

The program is fairly easy to follow; most of the procedures are actually for displaying the puzzle and the movements, rather than manipulating the data.

PROCjumble is used to jumble the pieces around for the start of a new game. My method is the opposite of the mathematician's even-odd parity check: start with the tiles in order, pick two of them and swap them over, and repeat. If you do this an even number of times then the puzzle will, by definition, be solvable. The program stores the position of each of the pieces in array q, with zero representing the gap. The FOR-NEXT loop (lines 2010-2090) performs the swaps. If the TO number is even then the puzzles produced are always solvable, as in the listing; emulating the modern version. You can change line 2010 to:

```
2010 FOR j=1 TO RND(16)
```

This way you can let the computer decide whether you get a solvable arrangement or not. Finally, try playing around with the TO number in line 1970 to make the puzzle easier or harder. I'll leave you to decide how the choice of a high or low number generally makes the puzzle easier or harder!

BIBLIOGRAPHY

HORDERN, L. Edward.
Sliding Piece Puzzles.
Oxford University Press, 1986
ISBN 0-19-853204-0

SLOCUM, Jerry and BOTERMANS, Jack.
Puzzles Old and New. How to make and solve them.
Washington University Press, USA, 1986.
ISBN 0-295-96579-7

```
 10 REM Program Fifteen Puzzle
 20 REM Author  Peter Brown
 30 REM Version B 2.4
 40 REM BEEBUG  March 1993
 50 REM Subject to Copyright
 60 :
100 ON ERROR PROCerror
110 MODE 129
120 DIM q(16)
130 DIM L 10
140 *FX 200,1
```

```
 150 *FX 4,1
 160 VDU 19,2,4,0,0,0
 170 VDU 23,1,0,0,0,0,0,0,0,0
 180 PROCchardefs
 190 PROCsetscreen
 200 PROCjumble
 210 PROCdisplay
 220 REPEAT
 230 IF INKEY(-58)PROCup
 240 IF INKEY(-42)PROCdown
 250 IF INKEY(-26)PROCleft
 260 IF INKEY(-122)PROCright
 270 IF INKEY(-17)PROCquit
 280 PROCcheckwin
 290 UNTIL FALSE
 300 END
 310 ENDPROC
 320 :
1000 DEFPROCerror
1010 CLS
1020 REPORT
1030 PRINT" at line ";ERL
1040 *FX 200,0
1050 *FX 4,0
1060 END
1070 ENDPROC
1080 :
1090 DEFPROCdrawbox(left,bott,right,top
,xset,yset)
1100 right=right+1:top=top-1
1110 x=FNxtextgr(left):y=FNytextgr(bott
)
1120 MOVE x-xset,y-yset:x=FNxtextgr(rig
ht)
1130 DRAW x+xset,y-yset:y=FNytextgr(top
)
1140 DRAW x+xset,y+yset:x=FNxtextgr(lef
t)
1150 DRAW x-xset,y+yset:y=FNytextgr(bot
t)
1160 DRAW x-xset,y-yset
1170 ENDPROC
1180 :
1190 DEFFNxtextgr(a)=32*a
1200 DEFFNytextgr(a)=((32-a)-1)*32
1210 :
```

```
1220 DEFPROClarge(x,y,text$,dt)
1230 X%=L MOD 256:Y%=L DIV 256:A%=10:A=
10
1240 FOR i%=1 TO LEN(text$)
1250 ?L=ASC(MID$(text$,i%,1))
1260 CALL &FFF1
1270 IF dt=0 THEN VDU 23,224,L?1,L?1,L?
2,L?2,L?3,L?3,L?4,L?4:VDU 23,225,L?5,L?5
,L?6,L?6,L?7,L?7,L?8,L?8:VDU 31,x+i%-1,y
,224,8,10,225
1280 IF dt=1 THEN VDU 23,224,L?1,L?1,L?
1,L?2,L?2,L?2,L?3,L?3:VDU 23,225,L?3,L?4
,L?4,L?4,L?5,L?5,L?5,L?6:VDU 23,226,L?6,
L?6,L?7,L?7,L?7,L?8,L?8,L?8:VDU 31,x+i%-
1,y,224,8,10,225,8,10,226
1290 NEXT i%
1300 ENDPROC
1310 :
1320 DEFPROCsetscreen
1330 CLS
1340 GCOL 0,3
1350 PROCdrawbox(1,30,38,1,30,30)
1360 PROCdrawbox(1,30,38,1,15,15)
1370 GCOL 0,1
1380 PROCdrawbox(10,4,29,2,10,10)
1390 PROCdrawbox(10,4,29,2,18,18)
1400 PROClarge(11,2,"The Fifteen Puzzle
",1)
1410 PRINT TAB(13,5)"By Peter Brown"
1420 COLOUR 3
1430 PROCboard
1440 PROCtileborder
1450 PROCinstructions
1460 ENDPROC
1470 :
1480 DEFPROCchardefs
1490 VDU 23,200,78,209,81,81,81,81,81,2
38
1500 VDU 23,201,34,102,34,34,34,34,34,1
19
1510 VDU 23,202,70,201,65,66,68,72,72,2
39
1520 VDU 23,203,78,209,65,70,65,65,209,
238
1530 VDU 23,204,70,202,82,82,95,82,82,2
31
```

```
 1540 VDU 23,205,95,208,80,94,65,65,81,2
38
 1550 VDU 23,176,0,127,96,80,72,68,67,66
 1560 VDU 23,177,0,254,6,10,18,34,194,66
 1570 VDU 23,178,66,67,68,72,80,96,127,0
 1580 VDU 23,179,66,194,34,18,10,6,254,0
 1590 VDU 23,166,0,255,0,0,0,0,255,0
 1600 VDU 23,169,66,66,66,66,66,66,66,66
 1610 VDU 23,180,16,32,64,254,64,32,0,0
 1620 VDU 23,181,16,56,84,16,16,16,16,0
 1630 VDU 23,182,16,16,16,16,84,56,16,0
 1640 VDU 23,183,0,8,4,254,4,8,0,0
 1650 ENDPROC
 1660 :
 1670 DEFPROCinstructions
 1680 VDU 28,1,30,38,23
 1690 CLS
 1700 VDU 26
 1710 GCOL 0,1
 1720 PROCdrawbox(1,30,38,23,3,0)
 1730 PROCdrawbox(1,30,38,23,4,1)
 1740 arrow$=""
 1750 FOR d=180 TO 183
 1760 arrow$=arrow$+CHR$(d)+CHR$(32)
 1770 NEXT d
 1780 PROClarge(1,24,"Arrange the tiles
in the correct order",0)
 1790 PRINT TAB(6,27)"Use "+arrow$+"to s
hunt tiles."
 1800 PRINT TAB(12,29)"Press Q to quit."
 1810 GCOL 0,3
 1820 FOR g=10 TO 16 STEP 2
 1830 PROCdrawbox(g,27,g,27,5,5)
 1840 NEXT
 1850 PROCdrawbox(18,29,18,29,5,5)
 1860 ENDPROC
 1870 :
 1880 DEFPROCcheckwin
 1890 winflag=1
 1900 FOR k=1 TO 15
 1910 IF k<>q(k)THEN winflag=0
 1920 NEXT k
 1930 IF winflag=1 AND q(16)<>0 THEN win
flag=0
 1940 IF winflag=1 THEN PROCwin
 1950 ENDPROC
 1960 :
 1970 DEFPROCjumble
 1980 FOR j=1 TO 15
 1990 q(j)=j
 2000 NEXT
 2010 FOR j=1 TO 16
 2020 tempno1=RND(15)
 2030 REPEAT
 2040 tempno2=RND(15)
 2050 UNTIL tempno2<>tempno1
 2060 tempno3=q(tempno1):tempno4=q(tempn
o2)
 2070 q(tempno1)=tempno4:q(tempno2)=temp
no3
 2080 q(16)=0
 2090 NEXT
 2100 ENDPROC
 2110 :
 2120 DEFPROCdisplay
 2130 a=1
 2140 FOR y=9 TO 18 STEP 3
 2150 FOR x=15 TO 24 STEP 3
 2160 IF q(a)<10 AND q(a)<>0 THEN VDU 31
,x,y,48+q(a)
 2170 IF q(a)>9 AND q(a)<>0 THEN VDU 31,
x,y,200+q(a)-10
 2180 IF q(a)=0 THEN PROCblanktile(x,y)
 2190 a=a+1
 2200 NEXT
 2210 NEXT
 2220 ENDPROC
 2230 :
 2240 DEFPROCblanktile(x,y)
 2250 VDU 31,x-1,y-1,32,32,32,10,8,8,8,3
2,32,32,10,8,8,8,32,32,32
 2260 ENDPROC
 2270 :
 2280 DEFPROCleft
 2290 IF q(4)=0 OR q(8)=0 OR q(12)=0 OR
q(16)=0 THEN ENDPROC
 2300 FOR n=1 TO 16
 2310 IF q(n)=0 THEN pos=n
 2320 NEXT
 2330 q(pos)=q(pos+1)
 2340 q(pos+1)=0
 2350 PROCswap(pos)
```

```
 2360 PROCdisplay
 2370 ENDPROC
 2380 :
 2390 DEFPROCright
 2400 IF q(1)=0 OR q(5)=0 OR q(9)=0 OR q
(13)=0 THEN ENDPROC
 2410 FOR n=1 TO 16
 2420 IF q(n)=0 THEN pos=n
 2430 NEXT
 2440 q(pos)=q(pos-1)
 2450 q(pos-1)=0
 2460 PROCswap(pos)
 2470 PROCdisplay
 2480 ENDPROC
 2490 :
 2500 DEFPROCdown
 2510 IF q(1)=0 OR q(2)=0 OR q(3)=0 OR q
(4)=0 THEN ENDPROC
 2520 FOR n=1 TO 16
 2530 IF q(n)=0 THEN pos=n
 2540 NEXT
 2550 q(pos)=q(pos-4)
 2560 q(pos-4)=0
 2570 PROCswap(pos)
 2580 PROCdisplay
 2590 ENDPROC
 2600 :
 2610 DEFPROCup
 2620 IF q(13)=0 OR q(14)=0 OR q(15)=0 O
R q(16)=0 THEN ENDPROC
 2630 FOR n=1 TO 16
 2640 IF q(n)=0 THEN pos=n
 2650 NEXT
 2660 q(pos)=q(pos+4)
 2670 q(pos+4)=0
 2680 PROCswap(pos)
 2690 PROCdisplay
 2700 ENDPROC
 2710 :
 2720 DEFPROCtileborder
 2730 FOR y=9 TO 18 STEP 3
 2740 FOR x=15 TO 24 STEP 3
 2750 VDU 31,x-1,y-1,176,166,177,10,8,16
9,10,8,179,8,8,166,8,8,178,8,11,169
 2760 NEXT
 2770 NEXT
 2780 ENDPROC
```

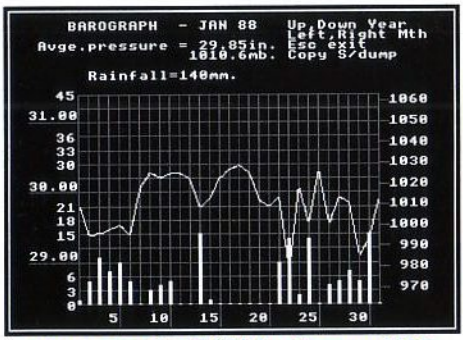
```
 2790 :
 2800 DEFPROCswap(pos)
 2810 IF pos=4 OR pos=8 OR pos=12 OR pos
=16 THEN xcol=3
 2820 IF pos=3 OR pos=7 OR pos=11 OR pos
=15 THEN xcol=2
 2830 IF pos=2 OR pos=6 OR pos=10 OR pos
=14 THEN xcol=1
 2840 IF pos=1 OR pos=5 OR pos=9 OR pos=
13 THEN xcol=0
 2850 IF pos<17 THEN yrow=3
 2860 IF pos<13 THEN yrow=2
 2870 IF pos<9 THEN yrow=1
 2880 IF pos<5 THEN yrow=0
 2890 VDU 31,14+(xcol*3),8+(yrow*3),176,
166,177,10,8,169,10,8,179,8,8,166,8,8,17
8,8,11,169
 2900 ENDPROC
 2910 :
 2920 DEFPROCboard
 2930 VDU 28,13,20,26,7
 2940 CLS
 2950 VDU 26
 2960 GCOL0,2
 2970 PROCdrawbox(14,19,25,8,15,15)
 2980 PROCdrawbox(14,19,25,8,10,10)
 2990 PROCdrawbox(14,19,25,8,25,25)
 3000 ENDPROC
 3010 :
 3020 DEFPROCquit
 3030 VDU 28,1,30,38,23
 3040 CLS
 3050 VDU 26
 3060 GCOL 0,1
 3070 PROCdrawbox(1,30,38,23,3,0)
 3080 PROCdrawbox(1,30,38,23,4,1)
 3090 PROClarge(5,24,"Are you sure you w
ant to quit ?",0)
 3100 PRINT TAB(3,27)"Press Y to re-star
t, N to continue"
 3110 PRINT TAB(7,29)"or Q to quit the p
rogram."
 3120 GCOL 0,3
 3130 PROCdrawbox(9,27,9,27,5,5)
 3140 PROCdrawbox(24,27,24,27,5,5)
 3150 PROCdrawbox(10,29,10,29,5,5)
 3160 *FX15,0
```

Continued on page 24

Heavy Weather

Nick Case adds to the improvements on his weather station program.

Having seen the improvements to my original weather station program on the BEEBUG Vol.11 No.6 disc (*BARTEMP*) I decided to go one better and the following changes to *that* version allow rainfall *and* pressure to be displayed together. In the educational context this should point up the relationship between pressure and rainfall. The ability to move from month to month or year to year, then dump your choice to a printer, is still available.



Pressure and rainfall for January 1988

Type in the changes, listed below, to the program BARTEMP and save it as *BARTEM2*. This program is very tight, so *no* extra spaces or REM's.

If it is all correct, when you run it you get the same display, requesting '<B>ar. or <T>emp.', the Temperature option will work as before. For the Barometer option to work, first go to all your b. files (e.g. *b.JAN'88*) and add ',rainfall(mm.)' at the end of line 3060REM. To every DATA line add the rainfall in mm. at the end of the line so that it reads date, pressure, rainfall (e.g.DATA 1,29.83,4). Until you have proper data, random figures will give the same effect.

A simple and cheap raingauge may be obtained from your local garden centre, so start noting down the rainfall with those barometer readings every day and soon you will have your first month's worth. Edit them into the DATA lines (e.g. *b.JAN'93*), SAVE them and away you go. Your original data for BARTEMP could be updated with rainfall figures from backnumbers of your local paper.

PROGRAM NOTES

The setup is as before with minor changes. The updated *PROCreadplot* is only used by the Temp selection, with the new *PROCrnplot* called from the main program for the Bar selection. The rainfall graduations are shown in blue on the left-hand side of the graph, in between the barometer readings (inches) in black. The maximum rainfall is 45mm. (more than 1.75 inches!) If you think that you live somewhere that gets more than that in one day (AGH!) increase the multiplier in line 695, e.g.PRINT r%*4, and the divisors in line 1470,e.g. ...hght%/4 (twice). If you live in a desert area you could decrease them. The new *PROCrnplot* produces the rainfall columns first using PLOT 85 to draw two triangles, (one of them inverted), then draws the pressure line. *PROCsum* will then print the month's averages.

```
 10 REM Program BarTem2
 20 REM Version B 1.03
 60 REM Adapted by N.J.Case
 70 *K.10 CH."BARTEM2"|M
120 dir$=CHR$(D%)+"."
130 :
230 :
270 O%=&400
300 IF T% hght%=25:wdth%=37:B$="TEMPER
```

```
ATURES":ELSE hght%=45:wdth%=32:B$="   BA
ROGRAPH"
  330 IF T% PROCreadplot:GOTO340
  335 PROCrnplot
  420 IF INKEY(-106) THEN PROCdump:found
%=0
  470 COLOUR2:PRINTTAB(1,1)B$
  680 FOR mb%=970 TO 1060 STEP10:MOVE 16
0+(30*wdth%),80+(((mb%/33.86)-28.40)*(5*
hght%)):DRAW 160+(31*wdth%),80+(((mb%/33
.86)-28.40)*(5*hght%))
  690 MOVE 160+(25*wdth%),95+(((mb%/33.8
6)-28.40)*(5*hght%)):PRINT mb%;:NEXT
  695 @%=&02:GCOL0,2:FOR z%=0 TO 2:FOR r
%=z% TO 15 STEP 5:MOVE 90,80+(r%*hght%)+
10:PRINT r%*3:NEXT:NEXT:@%=&90A
  720 DEF PROCreadplot:totalmax%=0:total
min%=0
  750 :
  800 :
  870 VDU5:PLOT 5,160+(X-1)*wdth%,80+(Y-
28.4)*(5*hght%):VDU4
```

```
  925 COLOUR2:PRINTTAB(6,6)"Rainfall=";t
otalrain;"mm."
 1370 :
 1410 :
 1430 DEF PROCrnplot
 1440 VDU5:GCOL0,2:totalrain=0:totalpres
s=0
 1450 RESTORE3040:READend:FOR N=1 TO end
:READ X,Y,R:totalpress=totalpress+Y:tota
lrain=totalrain+R
 1460 MOVE160+(X*wdth%)-(wdth%-4),80:MOV
E160+(X*wdth%)-(wdth%+4),80
 1470 PLOT85,160+(X*wdth%)-(wdth%-4),80+
(R*hght%/3):PLOT85,160+(X*wdth%)-(wdth%+
4),80+(R*hght%/3):NEXT
 1480 RESTORE3040:READend:READ X,Y,R:MOV
E160,80+((Y-28.4)*(5*hght%)):GCOL0,1:FOR
 N=2 TO end:READ X,Y,R:PLOT 5,160+((X-1)
*wdth%),80+((Y-28.4)*(5*hght%)):NEXT
 1490 VDU4
 1500 ENDPROC
 1510 :
```

The Fifteen Puzzle (continued from page 22)

```
 3170 g$=GET$
 3180 IF g$="Y" OR g$="y" THEN PROCjumbl
e:PROCboard:PROCtileborder:PROCdisplay:P
ROCinstructions:ENDPROC
 3190 IF g$="Q" OR g$="q" THEN PROCleave
 3200 PROCinstructions
 3210 ENDPROC
 3220 :
 3230 DEFPROCwin
 3240 VDU 28,1,30,38,23
 3250 CLS
 3260 VDU 26
 3270 PROCdrawbox(1,30,38,23,3,0)
 3280 PROCdrawbox(1,30,38,23,4,1)
 3290 PROClarge(6,24,"Phew! Finished! We
ll done!",1)
 3300 PRINT TAB(9,28)"Again ? Press Y or
 N"
 3310 PROCdrawbox(23,28,23,28,5,5)
```

```
 3320 PROCdrawbox(28,28,28,28,5,5)
 3330 *FX15,0
 3340 g$=GET$
 3350 IF g$="Y" OR g$="y" THEN PROCjumbl
e:PROCboard:PROCtileborder:PROCdisplay:P
ROCinstructions:ENDPROC
 3360 PROCleave
 3370 ENDPROC
 3380 :
 3390 DEFPROCleave
 3400 CLS
 3410 PRINT"Bye!"
 3420 *KEY 0 RUN|M
 3430 PRINT"Press the red key f0 to re-s
tart."
 3440 *FX 200,0
 3450 *FX 4,0
 3460 END
 3470 ENDPROC
```

Troubleshooting Guide (Part 3)

Gareth Leyshon helps take the pain out of printers.

The most common BBC accessory after the disc drive is the printer, which connects to the computer in one of two ways. It may connect to the socket next to the disc drive socket underneath the keyboard, via a ribbon cable. This is known as the parallel printer port. Otherwise it will connect to the serial port, marked *RS423*, at the back of the computer.

Connecting the printer lead is a source of many headaches. With parallel printers the plug on the lead must be pushed firmly into the socket, so that the side lugs click themselves into place. A serial printer's DIN plug, connecting to the RS423 port (which is also used by modems), is easily inserted upside down, a common cause of printers or modems refusing to work - the only solution is trial and error. If you're given a choice for your machine, it's probably better to use the parallel port as this leaves the RS432 free, and works faster.

TALK TO YOUR COMPUTER

You will need to tell your computer what sort of printer you are using. A Master checks its configuration setting while a Beeb will assume that you are using a parallel printer unless you tell it to use the serial port. To do this type *FX5,2 which is effective until a Ctrl-Break. *FX5,1 selects the parallel port again. In the same vein, *CONFIGURE PRINT 1 selects parallel on the Master; use *CONFIGURE PRINT 2 for serial: *CONFIGURE PRINT 0 disables printing. You may have to press Ctrl-Break after *CONFIGURE commands to make them work.

When unpacking a new printer, first check to see if there are any polystyrene or plastic restrainers inside the case to protect it in transit - the mechanism can jam on these if they are not removed. Many printers have a built-in test routine, which automatically prints all the letters it can; check your printer manual to see how this is activated.

Standard printing width is 80 characters per line, with 66 lines per page; most computer stationery is made for this. A4 paper is slightly narrower (it still takes 80 characters) but 70 lines long. If you are using tractor-feed paper held on by sprockets, think about how you want to use the paper. Usually the sprocket pins are 'downstream' of the printing head. If you want to print lots of short pieces, put the perforations in the paper downstream of the sprockets. If you are printing a document several pages long, set the perforations just above the level of the printing head. This should make the computer start a new page in the right place. Switch the printer off and on to reset it in this position or press the *Top of Form* button if there is one.

Aligning the head with the perforations for short documents proves awkward as the perforations don't end up where you want to tear them so, unless you need the printing to start close to the top of the sheet, it's easier to put the perforations beyond the sprockets. If the sprockets are upstream of the print-head, you don't have a problem.

Next, you must get acquainted with the controls on your printer. There will be at least three buttons, marked *On line*, *Form Feed*, and *Line Feed*, a power switch and various levers. Even though the printer's power is on it will still not

act on instructions from the computer unless it is also placed on-line; there is usually an LED, often an LED which lights up when it *is* on line. To pause a printer in mid-print, to change the paper for instance, press the on line button to stop it, change the paper and press again - the printer carries on where it left off. Beware, however, that your printer may beep angrily at you and that some software may stop, assuming a printer error.

Most printers have some internal memory that holds information coming from the computer and various details about printing style. Turning the printer's power switch off will wipe its internal memory, so turning it back on will reset it. It forgets if it was printing in a special mode, such as bold or underlining, and the next few lines which it was going to print. Form Feed ejects a whole page while Line Feed moves the paper up a line, but these usually (though not invariably) only work when the printer is off line.

If the printer only has a limited memory the computer has to wait for it to finish, and appears to *hang up*. If the computer is waiting for the printer, both the CAPS LOCK and SHIFT LOCK lights come on.

When the printer has its power turned off and on, it forgets what has already been sent, but the computer may still have data to send. To totally interrupt a printout, you must stop the computer sending data (use Escape, or Break if there's no other way) so that at least one of the CAPS LOCK/SHIFT LOCK LEDs goes out. Only then can you put the printer back on.

Some printers have a *Print Mode* selector switch, which selects print styles. High quality is known as Letter Quality (LQ) on high-resolution printers, or Near Letter Quality (NLQ) on lower resolution machines. The switch may not take effect if moved in the middle of a print run, only causing a style change when the printer finishes printing or is put off and back on line.

Virtually all printers have a paper lock or friction/tractor lever. This usually has two positions, which may be labelled F and T. In the locked or F(riction) position, the paper is friction fed and can only be moved by pressing Line Feed, Form Feed or turning a knob on the printer case. In the unlocked or T(ractor) position, the paper is free to be adjusted and aligned as required. Loose leaf paper is aligned on T and used on F; when tractor feed paper is installed, it should be pulled through by friction, clamped over the sprocket-belts and used in T-mode.

A few printers have adjustable sprockets which can be placed downstream of the paper to pull it or upstream to push, in which case you may have a three-position lever marked F, T-pull and T-push. Beware of clamping tractor feed paper on to the sprockets but leaving the switch on friction. Slight differences between the distance fed by the sprocket drive and the friction drive can cause the paper to jam in the sprockets after prolonged printing.

Another lever, often hidden inside the case, varies the distance between the printing head and the paper. Moving the head closer to the paper provides darker printing, and can compensate for a fading printer ribbon for a time. If printing on a thick or doubled sheet of paper, such as an envelope, you should move the head further away from the paper.

Eventually a fading ribbon will need to be replaced; some contain an extra portion of ink which is made available when you push a pencil into the marked hole. Some companies offer a ribbon re-inking service. Look out, however, for the fabric of the ribbon becoming too frayed; once this is too thin then it will be of no further use.

Most printers have a *paper-out* detector an inch or two upstream of the printing head, which stops the printout before the paper runs out. To print closer to the bottom of the page, you can stick a piece of paper down the input slot to push the detector lever back into place - but be careful, don't push the second sheet down so far that it catches in the roller, and pull it out or put the printer off line before the main sheet comes to the end. If you find the paper-out detector is really irritating you may find it can be switched off using a DIP switch - see below.

Your main role for a printer will probably be to print results from the software you use. However, it is worth knowing how to use a printer directly for simple jobs. Normally, all the computer's output goes to the screen. If you want everything displayed on screen to be sent to the printer as well, type VDU 2 or press Ctrl-B - the two operations have exactly the same effect. Make sure that the printer is ready - there should be paper in and the on line LED should be illuminated. Now anything that is displayed on the computer screen will also be sent to the printer. For instance, to get a paper record of a Master's status, simply type *STATUS and the results appear on the printout. To conclude the printing, wait for the ready prompt (the arrow '>') and type VDU 3 or press Ctrl-C (if using Ctrl-C the arrow '>' will be sent to the printer first, but isn't printed until the printer is given a new line). You should note that mode 7 screen-control characters will not appear.

Such use of Ctrl-C or VDU 3 may also halt a printout if you find that the computer hits a problem (say a program stops due to an error while in a printout routine).

DEEPLY DIPPY

Many of the problems experienced with printers are due to the computer and printer not being set up in a compatible way. Printers are configured using a set of eight or so switches, called DIP switches, usually placed deep inside, but with an access hole. The switches can be thrown with the point of a pencil and control various functions. For instance, the printer needs to know whether it should feed forward the paper when it receives an end-of-line (carriage return) code. The computer can also be set to either force a new line or let the printer do it. So if all your text prints on the same line, or if a blank line is left between each line of print, it's probable that either both or neither device has been told to feed the paper.

Other DIP switches may control whether you are using single sheet or perforated tractor fed (fan-fold) paper, or whether the printer stops when the paper passes the sheet-end detector. Check your printer manual for details. A serial-port printer may have a second set of switches which controls how fast it receives data from the computer - sometimes called the BAUD rate.

Beeb users should note that certain commands, effective until Ctrl-Break is pressed, can be applied from the computer: *FX6,0 tells the computer to send line feeds at the end of a line (curing all the text appearing on the same line) while its antidote, *FX6,10 disables

line feeds, curing double-spaced lines. Master users should use *CONFIGURE IGNORE 10 to disable line feeds, but use *CONFIGURE NOIGNORE to send the line feeds.

Beeb users with a serial printer must use *FX 8,*n* to set the speed at which the computer talks to their machine. Master: *CONFIGURE BAUD *n*. Find the highest speed (baud rate) your printer works at and use the appropriate *n*. If the printer doesn't work reliably at this speed, try a slower baud rate setting.

BAUD RATES AND N-VALUES

n	Baud
1	75
2	150
3	300
4	1200
5	2400
6	4800
7	9600
8	19,200

A different problem concerns the pound sign. Most printers are exported to several countries and have several international character sets built in, any set being selected by a combination of DIP switches. The exact details depend on the printer (see the manual), but typically, in *USA* mode the BBC key marked '#' will give a hash symbol on the printer, while the key marked '£' will give a back-apostrophe. In the *English* mode the pound sign still gives the back-apostrophe while the hash now gives the '£' symbol on paper. The simplest solution is to set your DIP switches to English mode and always type '#' where you need '£'. If, however, you don't need to print a pound sign but do need a hash, leave the printer in USA mode. There are short machine code programs around that will solve this problem.

In most cases, a word processor or other program will be used to print your text. This will take care of codes for such functions as underlining, unless the printer uses a non-standard set of codes. Printers which use the standard set of codes are said to be *Epson compatible*: see the article in BEEBUG Vol.10 No.6 for more information. If you don't get the underlining or bold print you expect with a word processor or other software, perhaps the printer is non-standard. Some software can have its printer codes changed - you will find the codes in the printer manual, and the method of change in the word processor handbook. A word processor will often ensure that the keyboard pound sign produces a properly printed pound, too.

If you can set the width on your word processor, you may find that when printed, some long lines of text spill over on to the next printed line, leaving a large space on the right. Some systems, when working with an 80 column printer, need to be set to a page width of 79 to compensate for this. If printing in double-width or in two columns, you will need a width of 39 rather than 40.

There is no simple way to make a printer produce graphics. So-called *daisy wheel* printers cannot produce graphics at all, being confined to the characters available on the printing wheel installed; while a dot-matrix printer needs a suitable *printer dump* utility to generate graphics out of the dots.

You might not be able to print out a glossy edition of BEEBUG, but by now you should be able to produce a neat document with no paper jams. Next time we change gear and go for a spin to investigate problems and solutions with disc drives.

BEEBUG WORKSHOP

Tree Structures (Part 4)

by Mike Williams

INSERTING AND DELETING NODES

In all the work which we have contemplated on trees we have ignored how data may be inserted or deleted within a tree structure in a way which reflects any ordering of the data. We have used DATA statements as a simple means of specifying what data should be at any node. To conclude this particular Workshop we will now examine how data can be represented in a tree structure such that Inorder traversal represents an ordering of the data.

Let us assume that a tree structure already contains data. The Inorder approach ensures that the 'lowest' data item appears in the left-most node, and the 'highest' in the right-most node. To insert a new data item, we need to progress through the tree structure until we can determine the correct position for the new item. We must then create a new node in which to store this data, and complete appropriate links to other nodes. To delete a data item, we follow a similar procedure until the data item is located. Its node must then be removed by reassigning any links pointing to or from that node.

In some ways what we are attempting to achieve is not dissimilar to the implementation of doubly-linked lists which I gave in the Workshop in BEEBUG Vol.11 No.5. The difference lies in the way in which elements (or nodes) are linked together. The tree structure is particularly advantageous as a means of locating a piece of data (and trees are used for indexing in database systems for that reason), as it is the depth or number of levels in the tree which determines the search time, whereas the search time for a list is proportional to the length of the list.

The principal question, of course, is how we handle branches of a tree when inserting a new node, and what to do with the cut-off branches when deleting an existing node, remembering that we wish to preserve the structure of the tree in Inorder at all times. The rules for these functions have long been well tried and tested and can be stated as follows:

Insertion

Starting at the root, compare the new value with that at the current node; if the new value is less then follow the lefthand pointer if there is one and repeat this function in relation to the new node, otherwise place the new value as the lefthand node;

otherwise follow the right-hand pointer if there is one and repeat the whole function in relation to this node,

otherwise place the new value as the right-hand node.

Deletion

Locate the node to be deleted; the pointer to the deleted node must be made to point to the left sub-tree of the deleted node; the right sub-tree then becomes the right-most element of the left sub-tree of the deleted node.

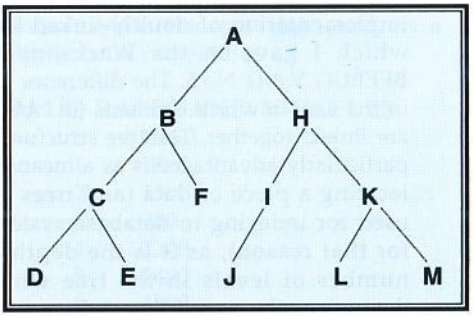


Figure 1. Initial tree

This may be easier to understand by looking at two examples. First we will add 'G' to the tree of figure 1 (assuming that the letters are ordered alphabetically). Compare 'G' with the value of the root; it is less than so follow the left sub-tree. Compare next 'G' with 'B'; it is greater so follow the right sub-tree. Then compare 'G' with 'F'; it is less than so follow the left sub-tree. Because there is no path to follow 'G' becomes the left sub-tree of 'F' (see figure 2).

Now we will follow the process of deleting 'H' from the tree of figure 2. Having located 'H', it effectively disappears from view by taking its left sub-tree and appending this to the pointer from 'A', and the cut-off right sub-tree of 'H' becomes the right sub-tree of 'I'.

This month's listing is an extension of routines which we have published recently in our discussion of tree structures. It is complete with a menu which allows data to be inserted or deleted, as well as an individual item to be located, or all the data to be output in Inorder order.

The insertion process is symmetrical depending on whether or not the route followed uses a left-hand or a right-hand path. This is reflected in the two procedures PROCleft and PROCright. These should be viewed as procedures local to PROCadd (line 1000), although BBC Basic makes no such distinction.

To delete a node we rely first on a function, FNfind, to locate the node. The function FNdelete (line 1190 and following) then deals with the hanging left and right-hand branches in turn.

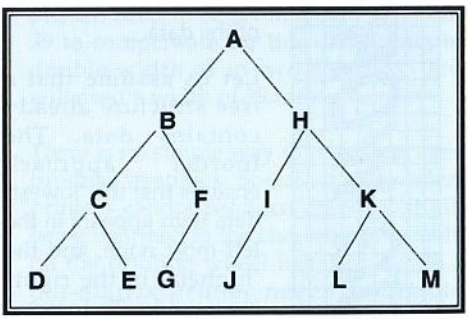


Figure 2. After adding 'G'

A list of free elements is maintained as a linked list as we have done before using just the right-hand pointers to provide the links. When a node is added to the tree structure, an empty element is taken from the head of the free list; when a node is deleted, its element is returned to the head of the free list.

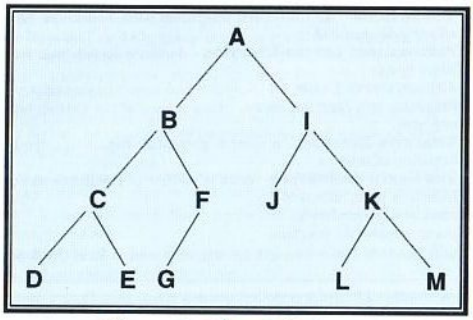


Figure 3. After deleting 'H'

I recommend that you try out the program, and also examine it thoroughly to see how the theory described in this and previous articles on the subject are translated into practice. As is often the case, coping with special cases, such as an empty tree, provide some of the most taxing programming challenges.

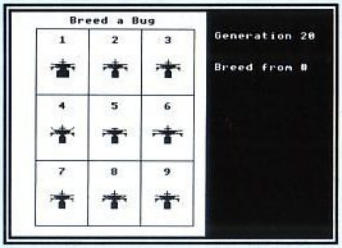
Next month I hope to tackle something new, but if readers have any suggestions for subjects they would like to see dealt with in the Workshop columns then please let me know.

```
  10  REM Program Tree04
  20  REM Version B 1.0
  30  REM Author  Mike Williams
  40  REM BEEBUG  March 1993
  50  REM program subject to copyright
  60  :
 100 ON ERROR REPORT:PRINT" at ";ERL:EN
D
 110 DIM Data(500),LLink(500),RLink(500
)
 120 FOR I%=0 TO 499:RLink(I%)=I%+1:NEX
T
 130 root=-1:free=0:end%=FALSE
 140 :
 150 REPEAT
 160 MODE7:choice=FNmenu
 170 IF choice=1 THEN PROCinput_data
 180 IF choice=2 THEN PROCdelete_data
 190 IF choice=3 THEN PROCfind_data
 200 IF choice=4 THEN PROCinorder1(root
)
 210 IF choice=5 THEN end%=TRUE
 220 UNTIL end%
 230 END
 240 :
1000 DEF PROCadd_element(data,node)
1010 LOCAL f%:f%=FALSE
1020 REPEAT
1030 IF data<=Data(node) THEN PROCleft
ELSE PROCright
1040 UNTIL f%
1050 ENDPROC
1060 :
1070 DEF PROCleft
1080 IF LLink(node)>-1 THEN node=LLink(
node):ENDPROC
1090 newnode=free:free=RLink(free):LLin
k(newnode)=-1:RLink(newnode)=-1
1100 LLink(node)=newnode:Data(newnode)=
data:f%=TRUE
1110 ENDPROC
1120 :
1130 DEF PROCright
1140 IF RLink(node)>-1 THEN node=RLink(
node):ENDPROC
1150 newnode=free:free=RLink(free):LLin
k(newnode)=-1:RLink(newnode)=-1
1160 RLink(node)=newnode:Data(newnode)=
data:f%=TRUE
1170 ENDPROC
1180 :
1190 DEF FNdelete_element(data)
1200 LOCAL node,L,R:node=FNfind_element
(data)
1210 IF node=-1 THEN =node
1220 REM ** Return node to free list **
1230 R=RLink(node):RLink(node)=free:fre
e=node
1240 REM ** Assign left subtree **
1250 IF root=node THEN root=LLink(node)
ELSE IF t$="L" THEN LLink(tnode)=LLink(
node) ELSE RLink(tnode)=LLink(node)
1260 REM ** Assign right subtree **
1270 L=LLink(node)
```

Continued on page 58

Best of BEEBUG

Applications I Disc



BUSINESS GRAPHICS - for producing graphs, charts and diagrams
VIDEO CATALOGUER - catalogue and print labels for your video cassettes
PHONE BOOK - an on-screen telephone book which can be easily edited and updated
PERSONALISED LETTER-HEADINGS - design a stylish logo for your letter heads
APPOINTMENTS DIARY - a computerised appointments diary
MAPPING THE BRITISH ISLES - draw a map of the British Isles at any size
SELECTIVE BREEDING - a superb graphical display of selective breeding of insects
THE EARTH FROM SPACE - draw a picture of the Earth as seen from any point in space
PERSONALISED ADDRESS BOOK - on-screen address and phone book
PAGE DESIGNER - a page-making package for Epson compatible printers
WORLD BY NIGHT AND DAY - a display of the world showing night and day for any time and date of the year

File Handling for All

on the BBC Micro and Acorn Archimedes

by David Spencer and Mike Williams

Computers are often used for file handling applications yet this is a subject which computer users find difficult when it comes to developing their own programs. *File Handling for All* aims to change that by providing an extensive and comprehensive introduction to the writing of file handling programs with particular reference to Basic.

File Handling for All, written by highly experienced authors and programmers David Spencer and Mike Williams, offers 144 pages of text supported by many useful program listings. It is aimed at Basic programmers, beginners and advanced users, and anybody interested in File Handling and Databases on the Beeb and the Arc. However, all the file handling concepts discussed are relevant to most computer systems, making this a suitable introduction to file handling for all.

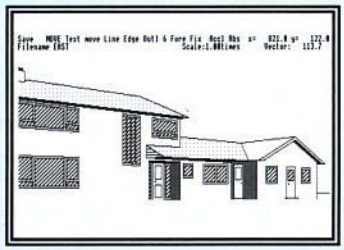
The book starts with an introduction to the basic principles of file handling, and in the following chapters develops an in-depth look at the handling of different types of files e.g. serial files, indexed files, direct access files, and searching and sorting. A separate chapter is devoted to hierarchical and relational database design, and the book concludes with a chapter of practical advice on how best to develop file handling programs.

The topics covered by the book include:

Card Index Files, Serial Files, File Headers, Disc and Record Buffering, Using Pointers, Indexing Files, Searching Techniques, Hashing Functions, Sorting Methods, Testing and Debugging, Networking Conflicts, File System Calls

The associated disc contains complete working programs based on the routines described in the book and a copy of Filer, a full-feature Database program originally published in BEEBUG magazine.

ASTAAD



Enhanced ASTAAD CAD program for the Master, offering the following features:

- full mouse and joystick control
- built-in printer dump
- speed improvement
- STEAMS image manipulator
- Keystrips for ASTAAD and STEAMS
- Comprehensive user guide
- Sample picture files

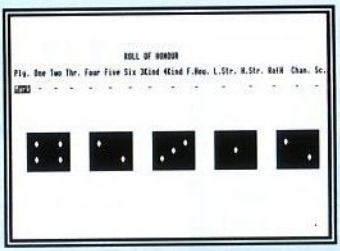
	Stock Code	Price		Stock Code	Price
ASTAAD (80 track DFS)	1407a	£5.95	**ASTAAD** (3.5" ADFS)	1408a	£5.95
Applications II (80 track DFS)	1411a	£4.00	**Applications II** (3.5" ADFS)	1412a	£4.00
Applications I Disc (40/80T DFS)	1404a	£4.00	**Applications I Disc** (3.5" ADFS)	1409a	£4.00
General Utilities Disc (40/80T DFS)	1405a	£4.00	**General Utilities Disc** (3.5" ADFS)	1413a	£4.00
Arcade Games (40/80 track DFS)	PAG1a	£5.95	**Arcade Games** (3.5" ADFS)	PAG2a	£5.95
Board Games (40/80 track DFS)	PBG1a	£5.95	**Board Games** (3.5" ADFS)	PBG2a	£5.95

All prices include VAT where appropriate. For p&p see Membership page.

RISC Developments Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS.

Best of BEEBUG

Board Games



Solitaire - an elegant implementation of this ancient and fascinating one-player game, and a complete solution for those who are unable to find it for themselves.

Roll of Honour - Score as many points as possible by throwing the five dice in this on-screen version of 'Yahtze'.

Patience - a very addictive version of one of the oldest and most popular games of Patience.

Elevenses - another popular version of Patience - lay down cards on the table in three by three grid and start turning them over until they add up to eleven.

Cribbage - an authentic implementation of this very traditional card game for two, where the object is to score points for various combinations and sequences of cards.

Twiddle - a close relative of Sam Lloyd's sliding block puzzle and Rubik's cube, where you have to move numbers round a grid to match a pattern.

Chinese Chequers - a traditional board game for two players, where the object is to move your counters, following a pattern, and occupy the opponent's field.

Aces High - another addictive game of Patience, where the object is to remove the cards from the table and finish with the aces at the head of each column.

Applications II Disc

Crossword Editor - for designing, editing and solving crosswords

Monthly Desk Diary - a month-to-view calendar which can also be printed

3D Landscapes - generates three dimensional landscapes

Real Time Clock - a real time digital alarm clock displayed on the screen

Running Four Temperatures - calibrates and plots up to four temperatures

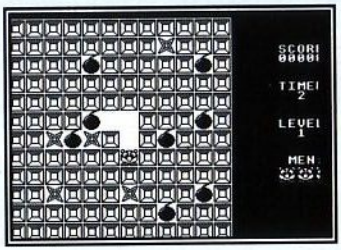
Julia Sets - fascinating extensions of the Mandelbrot set

Foreign Language Tester - foreign character definer and language tester

Share Investor - assists decision making when buying and selling shares

Label Processor - for designing and printing labels on Epson compatible printers

Arcade Games



George and the Dragon - Rescue 'Hideous Hilda' from the flames of the dragon, but beware the flying arrows and the moving holes on the floor.

Ebony Castle - You, the leader of a secret band, have been captured and thrown in the dungeons of the infamous Ebony Castle. Can you escape back to the countryside, fighting off the deadly spiders on the way and collecting the keys necessary to unlock the coloured doors?

Knight Quest - You are a Knight on a quest to find the lost crown, hidden deep in the ruins of a weird castle inhabited by dangerous monsters and protected by a greedy guardian.

Pitfall Pete - Collect all the diamonds on the screen, but try not to trap yourself when you dislodge the many boulders on your way.

Builder Bob - Bob is trapped on the bottom of a building that's being demolished. Can you help him build his way out?

Minefield - Find your way through this grid and try to defuse the mines before they explode, but beware the monsters which increasingly hinder your progress.

Manic Mechanic - Try to collect all the spanners and reach the broken-down generator, before the factory freezes up.

Quad - You will have hours of entertainment trying to get all these different shapes to fit.

	Stock Code	Price		Stock Code	Price
File Handling for All Book	BK02b	£ 9.95			
File Handling for All Disc (40/80T DFS)	BK05a	£ 4.75	**File Handling for All** Disc (3.5" ADFS)	BK07a	£ 4.75
Joint Offer book and disc (40/80T DFS)	BK04b	**£ 11.95**	**Joint Offer book and disc** (3.5" ADFS)	BK06b	**£ 11.95**
Magscan (40 DFS)	0005a	£ 9.95	**Magscan Upgrade** (40 DFS)	0011a	£ 4.75
Magscan (80T DFS)	0006a	£ 9.95	**Magscan Upgrade** (80T DFS)	0010a	£ 4.75
Magscan (3.5" ADFS)	0007a	£ 9.95	**Magscan Upgrade** (3.5" ADFS)	1458a	£ 4.75

All prices include VAT where appropriate. For p&p see Membership page.

512 Forum

by Robin Burton

As mentioned in the last issue, this month we'll take a look at PC graphics standards. I'll also tell you the end of the story about directory names, which will presumably be a surprise to all but one Forum reader.

HARDWARE CONSIDERATIONS

Compatible software has featured in the Forum recently and since the display is increasingly the problem these days an outline of PC display standards might be illuminating. It should clarify why some programs will never work in the 512 and might provide an insight into programs that do stand a chance before you try them, i.e. before you spend money. After all, even though shareware is by far the cheapest and best source of programs for the 512, it's not entirely free.

First let's summarise the 512's hardware. The 512's physical display is handled by the BBC micro. Obviously if the BBC micro can't display something, then the 512 can't either. That's where we start. The story begins ten years ago.

The 512's display problems are a consequence of the BBC micro's design, specifically its limited screen RAM. Screen RAM is limited because it's part of main memory, which in turn is limited by the maximum addressing range of the 8-bit 6502 processor, 64K bytes. Of this the MOS, a language and a bit of workspace take over half leaving, even in a Master, less than 32K for data, programs and display RAM. Shadow RAM eases the strain a little, but the 64K limit in the processor is the real block on any attempt at true memory expansion, whether for screen RAM or for programs and data.

There are two ways round such a limit. One is to change the processor to a 16-bit or 32-bit type that can address more memory. The other is to separate display RAM totally from main memory. Naturally, the two solutions combined are even better, but of course both increase the complexity, hence the cost of hardware. For the BBC micro's intended market the costs weren't acceptable. Bear in mind too, that when the BBC micro first appeared its graphics were pretty hot stuff, close to state of the art both within its price bracket and a bit beyond, as you'll see.

PC DESIGN

In contrast, PCs (apart from the best forgotten PC Junior) were aimed at business, so a machine's manufacturing costs weren't such a limitation. Because of this, PCs use a modular approach, where the basis of the system is the processor and its main circuit board plus a keyboard, but just about everything else, including RAM, is a plug-in extra.

This obviously allows a wide range of basic machine configurations, giving the purchaser more choice. Just as important, individual items can be replaced easily and cheaply if they fail or are superseded by new, higher specification hardware options. This design philosophy is precisely why there have been so many PC display standards.

If you look inside any PC from an XT onwards, apart from the very earliest machines and a few odd ones, you'll find that floppy disc control, hard disc control, serial ports, the parallel port and the display controller are all provided by extra, plug-in interface boards. These are called expansion cards, which plug into expansion-slots on the main board. Six to eight expansion slots is normal in PCs today.

Using more sophisticated, hence smaller and fewer chips than earlier cards, it's common to find several functions grouped together on a modern expansion board, particularly if they're facilities that logically belong together and that virtually everyone needs. For example AT-bus cards (known as ISA - Industry Standard Architecture, by far the most common PC architecture) will typically have two serial ports plus a parallel port, sometimes with a games port too, on one card called an IO card. A second card usually caters for two floppy drives plus two hard (winchester) drives.

In most machines this leaves only display support and, interestingly, display support is usually provided by a dedicated card, for various reasons. For one, display standards change more quickly than those for other peripherals, so a graphics adaptor and screen are likely candidates for upgrade during a machine's lifetime.

Also, it's not unusual to have two graphics adaptors and two screens on a PC. My old XT for example has a mono card and screen, plus a switchable CGA/EGA card and colour screen. Both screens can be used together, so I can run a program in the debugger on one and watch the program's output on the other at the same time. This two screen technique is most frequently used for separating very high definition graphics from text. In fact one or two 'high-end' applications almost demand two screens. They'll run on one, but they tend to be much less convenient to use.

Naturally, putting other interfaces on a display card would increase its cost, so it's never been considered a good idea.

Nowadays there's a further consideration. Even the most meagre VGA card has 256K of RAM with slots allowing up to a megabyte, which is both more useful and more usual these days. In fact the most sophisticated (but not particularly specialised and rapidly becoming standard for Windows and CAD) graphics cards now have up to 3 megabytes of RAM driven by a dedicated processor, using a special colour chip called a Ramdac. As you can imagine, these cards are pretty full already without other hardware.

PC GRAPHICS

The main point to appreciate is that the way a PC outputs display data depends on the display card being used, rather than on the PC itself or its main circuit board as is the case in the BBC micro.

The most basic PC display is a monochrome display adaptor (MDA). These are still useful for text, though mono VGA is the standard these days. MDA cards contain screen refresh circuitry, hardware connections and little else, so MDA is the cheapest display type and, incidentally, it supports no graphics whatsoever. Clearly MDA programs should run in the 512, at least as far as screen output is concerned. Because it's an old standard, there should be few problems in other areas too.

Hercules was a mono display which did support graphics. Although popular for a time it never became a real standard for two reasons. It wasn't an IBM product and most graphics users wanted colour, not mono.

CGA (Colour Graphics Adaptor) was the first colour display standard for PCs, with a resolution similar to the BBC micro's but with extras, particularly for text. CGA offers two graphics modes, either 320x200 pixels in four colours or 640x200 in two (note that in DOS graphics modes are quite distinct from text modes: see table). The BBC micro's displays range from 640 by 256 in two colours (mode 0), through 320 by 256 in four colours (mode 1) to 160 by 256 in 16 colours (mode 2), so the humble BBC can actually surpass CGA graphics

both in the number of colours or maximum resolution, so where's the problem?

Mode	Display resolution
0	40x25 B&W text, colour adaptor
1	40x25 16 Colour text
2	80x25 B&W text, colour adaptor
3	80x25 16 Colour text
4	320x200 4 colour graphics (colour burst on)
5	320x200 4 colour graphics (colour burst off)
6	640x200 2 colour graphics
7	Monochrome adaptor text
8	160x200 16 colour graphics (PC Jnr)
9	320x200 16 colour graphics (PC Jnr)
10	640x200 4 colour graphics (PC Jnr)

DOS 3.x MDA/CGA & PC Junior display modes

Well, first you can see that, although two of the horizontal resolutions match exactly, none of the verticals do. DOS's 320 by 200 in four colours is nearest to the BBC's mode 1, so this is what the 512 uses for four colour graphics with good results for programs that work. Likewise 640 by 200 CGA display almost matches BBC mode 0, so this is where the 512's two colour graphics come from. As you can see, the BBC's vertical display resolution is actually 25% better than CGA's. As I said, the BBC was close to the state of the art at the time (unless you were paying five figures).

Of course CGA also displays text, but here the problems are potentially worse. A CGA 40 or 80 character by 25 line display can include up to 16 foreground colours. These are the usual seven plus black (as in the BBC), plus high-intensity versions of the seven plus grey. The first eight colours can also be used for background and while all this is going on (120 sensible combinations so far) foreground characters can flash and, if that's not enough they can be underlined. There's no limit to how all these are used, so any character can be in any combination of the above, regardless of neighbouring characters.

For text the 512 uses BBC mode 3, with the BBC's 6845 chip reprogrammed to move the lines together, unless you prefer 'PCSCREEN 7', which is standard BBC mode 3 with gaps between lines. As we know, mode 3 is two colour only, so the 512's text output can offer only standard, inverse, bold and underlined characters.

This is clearly an extremely limited CGA text emulation, much, much worse than for graphics. In fact it would be more accurate to say the 512 provides CGA graphics, but only MDA for text modes. If you have a text program that insists on exploiting CGA colour with no mono option (or no user settings for specific colours) you'll end up with unsightly bold characters, invisible text, a glaring background or sometimes a mixture of all three.

You can see why you should configure 512 applications for a mono display when there's a choice. In monochrome (using B&W text modes) programs will restrict themselves to normal, inverse and underlined text, so the author will (unwittingly) already have made the right choices for an acceptable (i.e. visible and legible) display in the 512.

All in all it's surprising that the 512 successfully runs more text programs than graphics, since it's potentially superior to CGA in colour graphics, but a long way short in text modes.

As usual there's more to it than just the numbers. Some PC graphics programs poke RAM directly, as do many BBC programs. The reason is that DOS graphics support is limited to setting physical to logical colours and reading or writing a single pixel. It's pretty slow too (forget PLOT, MOVE and DRAW: in DOS it's all strictly DIY). Just as in the BBC micro, programs that poke RAM directly won't work in other hardware. Consider: we all know of Master programs that

won't run in a model B/B+ (and vice-versa) and they're much more closely related than the 512 and a PC.

Legal graphics and text output is handled remarkably well by the 512's XIOS, even down to ROM BIOS level, although unmodified graphics are squashed 20% vertically for obvious reasons. On the other hand, direct memory pokes by-pass the XIOS, so they never get across the tube to the display. Result? A blank screen!

It's worth remembering that in some CGA programs it can seem that the 512 has crashed. However, if you know how to exit to DOS and press the appropriate keys 'blind', you may well find the 512's fine, only the display wasn't working. This is always worth trying if you can before you resort to re-booting, if you get a blank screen on trying a new program.

MYSTERY SOLVED!

In Vol.11 Nos.5 and 6 I mentioned that DOS Plus doesn't like extensions to directory names. That's incorrect, but it's only thanks to the persistence of Philip Draper that I found out.

When I first used DOS Plus 2.1 I only had floppy discs and found that DOS Plus gave an error if directory names had an extension. That this appeared again when I tested the CHKDSK example for Vol.11 No.5 was, therefore, no surprise.

The fact that CHKDSK created the problem didn't bother me either. After all quite a number of the 512's modified DOS utilities have bugs or omissions. I viewed this as just another one. For obvious reasons the CHKDSK examples were tested on floppy disc and the problem of the directory extension was, to me, quite normal. It wasn't to Philip Draper though. He wrote saying he had no such problems and had had such a directory name on his winchester.

I checked and again verified that my version of DOS plus didn't like directory name extensions. However, since I'd never tried it on the hard disc I did so after Philip's letter. Lo and behold, it worked! I again tried on floppy with the same result as before! This wasn't reasonable. I re-booted and re-checked a couple of times - everything was completely consistent, but far from satisfactory.

It next occurred to me that all my installed copies of DOS Plus, my emergency boot floppy, the winchester and all backups have always come from the same source - the copy of issue disc 1 taken when I first acquired DOS Plus 2.1. I therefore took a new copy from my issue disc, re-installed it and tried again. No, not yet! It was exactly the same as before. Hmmm....

Not really expecting much luck, but short of other constructive ideas, I took another copy of issue disc 1, but this time from Mike Ginns' master disc, and tried again. Finally, the floppy directory name extension problem vanished.

A double check confirmed all. Booting from my own issue disc brought the problem back, booting from Mike's cured it. It seems my issue disc has always had a small corruption in the XIOS which caused the directory name problem on floppy discs. That it didn't happen on the hard disc is easy to understand, as most of the hard disc XIOS code is quite separate from that for floppies. That I didn't discover this before is also easy to understand, I tend not to do things that I know don't work!

Thanks to Philip I've now corrected a bug in my copy of DOS Plus that for years I'd accepted as 'standard'. It turns out it was nothing of the sort!

The Sideways Poet (Part 1)

David Houlton shows Master owners how to wax lyrical.

Are you old enough to remember that wonderful radio show called "Round The Horne"? I'm afraid I am. Kenneth Williams used to play this outrageous old folk-singer called Rambling Syd Rumpo, whose nonsensical parodies were always obviously obscene without ever being quite comprehensible. One of my favourites begins:

Ye nurbs and bogles o'bonnie Glen Postule, Oft have I greebled among your trees; Whirdled my lassie among your nettles, Sworn my devotion and stung both my knees.

I was listening to my Rambling Syd album one night, (drunk, as usual) when it occurred to me that - begging yer pardon, Frank Muir and Dennis Norden - my old Beeb could do stuff like that, suitably primed. Poetry from a Beeb ought to be feasible, and seemed reasonable in a world where paintings by Chimpanzees can fool art-critics. Just type in a lot of alternatives to each word of the original, and get the machine to create a new poem by choosing one word at random from each group.

Writing the routine in Basic wasn't hard: you just choose one word at random out of a line of DATA, RESTORE the next line and so on round and round. What was much harder was setting up the data so that all possible sequences made sense. If one line was "DATA I,we,you,she" and the next line went "DATA has,was,went", then I soon got combinations like "we was" or "you has". Making some kind of sense was surprisingly difficult, and rhymes proved a real problem. Anyway, I got three "poets" working more or less plausibly: Burns, Shakespeare and Wordsworth; then one night, even drunker than usual, I rewrote the thing as a sideways ROM, as a private penance to St. Cecilia.

```
AN OLD LOWLAND CHOTTER SONG

Oh fumblous glens o' black Glen Gropie,
Noo hae we lain here between these leaves;
Phurgled my bagpipes aboon thy peat bogs;
Aye, spoilit her scroatie wi' unco' heaps!


Aye, Doctor Finlay. There's mony a mickle macks a muckle.

Dump tae the wee printer, the noo? (Aye/Nae)
Or ye can quit, ye ken. (Q)
```

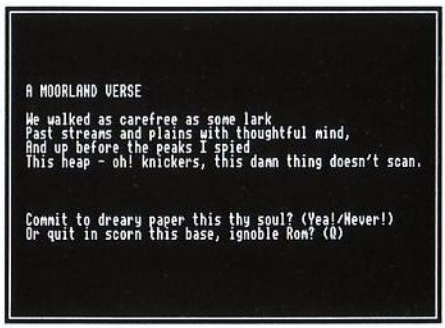
Burns poem

Now, in an attempt to persuade my wife that all those hours were not a complete waste of time, I am trying to entice you, gentle reader, to type the silly thing in and have a bit of fun with it this month. The serious bit comes next month, when I'll be guiding you through the not-too-difficult process of adding a "poet" or two of your own. The program is designed to make it easy for a non-programmer to add his own "poet" and/or extend the existing ones. The version given here is barely adequate in the number and length of the offerings, and could do with a lot more alternatives in some of the choice-groups, because I had to think of all the poor suckers typing all that data in.

I sincerely hope that teachers of language will consider using the Sideways Poet as an entertaining but valid exercise. Adding a "poet" would make a useful

Shakespeare

English/IT project for small groups of third or fourth year pupils, following a couple of classroom lessons using extracts from the writer concerned to try and pin down some aspects of literary style. My own subject is Modern Languages; despite the fact that the program cannot easily be made to cope with accents, it does have some value in practising verb-endings, adjectival agreements, noun-pronoun replacements and the like, though producing a single short sentence in French or German is quite a stiff task.



Wordsworth

Next month we'll go into the nuts and bolts of installing a new "poet". For now, type in the listing, saving frequently, and run it. Once the program has been run, the code is installed in a SRAM slot, and nothing short of power-off will get rid of it. Just do *PO to call it from anywhere. Wordsworth, thou shouldst be living at this hour!

```
   10 REM Program The Sideways Poet
   20 REM Version B1.0
   30 REM Author  David Holton
   40 REM BEEBUG  March 1993
   50 REM Program and data subject to co
pyright
   60 :
  100 PROCassemble
  110 FOR N%=7 TO 4 STEP-1
  120 IF N%?&2A1 NEXT:PRINT'"Sorry! No f
ree SRAM slot!":END
  130 OSCLI "SRWRITE "+STR$~Z%+" "+STR$~
(1+O%)+" 8000 "+STR$~N%
  140 N%?&2A1=&82
  150 PRINT''"Poet installed in SRAM slo
t ";N%
  160 N%=4:NEXT:END
  170 ::::::::::::::::
  180 DEF PROCassemble
  190 indexLo=&B0
  200 indexHi=indexLo+1
  210 ctrLo=indexLo+2
  220 ctrHi=indexLo+3
  230 pVecLo=indexLo+4
  240 pVecHi=indexLo+5
  250 osrdch=&FFE0
  260 osasci=&FFE3
  270 osbyte=&FFF4
  280 :
  290 FOR N%=4 TO 6 STEP 2
  300 Z%=?2+&100*?3+&200*ABS(N%=4)
  310 P%=&8000:O%=Z%
  320 [ OPT N%
  330 :
  340 BRK:BRK:BRK
  350 JMP canItBeForMe
  360 EQUB &82
  370 EQUB offset MOD &100
  380 EQUB &92
  390 .title
  400 EQUS "The Sideways Poet"
```

```
 410 .offset
 420 BRK:EQUS "(C) Beebug ?? 199?"
 430 :
 440 .help
 450 LDA (&F2),Y
 460 CMP #&0D:BNE nobodyLovesMe
 470 LDA &F4:ORA #&30:STA slotNum
 480 LDX #&FF
 490 .loop
 500 INX:LDA helpThingy,X
 510 JSR osasci:BNE loop
 520 :
 530 .nobodyLovesMe
 540 PLY:PLX:PLA:RTS
 550 :
 560 .canItBeForMe
 570 PHA:PHX:PHY
 580 CMP #9:BEQ help
 590 CMP #4:BNE nobodyLovesMe
 600 LDA (&F2),Y:AND #&DF
 610 CMP #ASC"P":BNE nobodyLovesMe
 620 INY:LDA (&F2),Y:AND #&DF
 630 CMP #ASC"O":BNE nobodyLovesMe
 640 INY:LDA (&F2),Y
 650 CMP #&0D:BNE nobodyLovesMe
 660 :
 670 .arsMagna
 680 JSR setUpRnd
 690 LDX #&FF
 700 .loop
 710 INX:LDA menu,X
 720 JSR osasci:CMP #7:BNE loop
 730 :
 740 .loop
 750 JSR inKey
 760 CMP #ASC"S":BEQ Shake
 770 CMP #ASC"R":BEQ Report
 780 CMP #ASC"W":BEQ WillyWW
 790 CMP #ASC"B":BNE loop
 800 :
 810 .Burns
 820 LDX #BurnsData MOD &100
 830 LDY #BurnsData DIV &100:JMP offWeG
o
 840 :
 850 .Report
 860 LDX #ReportData MOD &100
 870 LDY #ReportData DIV &100:JMP offWe
Go
 880 :
 890 .Shake
 900 LDX #ShakeData MOD &100
 910 LDY #ShakeData DIV &100:JMP offWeG
o
 920 :
 930 .WillyWW
 940 LDX #WillyWWdata MOD &100
 950 LDY #WillyWWdata DIV &100
 960 :
 970 .offWeGo
 980 LDA #prntTxt DIV &100:STA pVecHi
 990 LDA #prntTxt MOD &100:STA pVecLo
1000 LDA #&0C:JSR osasci
1010 :
1020 .nextGroup
1030 JSR random
1040 .startOfGrpCoords
1050 STY indexHi:STX indexLo
1060 :
1070 .countDownOneWord
1080 DEC counter:BMI atLast
1090 :
1100 .letterByLetter
1110 JSR nextChar
1120 CMP #ASC":":BEQ countDownOneWord
1130 CMP #ASC"#":BEQ startOfGrpCoords
1140 CMP #ASC"*":BNE letterByLetter
1150 JMP startOfGrpCoords
1160 :
1170 .atLast
1180 LDA #ASC" ":JSR go
1190 :
1200 .prntWrdOnScrn
1210 JSR nextChar
1220 CMP #ASC"*":BEQ doYouWantAnyMore
1230 CMP #ASC"#":BEQ hashExit
1240 JSR screenAndStore
1250 CMP #ASC":":BNE prntWrdOnScrn
1260 :
1270 .scrapTheRest
1280 JSR nextChar
1290 CMP #ASC"*":BEQ doYouWantAnyMore
```

```
1300 CMP #ASC"#":BNE scrapTheRest
1310 :
1320 .hashExit
1330 JSR nextChar
1340 LDX indexLo:LDY indexHi
1350 JMP nextGroup
1360 :
1370 .nextChar
1380 INC indexLo:BNE indHiOk
1390 INC indexHi
1400 .indHiOk
1410 LDA (indexLo):RTS
1420 :
1430 .inKey
1440 LDX #1:LDY #0:LDA #&0F:JSR osbyte
1450 JSR osrdch:BCS outAndClaim
1460 AND #&DF:RTS
1470 .outAndClaim
1480 PLY:PLX:PLA:LDA #0:RTS
1490 :
1500 .doYouWantAnyMore
1510 JSR inKey
1520 CMP #ASC"A":BEQ printOut
1530 CMP #ASC"Y":BEQ printOut
1540 CMP #ASC"Q":BEQ outAndClaim
1550 CMP #ASC"N":BNE doYouWantAnyMore
1560 JMP arsMagna
1570 :
1580 .printOut
1590 LDA #prntTxt DIV &100:STA pVecHi
1600 LDA #prntTxt MOD &100:STA pVecLo
1610 LDA #2:JSR osasci
1620 LDA #&15:JSR osasci
1630 .loop
1640 LDA (pVecLo):JSR osasci
1650 INC pVecLo:BNE pvhOK
1660 INC pVecHi
1670 .pvhOK
1680 CMP #7:BNE loop
1690 LDA #3:JSR osasci
1700 LDA #6:JSR osasci
1710 JMP arsMagna
1720 :
1730 .screenAndStore
1740 CMP #ASC":":BEQ ret
1750 CMP #ASC"+":BNE notPlus
1760 LDA #&0D
1770 .notPlus
1780 CMP #ASC">":BNE not7
1790 LDA #7
1800 .not7
1810 CMP #ASC"<":BNE go
1820 LDA #8
1830 .go
1840 JSR osasci
1850 STA (pVecLo):INC pVecLo:BNE ret
1860 INC pVecHi
1870 .ret
1880 RTS
1890 :
1900 .setUpRnd
1910 LDA &FE58:AND #3
1920 ADC #&E0:STA ctrHi
1930 LDA &FE78:STA ctrLo
1940 .random
1950 LDA (ctrLo):EOR &FE58
1960 AND #&0F:STA counter
1970 INC ctrLo:BNE return
1980 INC ctrHi:LDA ctrHi
1990 CMP #&FC:BEQ setUpRnd
2000 .return
2010 RTS
2020 ::::::::::::
2030 .helpThingy
2040 EQUW &0D0D:EQUS "THE SIDEWAYS POET
is alive and well and"
2050 EQUB &0D:EQUS "living in slot "
2060 .slotNum
2070 EQUS "?.  One day a beautiful"
2080 EQUB &0D:EQUS "princess will come
along and type *PO"
2090 EQUW &0D0D:BRK
2100 :
2110 .menu
2120 EQUW &8016:EQUW &11
2130 EQUW &8711:EQUB &0C
2140 EQUW &0117:EQUD 0:EQUD 0
2150 EQUW &0D0D
2160 EQUS"    THE SIDEWAYS POET"
2170 EQUB &0D
2180 EQUS"  Burns          (B)"
2190 EQUB &0D
```

```
 2200 EQUS"  Shakespeare    (S)"
 2210 EQUB &0D
 2220 EQUS"  Wordsworth     (W)"
 2230 EQUW &0D0D
 2240 EQUS" Non-cultural offerings:"
 2250 EQUW &0D0D
 2260 EQUS"  School Report (R)":EQUB 7
 2270 :
 2280 .BurnsData
 2290 EQUS ":++#"
 2300 EQUS ":AN OLD:A SILLY:A DIRTY AULD
:A GROTTY WEE#"
 2310 EQUS ":SCOTTISH:HIELAND:LOWLAND:HE
BRIDEAN:McQUERGLE:MacGILLICUDDIE:ROBBY B
URNS#"
 2320 EQUS ":LAMENT:DIRGE:CHOTTER SONG:H
OWKING CHANT:PIBROCH:ROUNDELAY:FOLK BALL
AD:ODE TAE A GERBIL#"
 2330 EQUS ":++#"
 2340 EQUS ":Oh,:Oh:Ach,:Ye:My:Och,:Aye#
"
 2350 EQUS ":bonny:sleekit:wee bit:grump
it:bunglie:fumblous:gurly#"
 2360 EQUS ":Braes:plains:dells:woods:gl
ens:lochs#"
 2370 EQUS ":o':of:by:nigh#"
 2380 EQUS ":Braw:auld:fair:foul:black#"
 2390 EQUS ":Glen:Loch:Ben:Mac:Aber:Brae
#"
 2400 EQUS ":Wibblie,:Chaunter,:Gropie,:
Screepit,:Tattie,:Frumpie,:Snodule,:Scro
aggle,:Greeblie,#"
 2410 EQUS ":+#"
 2420 EQUS ":Oft:Noo:Weel:Sair:Lang#"
 2430 EQUS ":hae:have#"
 2440 EQUS ":I:we#"
 2450 EQUS ":phurgl'd:scropled:greebled:
spleekit:lain here:sat there:winded:fumb
led#"
 2460 EQUS ":amang:beside:betwixt:betwee
n:ahint:aboon#"
 2470 EQUS ":your:thy:the:these#"
 2480 EQUS ":leaves:sheep:peaks:sleet#"
 2490 EQUS ":<;:<,:-#"
 2500 EQUS ":+#"
 2510 EQUS ":Phurgled:Wheebled:Scruppit:
Dongled:Handled:Scraped off#"
 2520 EQUS ":our:my:thy:her:his:its:the#
"
 2530 EQUS ":porridge:sporran:bagpipes:S
kean Dhu:haggis:whisky:tatties:chanter:d
rone-pipes#"
 2540 EQUS ":aboon:amang:ahint:before#"
 2550 EQUS ":thy:your#"
 2560 EQUS ":podules;:corries;:peat bogs
;:Bracken,:crofties,:thistles;:heather,:
torrents,#"
 2570 EQUS ":+#"
 2580 EQUS ":And:Weel:Aye,#"
 2590 EQUS ":frumpit:boggled:bloated:spo
ilit:ruined:tattied:soiled#"
 2600 EQUS ":my:our:her#"
 2610 EQUS ":claymore:sporran:bonnet:poc
kets:scroatie:byre:loamings:kilt-bag#"
 2620 EQUS ":with:wi':in:by#"
 2630 EQUS ":mony:unco':muckle:mickle:ha
iry#"
 2640 EQUS ":neaps:breeks:feet:peat:heap
s#"
 2650 EQUS ":<!:<.#"
 2660 EQUS ":++++>#"
 2670 EQUS "<<:Aye, Doctor Finlay.#"
 2680 EQUS "<<:There's mony a mickle mac
ks a muckle.+#"
 2690 EQUS "<<:It's a Braw, bricht, moon
licht nicht.+#"
 2700 EQUS ":++ Dump tae the wee printer
, the noo? (Aye/Nae)+ Or ye can quit, ye
 ken. (Q)++*"
 2710 :
 2720 .ShakeData
 2730 EQUS ":++#"
 2740 EQUS ":SHAKESPEARE#"
 2750 EQUS ":++#"
 2760 EQUS ":Now:Here:Today:Soon#"
 2770 EQUS ":is#"
 2780 EQUS ":our:the:this#"
 2790 EQUS ":winter:summer:summit:mornin
g#"
 2800 EQUS ":of#"
 2810 EQUS ":my:our:thy#"
 2820 EQUS ":discontent:countenance:turp
```

```
itude:royal throne:noble sire#"
 2830 EQUS ":+#"
 2840 EQUS ":Made:Turned:Left:Shewn:Foun
d#"
 2850 EQUS ":Nobly:fairly:greatly:long a
nd:fitly#"
 2860 EQUS ":glorious:splendrous:wondrou
s:radiant:beauteous#"
 2870 EQUS ":by this:with this:by our:by
 my#"
 2880 EQUS ":son:sun:child:Lord:King:Pri
nce:Queen:Earl#"
 2890 EQUS ":of:from#"
 2900 EQUS ":Wales:France:Slough:Glamis:
York:Golders Green:Spain:Dollis Hill:The
bes:Welwyn Garden City#"
 2910 EQUS ":<!:<.#"
 2920 EQUS ":++++>#"
 2930 EQUS ":Wouldst thou:Wilt thou:Woul
d ye#"
 2940 EQUS ":now dump:dump all:resort#"
 2950 EQUS ":to th'printer,:to paper,:to
 printer,:to printing,#"
 2960 EQUS ":Noble sir:oh, fair coz:Nobl
e coz:Majesty#"
 2970 EQUS ": :<?:<,#"
 2980 EQUS ":(Y/N)#:+#"
 2990 EQUS ":Or:Else#"
 3000 EQUS ":wouldst thou:wilt thou:will
 you#"
 3010 EQUS ":sooner:rather:better:wisely
:straightly:forthwith:quickly#"
 3020 EQUS ":quit:leave:flee:fly#"
 3030 EQUS ":the:this#"
 3040 EQUS ":odious:hideous:ignoble:infa
mous:evil:wicked#"
 3050 EQUS ":Rom?  (Q)++*"
 3060 :
 3070 .WillyWWdata
 3080 EQUS ":++#"
 3090 EQUS ":A#"
 3100 EQUS ":LAKELAND:MOORLAND:SYLVAN:MO
UNTAIN#"
 3110 EQUS ":IDYLL:ODE:POEM:QUATRAIN:VER
SE#"
 3120 EQUS ":++#"
 3130 EQUS ":I:We:She:He#"
 3140 EQUS ":wandered:floated:meandered:
roamed:walked as:went as:flew as#"
 3150 EQUS ":lonely:carefree:careless:sp
rightly:freely:lightly:idly#"
 3160 EQUS ":as#<<:if#"
 3170 EQUS ":a:some#"
 3180 EQUS ":cloud:bird:leaf:star:sprite
:lark:clod - sorry, cloud#"
 3190 EQUS "<<:<,#"
 3200 EQUS ":+#"
 3210 EQUS ":Oe'r:By:Twixt:Past:'Mongst#
"
 3220 EQUS ":hills:dales:trees:woods:str
eams#"
 3230 EQUS ":and#"
 3240 EQUS ":paths:lakes:plains:moors:bo
gs#"
 3250 EQUS ":with#"
 3260 EQUS ":pensive:sombre:thoughtful:l
onely:inward#"
 3270 EQUS ":mind,:heart,:soul,:eye,:moo
d,#"
 3280 EQUS ":+#"
 3290 EQUS ":And:Then:Lo!#"
 3300 EQUS ":there:here:up:down#"
 3310 EQUS ":beneath:between:among:behin
d:before#"
 3320 EQUS ":the#"
 3330 EQUS ":trees:lakes:woods:flowers:d
ales:peaks:vales:tarns:dells:used condom
s#"
 3340 EQUS ":I:she:we:they:he#"
 3350 EQUS ":spied:saw:found:beheld:felt
#"
 3360 EQUS ":+#"
 3370 EQUS ":A:This:The#"
 3380 EQUS ":host of:crowd:pile:heap:thr
ong:blaze:riot:glory#"
 3390 EQUS ":-:<...#"
 3400 EQUS "<<:oh,:oh...:oh!#"
 3410 EQUS "<<:knickers,:blow,:curse -:h
eck!:damn!:blast,:drat,#"
 3420 EQUS ":this#"
 3430 EQUS ":thing:damn thing:one:blaste
d thing:line:bit#"
```

```
 3440 EQUS ":won't:doesn't:isn't going t
o#"
 3450 EQUS ":rhyme:turn out right:end pr
operly:scan#"
 3460 EQUS "<<:after all#"
 3470 EQUS ":<.:<!#"
 3480 EQUS ":++++>#"
 3490 EQUS ":Commit to dreary paper this
 thy soul? (Yea!/Never!)+ Or quit in sco
rn this base, ignoble Rom? (Q)++*"
 3500 :
 3510 .ReportData
 3520 EQUS ":++#"
 3530 EQUS ":FRENCH:MATHS:HISTORY:LATIN#
"
 3540 EQUS ":+:REPORT+#"
 3550 EQUS ":Jason:Darren:Lee:Warren:Gar
y:Wayne:Edmund:Justin:Miles:Charles#"
 3560 EQUS ":is among the:is one of the:
has to be among the:surely ranks as one
of the:is certainly one of the#"
 3570 EQUS ":+#"
 3580 EQUS ":biggest:worst:thickest:dozi
est:most boneheaded#"
 3590 EQUS ":pratts:cretins:drongoes:lun
atics:fudgeheads:mumblebrains#"
 3600 EQUS ":+#"
 3610 EQUS ":I have:we have:this school
has:any school has#"
 3620 EQUS ":yet:ever#"
 3630 EQUS ":been lumbered with:known:ha
d the misfortune to know:been unlucky en
ough to encounter:been saddled with#"
 3640 EQUS ":<!:<.#"
 3650 EQUS ":++#"
 3660 EQUS ":This kid:The child:The loat
hesome creature:This abject apology for
a human being:He:The repulsive brat:This
 malodorous creature#"
 3670 EQUS ":is#"
 3680 EQUS ":so lazy:so idle:indolent to
 such a degree:so dozy:so thick:such a b
asket-case#"
 3690 EQUS "<<:that#"
 3700 EQUS ":+#"
 3710 EQUS ":he wouldn't#"
 3720 EQUS ":remove a twenty-pound note
if you stuffed one up his nostril:run th
ree feet to avoid a falling asteroid:rol
l over to get off a wasps' nest"
 3730 EQUS ":lift a baby out of a food-m
ixer:blow his nose if there was a wasp u
p it"
 3740 EQUS ":warn his blind granny of a
hole in the floor:eat if you didn't prop
 his mouth open#"
 3750 EQUS ":<.++++>#"
 3760 EQUS ":Well? Don't just#"
 3770 EQUS ":sit there:stand there:stand
 around:mooch about#"
 3780 EQUS ":gawping:picking your nose:p
laying pocket billiards:scratching yours
elf#"
 3790 EQUS ":<!:<.#"
 3800 EQUS ":+#"
 3810 EQUS ":Do you want#"
 3820 EQUS ":a printout done:me to put i
t in writing:me to print it out#"
 3830 EQUS "<<:for you#"
 3840 EQUS "<<:or not:or don't you#"
 3850 EQUS ":<? (Y/N)+#"
 3860 EQUS ":Or do you want#"
 3870 EQUS ":to be expelled:the boot:to
be kicked out:me to expel you#"
 3880 EQUS ":<? (Q)*"
 3890 :
 3900 .counter
 3910 BRK
 3920 .prntTxt
 3930 ] NEXT:ENDPROC
 3940 Bufo scripsit 31 vii MCMXCII
```

..Points Arising..Points Arising..

RECURSIVE COMPACTION ON THE MASTER
(Hints and Tips Vol.11 No.6)

Line 200 of the listing given was unfortunately garbled when printed. The line should read:

```
200 PRINT "Compaction completed."
```

Thanks to J.H.Sephton for pointing this out.

Input (2)

Alan Wrigley continues his description of input.

Last month I described how to use the INPUT statement to process text strings or values entered at the keyboard in response to a prompt from the program. This is a very common requirement for many types of program, which allows the user to determine the data to be processed while the program is running. However, there are probably just as many situations where the kind of user input required is of a different type, and involves merely pressing a single key, either in response to a prompt or as part of a sequence of actions (e.g. a game). Using INPUT to enter a character and press Return would at best be a case of using a sledgehammer to crack a nut, and at worst would be completely useless - aliens are not going to hold their fire while you press Return!

For these situations, a range of keywords is provided in Basic which detect single keypresses and act on them straight away. These are GET, GET$, INKEY and INKEY$. As you might imagine from the names, these fall into two groups, and the difference between the groups is simple: GET and GET$ suspend operation of the program until a key is pressed, while INKEY and INKEY$ pause the program only for a specified time while waiting for a key to be pressed. This month's article will concentrate on the use of GET and GET$, while the INKEY functions will be described next month.

GETTING THE MESSAGE

GET and GET$ are both functions which cause the program to wait until the next key is pressed, whereupon they return the value of the key (incidentally, we call them functions rather than statements because their primary purpose is to return a value rather than to execute a command). The sole difference between GET and GET$ is that the former returns the ASCII code of the key, whereas the latter returns its character as a string. If you are using GET, therefore, you must assign the returned value to a numeric variable:

```
A%=GET
```

while for GET$ it must be a string variable:

```
A$=GET$
```

It is important to note that GET and GET$ do *not* display the character typed. If you want it to appear on the screen, you must put it there yourself after the value has been returned. We will come back to this in a moment.

You may be wondering why two functions are needed. After all, Basic can quite easily convert between ASCII codes and character strings and vice versa. Having two functions is just another example of the way in which BBC Basic has been designed with the user in mind. There are some situations where ASCII values are more convenient, and some where strings would be better. For example, suppose your program displays a menu on the screen, with items numbered 1 to 9. The user must select an item by pressing the appropriate digit on the keyboard. With only GET$ available, you would need either to convert the string to an ASCII value, or to use a cumbersome structure of the form:

```
A$=GET$:IF A$="1" THEN... ELSE IF A$="2"
THEN...
```

and so on.

With GET it is so much easier, allowing you to use ON...GOTO (or for Master owners the infinitely preferable ON...PROC). It will also be faster, since processing strings, or converting strings to ASCII values, takes up valuable time. Bearing in mind that "1" is ASCII code 49, "2" is 50 and so on, any correct menu selection returned by GET in the example quoted here will lie between 49 and 57, so you can process the input very neatly with:

```
A%=GET
ON A%-48 PROCone,PROCtwo,PROCthree,PROCf
our,PROCfive,PROCsix,PROCseven,PROCeight,
PROCnine ELSE PROCdonothing
```

Subtracting 48 from the value returned, as we have done here, ensures that the menu selections will lie between 1 and 9, allowing us to use an ON...PROC structure. The ELSE statement at the end is very important because it ensures that any keypresses which do not lie between 49 and 57 can be trapped. All that *PROCdonothing* needs to do is to return immediately:

```
DEF PROCdonothing
ENDPROC
```

The effect of this will be that pressing a non-digit key simply does nothing.

GET is also more useful in a situation such as we described last month. If you remember, we supplied a listing which provided a flexible input routine to get around some of the disadvantages of using INPUT to receive data from the user. GET was used in that listing to take each key typed in turn and process it according to its value. If you refer back to the listing, you will see that GET appears in line 1030, and the resulting value is processed by *PROCchar* (unless it is a carriage return or a disallowed character).

GET is clearly preferable to GET$ in this situation, since the keypresses you should expect will be a mixture of standard characters and control codes (Return, Delete and in more complex routines, cursor keys). Lines such as:

```
IF A$=CHR$(137) OR A$=CHR$(138) THEN...
```

are rather unwieldy, and also take longer for Basic to interpret and execute than the equivalent:

```
IF A%=137 OR A%=138 THEN...
```

It is often important, as it is here, for the user to be able to see what has been typed in, and as we mentioned earlier, this is not done automatically when you use GET or GET$. In the case of GET, it is quite simple to echo the character typed on the screen by using VDU. For example:

```
A%=GET:VDU A%
```

will display each character as it is typed. This will even work correctly with the Delete key - Delete returns a value of 127, and VDU 127 will move the cursor back and erase the previous character. You can see the process at work in last month's listing (lines 1090 and 1100).

Before we leave GET, it is worth noting that in some circumstances there is no need to assign it to a variable. For example, if the user merely has to press the space bar to continue, there is usually no need to keep a copy of the key pressed, and so the following line is all that is required:

```
REPEAT UNTIL GET=32
```

since 32 is the ASCII code for Space. However, there is a trap here for the unwary. Suppose that you want to detect either Space or Escape pressed; you might think that the following will do the trick:

```
REPEAT UNTIL GET=32 OR GET=27
```

The problem with this is that, since the action of the GET keyword is to call a

function, this line will cause the function to be called twice, and the loop will not terminate until *two* keypresses have been made. In this case, you must use the assigned variable method:

```
REPEAT A%=GET:UNTIL A%=32 OR A%=27
```

GET$

When GET is so flexible to use, why do we need an alternative in GET$? Suppose the user is faced with a prompt such as "Do you want to continue? Press Y or N". The ASCII codes for "Y" and "N" are not consecutive, and so you cannot sensibly use ON...GOTO or ON...PROC. Not only that, but if you are providing a prompt of this nature you really should cater for lower-case keys pressed too, which doubles the number of codes to be detected. There is nothing more annoying and frustrating for a user than to find that the program apparently refuses to acknowledge the key pressed - all because Caps Lock is off. So if GET were the only function available the code might look like this:

```
REPEAT A%=GET
UNTIL A%=ASC("Y") OR A%=ASC("y") OR A
%= ASC("N") OR A%=ASC("n")
```

and you still have some processing to do to work out whether Y or N was pressed.

In fact, you can improve on this considerably by taking advantage of the fact that any alphabetic character, upper or lower case, can be ORed with 32 to give its lower-case ASCII value. Since "n" is ASCII 110 and "y" is ASCII 121, you can condense the above lines to:

```
REPEAT A%=GET OR 32
UNTIL A%=121 OR A%=110
```

Now A% is 121 for Yes, 110 for No.

With GET$, however, there is a much better way of doing this. First of all, let me show you how *not* to do it. You have probably all seen programs with lines such as:

```
100 A$=GET$:IFA$<>"Y" AND A$<>"y" AND A
$ <>"N" AND A$<>"n" GOTO100
```

This is ugly, verbose and unstructured, with the dreaded GOTO thrown in for good measure. There is in fact a very much neater way to achieve the required result, as follows:

```
REPEAT A%=INSTR("YyNn",GET$):UNTIL A%
```

Here, GET$ is called once each time round the loop, and its returned character is checked by the INSTR statement to see if it appears within the control string "YyNn". If the key pressed is not one of these, the value of A% will be zero and the loop will repeat, so the program will not continue until a correct key has been pressed. Not only this, but since A% reflects the position of the returned character within the control string, it will be 1 or 2 for Yes, 3 or 4 for No. This means that you can process the input with an ON...PROC very simply as follows:

```
ON (A%+1) DIV 2 PROCyes,PROCno
```

For more complex choices involving perhaps a number of keys scattered about the ASCII spectrum, this approach makes even more sense. For example, suppose you offer the user a choice of pressing F for forward, B for back, M for menu or Q for quit; this can be achieved quite elegantly with:

```
REPEAT A%=INSTR("FfBbMmQq",GET$)
UNTIL A%
ON (A%+1) DIV 2 PROCforward,PROCback,PR
OCmenu,PROCquit
```

In this case you do not even need an ELSE at the end of the ON...PROC, since only values of 1 to 4 will ever be passed to the third line.

Next month we will conclude our look at input by considering the INKEY functions.

Public Domain Software

Alan Blundell brings together an assorted collection of news on the PD scene this month.

As promised in the last issue, I'm going to look at one or two items of PD software which I have somehow not given the attention they deserve in a previous column. Before I do that, though, there are several developments which I thought I could slot in as they are likely to be of interest to a number of readers.

EX-COMMERCIAL SOFTWARE

The first of these follows on neatly from last month, where I discussed my ideas about commercial software which is no longer actively marketed. I recently heard from Joyce and Derek Haslam with some very good news. As you may remember from a recent column, Derek wrote Acornsoft's 'Gateway to Karos' adventure game, and generously released the sequel, 'Mirror of Khoronz' as public domain. Well, the latest news is that Joyce has secured permission from Acorn to re-release the original 'Gateway to Karos' as PD! This is an important development, as Acornsoft was in no way a small software house and the fact that one of their titles is now PD is a step along the road to persuading more people. It is also an excellent game, and its re-release will enable a wider audience to appreciate just how entertaining a good adventure game can be.

Naturally, I immediately wrote to Acorn to press my case for re-releasing more of their old products but unfortunately this isn't going to happen overnight. This isn't a case of Acorn being unhelpful, though: there are all sorts of considerations in re-releasing software, such as the views of the author (Joyce had an advantage there!) and arrangements made with other software houses (many Acornsoft games passed to Superior Software, for example). Still, one step at a time can take us a long way...

MAGAZINE DISCS

Good news for some will be that the software from the first five volumes of BEEBUG is now public domain. As the discs from volumes 3 to 5 (volumes 1 and 2 were tape only) are no longer available from the back issues department, if you missed these first time round, then you had missed them, full stop, until now. The early volumes of BEEBUG contained some excellent software, and I am very pleased that it is now once again available. If space permits, I may briefly remind you of some of the highlights in a future column.

ANTIPODEAN SOFTWARE

The BBC Micro has users in all parts of the world, and I regularly receive airmail correspondence. Airmail can be quite speedy, but it isn't the cheapest way to receive PD software. So, I am pleased to be able to mention Southern Beeb PD, an Australian PD library dedicated to the BBC Micro and Master series. Run by Jock Smylie, the library caters for users in Australia and New Zealand (only). The address is:

Southern Beeb PD
P O BOX 409
Kingswood, 5062
South Australia

A stamped addressed envelope (or an envelope with an international reply coupon) will bring you a free printed summary catalogue; unfortunately I have no information on SBPD charges or available disc formats. The software in the SBPD catalogue is taken from UK libraries at present, but Jock hopes to be able to swap new Australian software with me in the near future!

BACK TO NOW

Apologies for the 'bittiness' of this column so far; I felt that these news items would be of wide enough interest to justify including them. For the rest of the column this issue, I will look at some

of the software I meant to mention before. In keeping with the style of the column so far, I want to mention three pieces of software, all of which are well worth noting, but which have absolutely nothing else in common.

The first item is the Harston ADFS ('HADFS'). This is an ADFS-like filing system for single density DFS users. Whilst the HADFS does not offer compatibility with normal ADFS, in that you can't use HADFS to read or write to an ADFS disc or vice versa, it is compatible in that it offers a true hierarchical file structure. Under HADFS, a disc is not limited to 31 files per surface, and nested subdirectories can be created. Files and directories can have names up to 10 characters in length. The disadvantages of HADFS lay in the fact that it is non-standard and in the incomplete state of some system utilities, but for your own use on a non-ADFS machine, it is a useful and well-documented program.

HADFS was written by the eponymous (I've always wanted to use that word!) Jonathan Harston, and an early version has been available for some time. However, I recently heard from Jonathan, who is now working in Hong Kong. He sent me the latest version of HADFS (version 0.45), which is closer to completion. Jonathan is obviously committed to making a professional job of HADFS and this shows in the quality of the ROM image, the many utility programs and the detailed documentation. Even if you don't plan to use HADFS, if you are of a technical bent, it is worth looking at HADFS just for the pleasure of seeing a job well done.

From recent correspondence, I know that a number of readers have an interest in genealogy, so it's about time I mentioned a program which came to my attention in the middle of 1992. 'Freeware Family History', written by James Farmer, uses the 'Aristotle' Wimp menuing environment, which was written by Andy Nibbs. Because of this, it only works on a BBC Master. It requires banks 4 & 6 of the Sideways RAM to be empty and enabled, and will only operate under ADFS. If your system can provide all of these, then 'Freeware Family History' is well worth a look. It uses windows and a pointer to create an easy to use system which looks good and provides all of the basic facilities needed to record a family history.

The program allows the recording of dates of birth, marriage and death, links to records for parents and details of up to 15 children, which should be enough for most families! Each record may also have free text information associated with it, which may be viewed in a window or edited. Navigation of the family tree is simple - 'clicking' on a husband or wife moves to a screen detailing their parents and family, and 'clicking' on a child moves to a screen detailing any marriage and/or grandchildren. Multiple marriages and inter-marriages can be catered for using the 'Make Link' icon which appears in a line of utility icons at the base of the screen.

Finally, I will briefly mention a disc of utilities and other software for radio amateurs which was recently sent to me by Glyn Fowler. The disc was compiled by Richard Sterry, G4BLT, of Wakefield BBC Micro User Group. Richard kindly gave permission for the disc to be distributed as public domain. Although I know that a number of readers are radio amateurs, I am not, so perhaps it would be better if I refrained from passing too much comment myself. All I know is that I seem to hear a lot from people interested in radio software and that this software has been described to me as being of 'near professional quality'. Subjects include CW transceiver, Locater & contest coring, TNC driver, Morse tutor, RTTY transceiver and UU encoder/decoder.

I hope that makes more sense to you than it does to me! If so, perhaps you should take a look at this disc. If not, perhaps your favourite hobby will feature in next issue's round up of recent PD releases.

Note: BEEBUG programs from Volumes 1-5 will be available only from BBC PD, not from RISC Developments.

Machine Code Corner

This month the voice from the pond talks profoundly on the subject of paging.

Hi there, Toad fans. I recently had a very nice letter from Silas Brown of Bridport, who submitted two rather late entries for the assembler-limerick competition. Alas, one was all "\" comments - which is cheating - and was anyway obscene. Pity, it was fun. The other was weaker and also contained various devious cheats, so tough bananas, Silas, it doesn't get printed. I'll send you an 'I'M A SWOT' badge anyway, for a darn good try.

Silas also sent in his views on the pronunciation of assembler mnemonics. Remember? Like me, he says "string" for '$', not the meaningless "dollar" which is now the norm. Good man. Silas also gave me one or two other worthwhile ideas and then asked some very interesting questions: Do toads really have 'tadpoles'? He thought it was only frogs. How do you page ROMs, how can you read ADVAL and -INKEY from machine code, and how do you access Basic commands from machine-code?

Sorry about the first one, Silas - the 'tad-' part is the same word as 'toad'. I'll detail the reasons for the modern difference in the vowels if anyone cares to buy me a pint.

The answers to the other questions are all linked together; it comes down in the end to the Beeb's memory-map. Old hands must forgive me for getting a bit elementary at this point; what I'm going to do is take the question about paging ROMs right back to the hardware level to see why it's needed, then later we can talk about Basic and the low-level MOS routines which it often uses.

Basic is just another ROM in the memory-map. If you have a Master, do *ROMS now. View, Edit, all that stuff on the Master, all the EPROMS you buy and put in cartridges, all Mr T's ROM images, all are equal in the eyes of the 6502 CPU. The code in them all begins at &8000, the highest address in each is &BFFF. Only one at a time can be in use, although they can be swapped around pretty fast. The 'switching' in and out isn't mechanical, of course, it's done by the Memory Controller chip.

Why such a complicated way of going about things? Why not use consecutive addresses and map them all in? Simply because there is far more memory, ROM and RAM, in the Beeb than the 6502 can handle on its own. Eight-bit processors like the 6502 or Z80 can only cope with &10000 (65,536) memory addresses. From the memory point of view they are actually sixteen-bit chips - the address bus is sixteen bits wide, i.e. there are sixteen little silvery tracks on the circuit board, each connected to one of the sixteen address pins of the CPU and to the RAM and ROM chips. It's the data bus which is only eight bits wide: the biggest number that will fit on it is &FF. Now sixteen ones in a row is binary for &FFFF, or 65,535, therefore the highest address the 6502 can talk to is &FFFF. Since the lowest address is 0, the total number of addresses it can cope with is &FFFF+1, which is &10000, or 65,536. To put it yet another way, 64K.

ON THE BUSSES

Going back to buses, there is also a control bus: one little silvery track for *read*, one for *write* and a couple more for other purposes. Most machine-code instructions involve a read (data passes from memory to the 6502) or write (data from 6502 to memory).

Here's an example of a read operation: LDA &A085 - load the accumulator A with the byte currently stored in memory location &A085. The CPU switches the voltages on the 16 address lines: each goes either 'high' or 'low', so that if you could see those voltages as colours, they would make a pattern like the binary number 1010000010000101, which is &A085. It switches the read line high and the write line low. Receiving this signal, the memory chip which contains location &A085 fetches the contents of that address and puts the binary pattern of that number, as voltages, onto the eight tracks of the data bus. The 6502 reads the pattern, stores it in A and moves on to deal with the next instruction, which is likely to be to do something with the number just read, unless the programmer is two transistors short of a flip-flop.

A write operation is similar. Using STA &DC03, the 6502 puts &DC03 on the address bus, that's 1101110000000011. It copies whatever number is in A onto the data bus and sets the write line high and the read line low. The chip containing location &DC03 takes the number from the data bus and stores it in &DC03. What if &DC03 happens to be in a ROM chip, and can't be written to? Tough bananas, programmer. The ROM chip couldn't obey the instruction, there's a bug in your software.

I hope it's now clear what limitations the 'architecture' of an 8/16 bit chip imposes on the memory-map of the computer - the old Spectrum, for example, had one 16K ROM and three 16K Ram chips, and that was that. But the Speccy, though great fun, was very limited. The Beeb has far more facilities, and there's 128K of RAM in the Master, not to mention ROM. The way it's done is to use a memory-control device to intercept the address-bus and switch ROMs and RAM chips in and out, much as trains are switched by points from one line to another. But at any given moment, only &10000 locations, 64K, are switched in. A major part of this switching system is the set of 'sideways' areas from &8000 to &BFFF; there are 11 chips and 5 empty sockets, all of 16K; we call them ROM slots, although four are in RAM, and only one at a time is active. In fact, that's an oversimplification, but the principle is right.

How are they 'paged' in and out? Obviously, your code must direct the memory controller chip. How? There is nothing in the instruction-set to enable you to talk to any chip but the 6502. Well, in the Beeb there are &100 addresses - that's one 'page' - on the address bus which don't connect with ROM or RAM chips, but with all sorts of odd bits, among them the memory controller. One of the 'latches' on that chip is wired in at &FE30, so that if your code instructs the 6502 to write to &FE30, the 6502 goes through the process just described, but it's the memory controller that gets the data, rather than a memory chip. All you do is write the number of the ROM to &FE30. LDA #&0C:STA &FE30 selects ROM &0C, which is Basic. Make it LDA #&0E and it's View. A cunning plan worthy of Baldrick himself.

The ROM select address &FE30 is so important that Acorn gave it a mnemonic, ROMSEL. I wonder why 'ROMSEL'? P'raps it's something to do with ROMans and SELling out, since Acorn is now owned by Olivetti. We shall never know. Another one is &FE34, mnemonic ACCON, which CONtrols ACCess to shadow RAM and 'private' RAM. Other addresses in page &FE (mnemonic SHIELA, honestly!) connect with input/output ports, timers, the tube and so on.

One of them, &FE60, is the user port; that's where the joystick goes. Now I think Silas's question about ADVAL may have meant 'How do I read the joystick?' The answer is, read from &FE60. Acorn once thought that the address of the user port might not be &FE60 in some future version, so they advise you to read it via OSBYTE &80, and indeed you can, if you wish. Before you read from &FE60 you must set the Data Direction Register at &FE62 to 'input', that is to zero. The read from &FE60 must then be masked off to the desired channel, as follows:

Bits 7, 6 & 5	irrelevant
Bit 4	joystick right
Bit 3	joystick up
Bit 2	joystick down
Bit 1	joystick left
Bit 0	fire button

In the case of OSBYTE &80, set X to 0. On exit, bit 0 of Y holds the status of the fire button. Thus:

```
LDA #&80:LDX #0:JSR &FFF4
TYA:AND #3
```

If A<>0, the button is down. To read stick movements, set X to the channel: 1 to 4. On exit, X and Y hold the value read, LSB in X as always.

Similarly, OSBYTE &81 will wait for a key press up to a certain time limit, then move on. That's your INKEY, Silas. Let *n* be the INKEY time parameter, in hundredths of a second. Load X with *n* MOD &100 and Y with *n* DIV &100. The call returns with the ASCII code of the key pressed, if any, in X, and Y=0. If no keypress, Y=&FF. If the key was Escape, then its code (&1B) is in Y, not X. Our example will wait for a keypress for 2.57 seconds:

```
  LDA #&81:LDX #1:LDY #1 \ now d'you see
why I made it 2.57?
  JSR &FFF4
  TYA:BMI noKeyPressed
  CMP #&1B:BEQ itWasEscape
  TXA \ ASCII is in A
```

That's ordinary INKEY. The 'minus' INKEY: "scan keyboard for a particular character", is still OSBYTE &81, but on entry X=INKEY number (NOT the ASCII) of the character and Y=&FF. On exit X=&FF as well as Y if the key was being pressed. It's easier to use the other way, I reckon, because those INKEY numbers are so confusing.

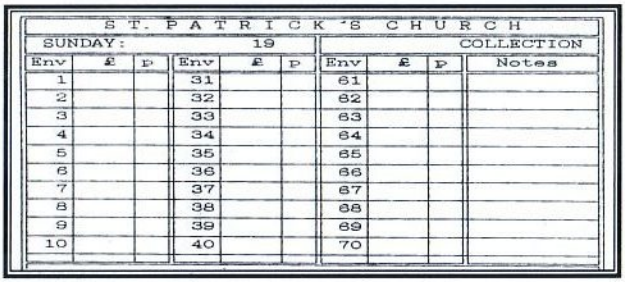
That's just to answer the two specific questions. Next time, we'll take a closer look at why it's very difficult, to say the least, to access Basic routines from machine-code, and at the low-level MOS routines which you can use instead for many purposes. Also, a potted biography of Osbert O'Beaohbeaugh, the mordant, incisive Irish poet in honour of whom OSBYTE was named.

This month's competition: are addresses &DC03 and &A085 in fact in ROM or RAM on the Master? Wrong answers win a course of lectures from Mr T on the transformation of the long vowels in the Middle and New English periods.

Form Designer (Part 2)

Lol Taylor explores just some uses for his form design shell.

The following are some examples of how the form designer can be used. You will need the two programs *FdShell* and *SetUpCh* listed in the article last month. Each of the following programs is added to FdShell to make it work. This has already been done for the versions on the magazine disc, which also includes an additional example.



S T. P A T R I C K 'S C H U R C H

SUNDAY:			19			COLLECTION			
Env	£	p	Env	£	p	Env	£	p	Notes
1			31			61			
2			32			62			
3			33			63			
4			34			64			
5			35			65			
6			36			66			
7			37			67			
8			38			68			
9			39			69			
10			40			70			

The Journal: FdShell plus +*Journ*. Only 7 complete strings are needed, and there is a lot of repetition. However, there are more short strings than there were with the FdBoxes program. It can be saved as *FdJourn*.

The Journal form can be used for invoices and statements. It would be a simple task to enter a firm's name with double width characters near the top. You will see how easy this is in later examples.

```
   10 REM Form Designer - FdJourn
   50 title$="Journal"
 4000 DEF PROCprinterOn
 4010 REM Printer codes: on, elite, doub
le width, 8 lines to the inch, English f
ont
 4020 VDU2,1,27,1,77,1,27,1,87,1,1,1,27,
1,48,1,27,1,82,1,3
 4030 ENDPROC
 4040 :
 4200 DEF PROCprinterOff
 4210 REM Default printer codes restored
in reverse order and off
 4220 VDU1,27,1,82,1,0,1,27,1,50,1,27,1,
87,1,0,1,27,1,80,3
 4230 ENDPROC
 4240 :
 5000 DEF PROCprintout
 5010 PRINT''a1$
 5020 PRINTa2$'a2$'a2$'a2$
 5030 PRINTa3$
 5040 FORI%=1TO32
 5050 PRINTa4$
 5060 PRINTa5$
 5070 NEXT
 5080 PRINTa4$
 5090 PRINTa6$
 5100 PRINTa4$'a4$
 5110 PRINTa7$
 5120 VDU1,12
 5130 ENDPROC
 5140 :
 6000 DEF PROCshortstrings
 6010 s46$=STRING$(46," ")
 6020 s3$=STRING$(3," ")
 6030 s25$=STRING$(25," ")
 6040 L46$=STRING$(46,we$)
 6050 L2$=STRING$(2,we$)
 6060 L3$=STRING$(3,we$)
 6070 L25$=STRING$(25,we$)
 6080 d46$=STRING$(46,WE$)
 6090 d2$=STRING$(2,WE$)
 6100 d3$=STRING$(3,WE$)
 6110 d25$=STRING$(25,WE$)
 6120 ENDPROC
 6130 :
 7000 DEF PROCcompletestrings
 7010 a1$=SE$+d46$+WS$
 7020 a2$=NS$+s46$+NS$
 7030 a3$=NSE$+d3$+WEs$+d25$+WEs$+d2$+WS
```

```
E$+d3$+WEs$+d2$+WSE$+d3$+WEs$+d2$+WNS$
 7040 a4$=NS$+s3$+ns$+s25$+ns$+"  "+NS$+
s3$+ns$+"  "+NS$+s3$+ns$+"  "+NS$
 7050 a5$=NSe$+L3$+wnse$+L25$+wnse$+L2$+
NSwe$+L3$+wnse$+L2$+NSwe$+L3$+wnse$+L2$+
NSw$
 7060 a6$=NSE$+d3$+WEns$+d25$+WEns$+d2$+
WNSE$+d3$+WEns$+d2$+WNSE$+d3$+WEns$+d2$+
WNS$
 7070 a7$=NE$+d3$+WEn$+d25$+WEn$+d2$+WNE
$+d3$+WEn$+d2$+WNE$+d3$+WEn$+d2$+WN$
 7080 ENDPROC
 7090 :
```

The Register: FdShell plus +*Reg1*. Save as *FdReg1*. Do not change any line numbers. Notice the printer codes; we are using 80 characters by 80 lines. Notice also the STRING$ commands in lines 6070-6100 where concatenation is used.

There are 40 squares across in which to enter marks, so this form would make a good class attendance register covering a fortnight. Or it would make an excellent weekly subscriptions schedule. I also use it for check lists.

```
   10 REM Form Designer - FdReg1
   50 title$="REGISTER "
 4000 DEF PROCprinterOn
 4010 REM Printer codes: on, 8 character
s per inch
 4020 VDU2,1,27,1,48
 4030 ENDPROC
 4040 :
 4200 DEF PROCprinterOff
 4210 REM Printer codes reversed and off
 4220 VDU1,27,1,50,3
 4230 ENDPROC
 4240 :
 5000 DEF PROCprintout
 5010 PRINT'a1$
 5020 FORI%=1TO3:PRINTa2$:NEXT
 5030 PRINTa3$
 5040 FORI%=1TO11:PRINTa4$:NEXT
 5050 FORI%=1TO30:PRINTa5$,a4$:NEXT
 5060 PRINTa6$
 5070 VDU1,12
 5080 ENDPROC
 5090 :
 6000 DEF PROCshortstrings
 6010 s4$=STRING$(4," ")
 6020 s13$=STRING$(13," ")
 6030 s73$=STRING$(73," ")
 6040 D13$=STRING$(13,WE$)
 6050 D73$=STRING$(73,WE$)
 6060 L13$=STRING$(13,we$)
 6070 Da$=STRING$(20,WEs$+WE$+WE$)
 6080 Db$=STRING$(20,WEn$+WE$+WE$)
 6090 sa$=STRING$(20,ns$+"  ")
 6100 La$=STRING$(20,wnse$+we$+we$)
 6110 ENDPROC
 6120 :
 7000 DEF PROCcompletestrings
 7010 a1$=s4$+SE$+D73$+WS$
 7020 a2$=s4$+NS$+s73$+NS$
 7030 a3$=s4$+NSE$+D13$+Da$+WNS$
 7040 a4$=s4$+NS$+s13$+sa$+NS$
 7050 a5$=s4$+NSe$+L13$+La$+NSw$
 7060 a6$=s4$+NE$+D13$+Db$+WN$
 7070 ENDPROC
 7080 :
```

The Numbered Register: FdReg1 plus +*Reg2*. This listing includes a new procedure, *PROCa7(I%)*. This replaces a4$ in line 5050. It adds the number I%, and right justifies it to a two character width.

```
   10 REM Form Designer - FdReg2
   50 title$="REGISTER with numbered lin
es "
   60 :
 5050 FORI%=1TO30:PRINTa5$:PROCa7(I%):NE
XT
 6110 s11$=STRING$(11," ")
 6120 ENDPROC
 6130 :
 7200 DEF PROCa7(I%)
 7210 PRINTs4$+NS$+FNrj(STR$(I%),2)+s11$
+sa$+NS$
 7220 ENDPROC
 7230 :
```

Tape Library Borrowers Sheets: FdShell plus *+Borro*. The main thing to notice here is that in order to get room to list ten tape numbers on a page we need to program the printer for 7 lines per inch approximately. Save the program as *FdBorro*.

```
   10 REM Form Designer - FdBorro
   50 title$="Tape Library: tapes out on
 loan"
 4000 DEF PROCprinterOn
 4010 REM Printer codes: on, 7 lines per
 inch approx
 4020 VDU2,1,27,1,51,1,30
 4030 ENDPROC
 4040 :
 4200 DEF PROCprinterOff
 4210 REM Default printer code restored
and off
 4220 VDU1,27,1,50,3
 4230 ENDPROC
 4240 :
 5000 DEF PROCprintout
 5010 PRINT'''a1$
 5020 PRINTa2$'a3$
 5030 FORI%=1TO10:PRINTa4$'a5$'a6$'a5$'a
6$'a5$:NEXT
 5040 PRINTa7$
 5050 VDU1,12
 5060 ENDPROC
 5070 :
 6000 DEF PROCshortstrings
 6010 s4$=STRING$(4," ")
 6020 s5$=STRING$(5," ")
 6030 s8$=STRING$(8," ")
 6040 s9$=STRING$(9," ")
 6050 L4$=STRING$(4,we$)
 6060 L8$=STRING$(8,we$)
 6070 L9$=STRING$(9,we$)
 6080 d4$=STRING$(4,WE$)
 6090 d8$=STRING$(8,WE$)
 6100 d9$=STRING$(9,WE$)
 6110 D1$=STRING$(3,WSE$+d9$+WEs$+d9$)
 6120 D2$=STRING$(3,NS$+"MEMBER'S "+ns$+
"  DATE   ")
 6130 D3$=STRING$(3,NS$+"   NAME  "+ns$+
" OUT"+ns$+"IN  ")
 6140 D4$=STRING$(3,WNSE$+d9$+WEns$+d4$+
WEns$+d4$)
 6150 D5$=STRING$(3,NS$+s9$+ns$+s4$+ns$+
s4$)
 6160 D6$=NSe$+L9$+wnse$+L4$+wnse$+L4$+S
TRING$(2,NSwe$+L9$+wnse$+L4$+wnse$+L4$)
 6170 D7$=STRING$(3,WNE$+d9$+WEn$+d4$+WE
n$+d4$)
 6180 ENDPROC
 6190 :
 7000 DEF PROCcompletestrings
 7010 a1$=s5$+SE$+d8$+D1$+WS$
 7020 a2$=s5$+NS$+"  TAPE  "+D2$+NS$
 7030 a3$=s5$+NS$+" NUMBER "+D3$+NS$
 7040 a4$=s5$+NSE$+d8$+D4$+WNS$
 7050 a5$=s5$+NS$+s8$+D5$+NS$
 7060 a6$=s5$+NS$+s8$+D6$+NSw$
 7070 a7$=s5$+NE$+d8$+D7$+WN$
 7080 ENDPROC
 7090 :
```

The Envelope Collection Form: FdShell plus *+Envs*. Here we are at last, this is the form which I mentioned at the beginning of last month's article. This was the one I built up for recording the church envelope collections, and it has worked very well. What you have learned from entering all the other forms should cover the intricacies of this one.

```
   10 REM Form Designer - FdEnvs
   50 title$="Planned Giving Collection
Envelopes"
 4000 DEF PROCprinterOn
 4010 REM Printer codes: on, elite, doub
le width, 8 lines to the inch, English f
ont
 4020 VDU2,1,27,1,77,1,27,1,87,1,1,1,27,
1,48,1,27,1,82,1,3
 4030 ENDPROC
 4040 :
 4200 DEF PROCprinterOff
 4210 REM Default printer codes restored
 in reverse order and off
 4220 VDU1,27,1,82,1,0,1,27,1,50,1,27,1,
87,1,0,1,27,1,80,3
```

```
 4230 ENDPROC
 4240 :
 5000 DEF PROCprintout
 5010 PRINT'
 5020 PRINTa1$'a2$'a3$'a4$'a5$'a6$'a7$
 5030 FOR I%=1TO10:PROCa8:PRINTa9$:NEXTI
%
 5040 PRINTa10a$'a11$'a12$'a9$
 5050 FOR I%=11TO20:PROCa8:PRINTa9$:NEXT
I%
 5060 PRINTa10a$'a11$'a12$'a9$
 5070 FOR I%=21TO30:PROCa8:PRINTa9$:NEXT
I%
 5080 PRINTa10b$
 5090 VDU1,12
 5100 PRINT'
 5110 PRINTa1$'a2$'a3$'a4$'a5$'a6$'a7$
 5120 FORI%=91TO100:PROCa8:PRINTa9$:NEXT
I%
 5130 PRINTa10a$'a11$'a12$'a9$
 5140 FORI%=101TO110:PROCa8:PRINTa9$: NE
XTI%
 5150 PRINTa10a$'a11$'a12$'a9$
 5160 FORI%=111TO120:PROCa8:PRINTa9$: NE
XTI%
 5170 PRINTa10b$
 5180 VDU1,12
 5190 ENDPROC
 5200 :
 6000 DEF PROCshortstrings
 6010 l2$=STRING$(2,we$)
 6020 l3$=STRING$(3,we$)
 6030 l4$=STRING$(4,we$)
 6040 l10$=STRING$(10,we$)
 6050 s3$=STRING$(3," ")
 6060 s4$=STRING$(4," ")
 6070 s6$=STRING$(6," ")
 6080 s10$=STRING$(10," ")
 6090 s11$=STRING$(11," ")
 6100 s46$=STRING$(46," ")
 6110 d2$=STRING$(2,WE$)
 6120 d3$=STRING$(3,WE$)
 6130 d4$=STRING$(4,WE$)
 6140 d10$=STRING$(10,WE$)
 6150 d22$=STRING$(22,WE$)
 6160 d23$=STRING$(23,WE$)
 6170 d46$=STRING$(46,WE$)
 6180 ENDPROC
 6190 :
 7000 DEF PROCcompletestrings
 7010 a1$=SE$+d46$+WS$
 7020 a2$=NS$+s6$+"S T. P A T R I C K 'S
  C H U R C H"+s6$+NS$
 7030 a3$=NSE$+d23$+WSE$+d22$+WNS$
 7040 a4$=NS$+" SUNDAY:"+s10$+"19   "+NS
$+s11$+"COLLECTION "+NS$
 7050 a5$=NSE$+d3$+WEs$+d4$+WEs$+d2$+WSE
$+d3$+WEs$+d4$+WEs$+d2$+WNSE$+d3$+WEs$+d
4$+WEs$+d2$+WSE$+d10$+WNS$
 7060 a6$=NS$+"Env"+ns$+"  "+`$+" "+ns$+
"p "+NS$+"Env"+ns$+"  "+`$+" "+ns$+"p "+
NS$+"Env"+ns$+"  "+`$+" "+ns$+"p "+NS$+"
  Notes   "+NS$
 7070 a7$=NSE$+d3$+WEns$+d4$+WEns$+d2$+W
NSE$+d3$+WEns$+d4$+WEns$+d2$+WNSE$+d3$+W
Ens$+d4$+WEns$+d2$+WNSE$+d10$+WNS$
 7080 a9$=NSe$+l3$+wnse$+l4$+wnse$+l2$+N
Swe$+l3$+wnse$+l4$+wnse$+l2$+NSwe$+l3$+w
nse$+l4$+wnse$+l2$+NSwe$+l10$+NSw$
 7090 a10$=d3$+WEn$+d4$+WEn$+d2$+WNE$+d3
$+WEn$+d4$+WEn$+d2$+WNE$+d3$+WEn$+d4$+WE
n$+d2$+WNE$+d10$
 7100 a10a$=NSE$+a10$+WNS$
 7110 a10b$=NE$+a10$+WN$
 7120 a11$=NS$+s46$+NS$
 7130 a12$=NSE$+d3$+WEs$+d4$+WEs$+d2$+WS
E$+d3$+WEs$+d4$+WEs$+d2$+WSE$+d3$+WEs$+d
4$+WEs$+d2$+WSE$+d10$+WNS$
 7140 ENDPROC
 7150 :
 7500 DEF PROCa8
 7510 PRINTNS$+FNrj(STR$(I%),3)+ns$+s4$+
ns$+"  "+NS$+FNrj(STR$(I%+30),3)+ns$+s4$
+ns$+"  "+NS$+FNrj(STR$(I%+60),3)+ns$+s4
$+ns$+"  "+NS$+s10$+NS$
 7520 ENDPROC
 7530 :
```

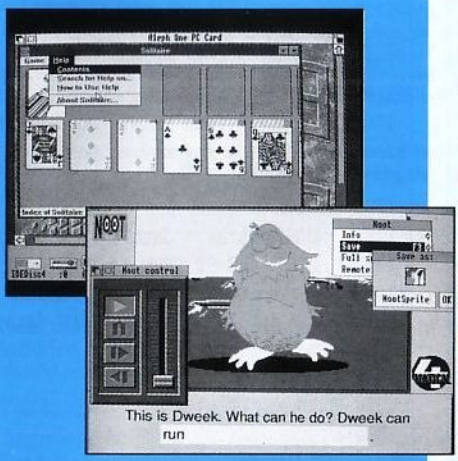
It's up to you where you go from here. Try designing invoices or letter heads, for example. Or you might use a frame around a fancy title page for you memoirs? What about printing framed tickets for amateur shows? There are plenty of possiblities; good luck!

New Generation!

RISC user

RISC User, the highly popular magazine for Archimedes users, is bigger and better. The new RISC User is now B5 size which offers a sophisticated design, bigger colour illustrations and bigger pages with more information. Altogether better value and no increase in price.

RISC User is still a convenient size to assemble into an easy-to-use reference library, containing all the information you need as an Archimedes user. Every issue of RISC User offers a wealth of articles and programs with professionally written reviews, lively news, help and advice for beginners and experienced users, and items of home entertainment. Altogether RISC User has established a reputation for accurate, objective and informed articles of real practical use to all users of Acorn's range of RISC computers.



YOUR ARCHIMEDES ON THE PHONE
A guide to communications introducing the readers to bulletin boards, file transfer protocols, conferencing and comms software.

REVELATION IMAGE PRO
A review of this impressive painting package from Longman Logotron.

RISC OS 3 SCREEN MODES
A comprehensive look into the range of Desktop modes available to Arc users.

THE PC EMULATOR SURVIVAL GUIDE (1)
New series on how to use the PC emulator which starts with introducing the PC world.

FAXPACK UPDATE
A look at the latest addition to Computer Concept's Fax Pack which now allows you to use your computer as an answering machine.

QUICK INDEX
A useful index generator for books and manuals.

WRITE-BACK
The readers' section of RISC User for comment, help, information - a magazine version of a bulletin board.

WP/DTP
Articles on using different DTP and WP packages.

INTO THE ARC
A regular series for beginners.

TECHNICAL QUERIES
A column which answers your technical queries.

NOOT
Review of the new animation package for the educational market.

SUBSCRIPTION DETAILS

As a member of BEEBUG you may extend your subscription to include RISC User for only £10.50 (overseas see table).

Destination	Additional Cost
UK,BFPO &Ch Is	£10.50
Rest of Europe and Eire	£15.40
Middle East	£19.60
Americas and Africa	£21.90
Elsewhere	£33.00

```
 1280 IF root=-1 THEN root=R:=node
 1290 IF L=-1 THEN RLink(tnode)=R:=node
 1300 REPEAT:tnode=L:L=RLink(L):UNTIL L=
-1
 1310 RLink(tnode)=R
 1320 =node
 1330 :
 1340 DEF FNfind_element(data)
 1350 LOCAL f%,node:f%=FALSE:node=root
 1360 REPEAT
 1370   IF data=Data(node) THEN f%=TRUE
ELSE tnode=node:IF data<=Data(node) THEN
 node=LLink(node):t$="L" ELSE node=RLink
(node):t$="R"
 1380 UNTIL f%=TRUE OR node=-1
 1390 =node
 1400 :
 1410 DEF PROCinorder1(node)
 1420 PRINT''"Inorder Display"
 1430 IF root=-1 THEN PRINT'"No data in
tree" ELSE PROCinorder(node)
 1440 PRINT"Press any key to continue":L
=GET
 1450 ENDPROC
 1460 :
 1470 DEF PROCinorder(node)
 1480 IF LLink(node)>-1 THEN PROCinorder
(LLink(node))
 1490 PRINT Data(node)
 1500 IF RLink(node)>-1 THEN PROCinorder
(RLink(node))
 1510 ENDPROC
 1520 :
 1530 DEF PROCinput_data
 1540 LOCAL f%:f%=FALSE
 1550 PRINT''"Input Data"
 1560 REPEAT
 1570   INPUT"Data: " data
 1580   IF data=9999 THEN f%=TRUE ELSE I
F root>-1 PROCadd_element(data,0) ELSE r
oot=free:free=RLink(free):Data(root)=dat
a:LLink(root)=-1:RLink(root)=-1
 1590 UNTIL f%
 1600 ENDPROC
 1610 :
 1620 DEF PROCdelete_data
 1630 LOCAL f%,f1%:f%=FALSE
 1640 PRINT''"Delete Data"
 1650 IF root=-1 THEN PRINT'"No data in
tree":PRINT"Press any key to continue":L
=GET:ENDPROC
 1660 REPEAT
 1670   INPUT"Data: " data
 1680   IF data=9999 THEN f%=TRUE ELSE f
1%=FNdelete_element(data):IF f1%=-1 THEN
 PRINT data;" not found" ELSE PRINT data
;" deleted"
 1690 UNTIL f% OR root=-1
 1700 IF root=-1 THEN PRINT'"No data in
tree":PRINT"Press any key to continue":L
=GET
 1710 ENDPROC
 1720 :
 1730 DEF PROCfind_data
 1740 LOCAL f%,f1%:f%=FALSE
 1750 PRINT''"Find Data"
 1760 IF root=-1 THEN PRINT'"No data in
tree":PRINT"Press any key to continue":L
=GET:ENDPROC
 1770 REPEAT
 1780   INPUT"Data: " data
 1790   IF data=9999 THEN f%=TRUE ELSE f
1%=FNfind_element(data):IF f1%=-1 THEN P
RINT data;" not found" ELSE PRINT data;"
 found at position ";f1%
 1800 UNTIL f%
 1810 ENDPROC
 1820 :
 1830 DEF FNmenu
 1840 LOCAL c%
 1850 PRINT"BINARY TREE DEMONSTRATION"'
 1860 PRINTTAB(5)"1. Input Data"
 1870 PRINTTAB(5)"2. Delete Data"
 1880 PRINTTAB(5)"3. Find Data"
 1890 PRINTTAB(5)"4. Inorder Display"
 1900 PRINTTAB(5)"5. Exit"
 1910 PRINT'"Use '9999' to terminate dat
a entry"
 1920 PRINT'"Enter 1 - 5:";
 1930 REPEAT:c%=GET-48:UNTIL c%>0 AND c%
<6
 1940 =c%
```

HINTS and tips HINTS and tips HINTS and tips HINTS and tips HINTS and tips

Please do keep sending in your hints for all BBC and Master computers. Don't forget, if your hint gets published, there's a financial reward.

CONTROL CHARACTERS IN FUNCTION KEY STRINGS

Cliff Blake

In Hints and Tips Vol.11 No.7, Brian Lowe described a method for getting the character with ASCII code 255 into a function key definition. May I remind readers that the more convenient method which can be put into an initialising program which sets up the keys is:

```
*KEY7|!|?
```

Essentially, the |! adds 128 to the ASCII code of the character which follows it, and | on its own subtracts 64 from the ASCII code (like pressing the Ctrl key). There is a slight oddity in that the ASCII code for the question mark is 63, so |? gives the value -1, but ignoring the sign bit gives 127. Thus the whole of the above expression is 128 + 127 = 255.

Thus the complete list of ASCII codes can be included in function key definitions by using the following table:

```
0 to 31        '|@'    to '|_'
32 to 126      ' '     to '~'
128 to 159     '|!|@' to '|!|_'
160 to 254     '|! '   to '|!~'
255            '|!|?'
```

Note that the single quote characters are only included to show where there are spaces: they should not be included in the actual definition.

PROGRAM PROTECTOR

Andrew Nelson

The following program will prevent anyone from using your private programs by making them unusable by those who do not know a password. It is in the form of a procedure which should be added to the end of the program you wish to protect.

```
32000 DEF PROCLOCK
32010 INPUT"PASSWORD ",KEY$
32020 I%=PAGE+3
32030 REPEAT
32040 FORX%=I%+1 TO I%-4+?I%
32050 Y%=&1F AND (ASC MID$(KEY$,X%
MOD LEN KEY$,1))
32060 ?X%=(?X% AND &E0)+((?X% AND
&1F)EOR Y%)
32070 NEXT
32080 I%=X%+3
32090 UNTILX%>TOP-&C1
32110 ENDPROC
```

To protect your program, you should append this procedure to the end of the program by using the spooling method. In summary, type in the above program exactly as shown, and type:

```
*SPOOL LOCK
LIST
*SPOOL
```

This will create a file called LOCK on disc. Now load in the program to be protected (whose last line number must be less than 32000), and type:

```
*EXEC LOCK
```

This adds the procedure to your program. Now type:

```
PROCLOCK
```

and enter a password when you are prompted, terminated by Return. The machine will pause for a while, while your program is scrambled. When the procedure finishes, if you list your program it will look like a mess: it is now scrambled. Save this program (though don't overwrite the original unscrambled copy of your program, just in case). To use the scrambled version of the program, load it in and type:

```
PROCLOCK
```

entering your password at the prompt. The machine will pause while the unscrambling takes place, and lo and behold! Your program will have reappeared. If you enter the wrong password for decoding, then you must re-load the scrambled program again.

BEEBUG CHRISTMAS COMPETITION RESULTS

After wading through the very large response we received from the BEEBUG Christmas Competition in Vol.11 No.7, we've managed to pick out five correct entries. the lucky winners are:

R.M.Brookes, West Midlands
Richard Driessen, Netherlands
Andrew Peel, Nottingham
Clare & Jean Wood, Roxburghshire
C.Bradbury, Suffolk

There were a number of answers permissible for each of the three alphametic problems we set, especially the MINCE PIES EATEN problem which had 16 different answers! There isn't room to list all the possible answers here, so we've included on the magazine disc a program which solves alphametics for you.

Thanks to everyone who entered, and better luck next time.

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 40p per word (inclusive of VAT) and these will be featured separately. Please send all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS.

M512 in excellent working order, included are two 5.25 floppies, AMX mouse, joysticks and the following EPROMs in addition to the resident View and Viewsheet: Epson printer driver, Wordwise Plus, Pascal, Screendump, Toolkit and Graphics, full original manuals and discs for the Master 512 plus various extra manuals, years of BBC dedicated magazines, software for both BBC & PC mode including games. Only £300 o.n.o. Tel. (0535) 662157.

M128, disc drive 40/80T, 50 games, 7 blank discs, Speech!, joystick, Edit, View, Viewsheet, SRAM £250. Tel. (0277) 890519.

65C102 co-processor for Master £50, BEEBUG C ROMS/discs inc. stand alone generator £40, BROM+ toolkit, Exmon II and Hyperdrive printer driver ROMs, Quad cartridge, £7 each, Overview £40, White Knight II chess £7, books advanced SWRAM User, Birmbaum Assembler, Smith's Assembler and Workshop £5 each. Tel. (0785) 42294.

WANTED: Copy of the ECONET level 1 or 2 Fileserver software. Tel. (0670) 521055 after 6pm.

WANTED: Hard drive utilities disc such as "Park" and "Format" for a 40Mb Seegate ST251-1 (ST506) fitted to a BBC B. **FOR SALE:** The Publisher ROM £25, Printwise £8, Hyperdriver £15, Also available WW+, ADU, ADI or ADT. W.H.Y? I have Interbase!, is there a User Group or any other users who have come to terms with its quirks? **HELP WANTED:** No software for 65c102 co-processor. Offers please after 6pm. Tel. (0525) 715013.

Switched mode PSU model EXT90/12 £25, Spellcheck III + ROM £15, Toolkit ROM £7, Spellcheck discs £10, OverView ROMs £55, Master OS ROM £18, BEEBUG discs list available £3 each, 2764 new EPROMs £2 each, 2732a new EPROMs £3 each, 6264 LP RAM £3 each, 6809 inch rack with guides £25, 6809 CPU computer complete with OS software. Tel. (0254) 701573.

Stop Press DTP (complete set of ROMs, discs and manual) £25, Vine Micros Replay (tape to disc) for Watford DDFS £10, Watford File Plus database (ROM based for high speed) £10, BEEBUG Basic programming Toolkit ROM £5, Masterfile II database £10, Word-Aid WW+ comprehensive utilities ROM £8. Tel. (0727) 830264.

Electron, Plus 1, AP3-AP4 disc interface, AP6 ROM expansion board, 5.25 + 3.5" disc drives in plinth, Acorn tape deck, View, Viewsheet, Games, leads and manuals, all original packing. Tel. (0722) 331763 eves.

WANTED URGENTLY: Latest versions 5.25" Shibumi Problem Solver + Tull mouse driver, Turbo Pascal V3 or V4 for the Master 512, we'll pay good price. Please write to: Mr Henri Comyn, Zandvoordsestraat No.12, 8902 IEPER (BELGIUM).

WANTED: Acorn User from No.1 in good condition, continuous runs would be preferred but not essential. Tel. 061-303 7414 after 6pm.

Complete BBC/Master upgrade system to Archimedes 310, Med. Res. colour monitor, 4Mb MEMC1a upgrade, interfaced 5.25" twin 80/40 disc drives in monitor plinth, serial link, quality software and games £700 o.n.o. Tel. (0572) 821313.

WANTED: AMX mouse for use with AMX Pagemaker, or perhaps you can tell me why my present mouse is now working only horizontally and how I can repair it? I'm really hoping for someone to donate rather than to buy. Tel. (0295) 720812.

BBC B issue 7 with Opus Challenger DDOS, 512k RAM disc, Wordwise, Interword, Spellmaster, many extras £225. Tel. (0245) 281988.

Exmon II ROM £8, Sleuth ROM £8, Printwise £8, Dumpmaster ROM £8, View printer driver Generator £5, all boxed as new, BEEBUG magazine Vol.1 Issue 1 to date (10+ vols.), including Magscan £30, Books £6 each; Dickens & Holmes New Advanced UG, Pharo Advanced Disc UG, Bray Dickens & Holmes Advanced UG, Pharo Advanced Basic ROM UG. Solidisk SWR 128 for BBC B complete £16 prices include postage. Tel. (0386) 556197.

Master 128, View, Viewsheet with Welcome manual and disc £175, Dump Out 3 ROM £8, Interword £20, Viewstore £12, Discmaster £8, AU with manuals, also numerous books and discs - send for complete list - bargains! - upgrading. Tel. (0286) 880997.

BBC B issue 7, Pace DFS, Tape/Disc ROM with user and advanced user manuals £110, Opus 40/80T disc drive £30, JVC colour monitor £70, Wordwise plus ROM £10, Exmon II £8, Caretaker £8, Disc Doctor £8, Accelerator/G Code ROMs £10, buffer/backup ROM £5, Pace modem/Commstar ROM £15, all with manuals, several books, numerous discs, magazines - send for complete list - bargains! - upgrading. Tel. (0286) 880997.

Help! WANTED: A main drive gear (very small cog) for a printer/plotter MCP40, (also sold by Tandy under a different number), this machine uses 4 small coloured ballpens, I only need the drive gear, but would consider buying a broken or working machine. Tel. (0734) 582115 day 081-954 3566 eves.

BBC B issue 7 with ADFS £90, Silver Reed EXP400 daisy wheel printer £55, prefer buyer to collect. Tel. (0226) 762450.

BBC B issue 7, WE DDFS, Acorn ADFS, ZIF socket, WE Shadow RAM, WE ROM board, all manuals included, other ROMs and software available £150 o.n.o. Could reluctantly separate. Offers? Tel. (0525) 715013 after 6pm.

M512 & double 80T 5.25" disc drive, black & white monitor, Panasonic KXP-1180 printer, mouse, plug in ROM cartridge, Prism 200 modem, BEEBUG designed EPROM programmer, all reference manuals, software including Overview, Masterfile, Dabs Press 512 shareware, some BEEBUG discs and tapes and complete set of BEEBUG mags from Vol.1 No.1 £400 o.n.o. buyer collects. Tel. (0923) 775098 after 7pm.

BBC B, Watford dual disc drive (3.5 and 5.25), numerous software, manuals, modem etc. £150. Tel. (0746) 765812.

EPROMs M27128 AFI PGM12.5v SGS-Thomson brand new in anti-static strip £1.25 each or £15 for strip of 15. Tel. (0234) 856070.

BBC B/Master Prestel adaptor complete with ROM, handbook and connectors, nearest offer to £25 secures. CC Interword ROM and handbook £25. Tel. 086 732 8778.

Master 128, BEEBUG C ROM and compiler, BB internal modem and chip, Viewstore, Viewspell, Micro Prologue, 4 manuals and 5.25" double disc drive. Offers? 081-200 7863 eves.

EMULATING SRWRITE/SRREAD

I am the proud owner of a BBC B with sideways RAM, ADFS and shadow RAM fitted. I am therefore very interested in magazine programs that are applicable to both sideways RAM and ADFS. I wrote previously about how to implement the *SRWRITE command on my Watford Electronics ROM/RAM board and this was covered in BEEBUG Vol.10 No.10 - thanks. I have now come across an ADFS program which uses the *SRREAD command. Can anyone provide a similar procedure to emulate this command on my system?

T.D.Parsons

MASTER COMPACT PLEA

I have decided to renew my subscription to BEEBUG for another year but I would like to point out that I feel too little attention is paid to the Master Compact. Although this machine is very similar to the BBC Master 128, there are sufficient differences to warrant at least the occasional article on the technical differences. In addition all your adverts tend to ignore the 3.5" disc required.

As there are many BBC machines now "doing the rounds" a second time, and new users learning about software and hardware that may have been reviewed some ten years ago, I think BEEBUG would benefit by introducing a "Second Time Around" article which covered items such as PageMaker, the Morley products, and other major events that made the BBC range of machines a success.

Brian Odurny

It is a fact of life that as interest in the various BBC micros has declined, it is the Master Compact which is the one machine to have fallen from favour quicker than most. Being the only BBC micro to use 3.5" discs and the ADFS exclusively, it was never as popular as it deserved to be in my view - I used a Compact for many years as my main work machine before eventually switching to an Archimedes.

As Brian Odurny says, there are differences between the Compact and the Master 128. If any readers can contribute something to BEEBUG on this topic we would be delighted to consider.

All BEEBUG magazine discs are available in 3.5" ADFS disc format, and so are all our other BEEBUG disc collections, Best of BEEBUG discs, ASTAAD, MagScan etc. - see the central pages for more details.

Repeating reviews of selected products might be useful to newer readers using a BBC micro for the first time, though supply might now be a problem. What do other readers think of this idea? We shall certainly be giving it some thought.

CHANGES TO TELETEXT

I have a Master 128 with a Teletext adaptor which I use to download onto disc several items from CEEFAX using *Grabit*. This system has worked very well for a long time until very recently when the BBC modified the CEEFAX system to speed it up. But this modification has upset the *Grabit* download and I get nothing but garbage when I try and get a printout. Can you please inform me what alterations should be carried out to the *Grabit* program to bring it into line with the new BBC setup?

S.A.Usher

We are unable to answer this question directly as we have no recent experience of using Teletext or the Grabit program for downloading. Can any other readers help in this matter?

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES

		BEEBUG & RISC USER
£18.40	1 year UK, BFPO, Ch.I	£28.90
£27.50	Rest of Europe & Eire	£42.90
£33.50	Middle East	£53.10
£36.50	Americas & Africa	£58.40
£39.50	Elsewhere	£62.50

BACK ISSUE PRICES

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. There is no VAT on magazines.

Volume	Magazine	5"Disc	3.5"Disc
6	£1.00	£3.00	£3.00
7	£1.10	£3.50	£3.50
8	£1.30	£4.00	£4.00
9	£1.60	£4.00	£4.00
10	£1.60	£4.75	£4.75
11	£1.90	£4.75	£4.75

POST AND PACKING

Please add the cost of p&p when ordering. When ordering several items use the highest price code, plus half the price of each subsequent code.

Stock Code	UK, BFPO Ch.I	Europe, Eire	Americas,Africa, Mid East	Elsewhere
a	£1.00	£1.60	£2.40	£2.60
b	£2.00	£3.00	£5.00	£5.50

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 840303, FAX: (0727) 860263
Manned Mon-Fri 9am-5pm (for orders only 9am-6pm and 9am-5pm Saturdays)
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by RISC Developments Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: Mark Moxon
Editorial Assistance: Marshal Anderson
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Subscriptions: Sue Baxter
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

Printed by Arlon Printers (0923) 268328 ISSN - 0263 - 7561

Magazine Disc

March 1993

ADFS BACKUP - This utility provides a fast and efficient way to back up ADFS discs particularly for single drive users.

GRAVITY AND ORBITS - Learn how to control the orbits of satellites with this interesting and challenging application in the world of physics.

FIFTEEN PUZZLE - This is an excellent implementation of the classic sliding-block puzzle, where 15 numbered squares have to be moved around a board to position them in numeric order.

WEATHER UPDATE - Further updates have been incorporated into our previous weather plotting programs to give the complete program provided here.

BEEBUG WORKSHOP - This month's program shows how insert and delete functions can be incorporated into the management of a binary tree.

SIDEWAYS POET - This intriguing application enables you to generate poetry in the style of such celebrated poets as Shakespeare, Burns and Wordsworth.

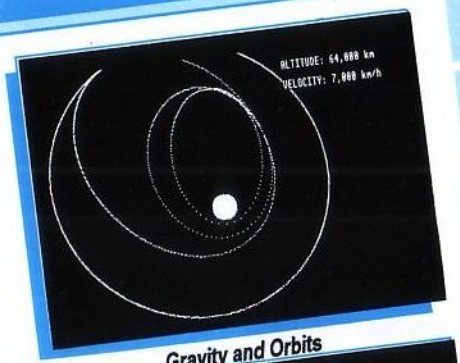
FORM DESIGNER (PART 2) - We are repeating the programs FDShell and Setupch from last month together with six complete examples of form design.

MAGSCAN DATA - Bibliography for this issue of BEEBUG (Vol.11 No.9).

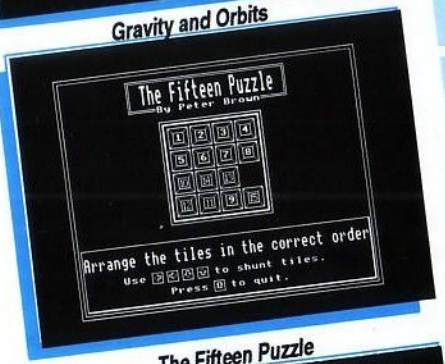
BONUS ITEMS

ALPHAMETICS - a program to help you solve alphametics like that in BEEBUG's Christmas competition.

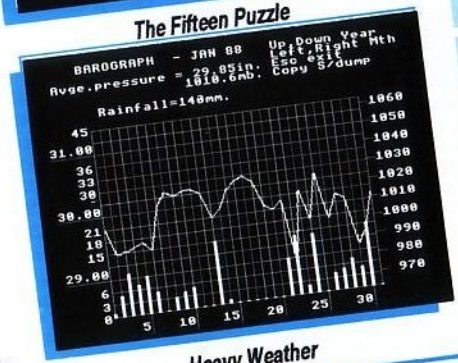
HYBRID MUSIC - two sample tunes for the Hybrid 4000 or 5000 from the Dudley College on-line music library.



Gravity and Orbits



The Fifteen Puzzle



Heavy Weather

Upgrading to an Archimedes

We know that many BEEBUG readers have already upgraded to an Archimedes, and no doubt many more will choose to follow a similar route. For their benefit we offer our advice to help them make a sensible decision on whether to upgrade and if so, what path to take.

Any prices quoted relate to our associated company Beebug Ltd., but note that all prices, particularly those on trade-ins and secondhand items, are likely to change without notice. You should always telephone or write for the latest information.

Archimedes A5000

What System to Choose

All new Archimedes systems are now supplied with the RISC OS 3.10 operating system. Any secondhand system should be upgraded to this. Based on the experience of existing users, we would strongly recommend a minimum of 2Mb of RAM. Most users find a hard disc adds significantly to the convenience of using an Archimedes, but you can always add a low-cost hard drive later, and more memory, but check on the likely price of future expansions - it is not necessarily the same for all machines. If you might be interested in more specialised add-ons (scanners, digitisers, etc.) then check the expansion capability of your preferred system.

Compatibility and Transferability

You will need to decide to what extent you wish to continue using existing discs and disc drives on an Archimedes. An Archimedes and a BBC micro can be directly connected for transfer of files. You can also connect a 5.25" drive to an Archimedes via an additional interface to continue to access 5.25" discs (ADFS format).

Our DFS Reader will also allow files to be transferred to the Arc from DFS format discs. However, none of this is possible with the latest A3010/A3020/A4000 systems.

Much BBC micro software will run directly on an Archimedes, or via the 6502 emulator. However, consider this carefully; in our experience, despite prior misgivings, most Archimedes users find that they rapidly adjust to the Desktop environment of the Archimedes, and quickly abandon the software and data of their old system after an initial period.

Software for the Archimedes

The Archimedes is supplied complete with a range of basic applications software. Before embarking on any further purchases it may be better to familiarise yourself with the new machine. Most users look for a word processor (or DTP package), maybe a spreadsheet, or a database, plus other more specialist software. We cannot give detailed guidance here, but back issues of RISC User contain a wealth of useful information - we can advise on suitable issues.

the outset. Note: the price on some systems includes a monitor; in other cases a choice of monitor is available at an additional cost. The details given in the table are minimum specifications of the different Archimedes models.

The A3010

It may also be possible to trade in an existing monitor and/or disc drive, but check if your existing monitor is suitable for use with an Archimedes first. You may find it better to advertise your BBC system in BEEBUG and sell privately - this applies particularly to any software and hardware add-ons which cannot be accepted for a trade-in. In future, all personal ads for Archimedes systems in RISC User will also be included in BEEBUG. You may also defer a trade-in until a later date provided you make this clear at the time of purchase.

Archimedes Systems - Typical or Current Prices

		Secondhand	New
+	A310 1Mb RAM	£350	
+	A410/1 1Mb RAM	£565	
+	A420/1 2Mb RAM, 20Mb hard drive	£650	
+	A440/1 4Mb RAM, 40Mb hard drive	£725	
+*	A3000 1Mb RAM	£350	
*	A3010 1Mb RAM, Family Solution		£ 499.00
*	A3020 2Mb RAM, 60Mb hard drive		£1056.33
*	A4000 2Mb RAM, 80Mb hard drive		£1115.08
*	A5000 2Mb RAM, 80Mb hard drive		£1643.83
+*	Acorn standard colour monitor	£145	£ 258.50

All systems above include a single floppy disc drive.
New (*) and secondhand (+) - all prices inc. VAT.
The A5000 price includes a multiscan colour monitor,
A3020/A4000 price includes standard colour monitor.

BBC Micros - Typical Trade-in Prices

Model B (Issue 7)	£ 35
Model B (issue 7) + DFS	£ 75
Master 128	£125
Master Compact	£ 50

General Advice

It is advisable to discuss your requirements with the BEEBUG technical team before making a final decision on what you want. Try to anticipate future expansion needs at