Page: 6

[GC1](factual) Heading is on p.90. The original table of contents prints p.87.

Page: 14

[GC2](factual) Also, an underscore (_) appears in MODE 7 as an en dash (–).

Page: 15

[GC3](typo) Should be a forward slash (/).

Page: 19

[GC4](typo) Missing underscore – line should read:

```
LET this_year=1984
```

Page: 33

[GC5](typo) Missing first parameter to GCOL – line should read:

```
280 GCOL 0,0
```

Page: 34

[GC6](typo) Missing ampersand for hex constant – read:

```
90 @%=&2020A
```

Page: 34

[GC7](typo) 2 and = characters transposed – corrected line reads:

```
320 X2=(-B-SQR(DISCRIM))/(2*A)
```

Page: 35

[GC8](typo) Add semicolon to read:

```
60 VDU 29,640;512;
```

Page: 41

[GC9](typo) Missing line 130 – beneath line 120 insert the following:

```
130 LDA #&0A : JSR write
```

Page: 41

[GC10](typo) Replace $ with & – line should read:

```
200 LDA #&0B : JSR write
```

Page: 54

[GC11](factual) ASCII codes specify 7 bits only; the sequence 1001010 corresponds to the hexadecimal value given for J.

Page: 59

[GC12](typo) Line 30 should have an extra space inserted, to read:

```
30 PRINT “HELLO ”;N$;“. HOW ARE YOU?”
```

Page: 59

[GC13](typo) When RUN, there will be no space between NAME and the question-mark (?).

Page: 61

[GC14](factual) This line is formatted to a field width of 10, not 12 as output by the program.

Page: 68

[GC15](info) When RUN the cursor appears right next to the question mark (?) while waiting for the user to enter a name. If (as here) the user types a leading space before SUSAN it will be stripped when it is put into NAME$ and the second line will still appear as shown. To preserve all leading characters use the INPUT LINE statement.

Page: 80

[GC16](factual) Not true when the STEP size is negative. (See keyword STEP.)

Page: 104
[GC17](typo) Missing variable name – replace line with:

```
10 DIM Y(5,3)
```

Page: 111
[GC18](typo) Read PROCgetdatetime.

Page: 111
[GC19](typo) Wrong parenthesis in TAB statement – change to read TAB(12,10)

Page: 112
[GC20](typo) Replace full stop(.) with comma (,)

Page: 112
[GC21](typo) Change letter s to dollar ($) – line 620 should retrieve the string variable month$.

Page: 116
[GC22](typo) Read: 30 X=INSTR(A$,B$,4)

Page: 134
[GC23](typo) There must not be a space between TAB and the opening parenthesis (.

Page: 135
[GC24](factual) An OR 32 must be included in the expression. Either change the end of the line to read:

…OR C% OR 160

or match the text description by using:

…OR C% OR 128 OR 32

Page: 135
[GC25](factual) This sentence is incorrect if taken out of context; the subroutine itself will return the character at the current position of the cursor, which first has to be moved to X%, Y% by line 330.

Page: 137
[GC26](factual) MODE 0 requires 20K of memory. (HIMEM = &3000)

Page: 137
[GC27](factual) MODE 1 requires 20K of memory. (HIMEM = &3000)

Page: 137
[GC28](factual) MODE 2 requires 20K of memory. (HIMEM = &3000)

Page: 137
[GC29](info) 16000 bytes are on display at any one time; the other 384 form part of a circular buffer and are overwritten when the computer performs hardware scrolling. (HIMEM = &4000)

Page: 137
[GC30](factual) MODE 4 requires 10K of memory. (HIMEM = &5800)

Page: 137
[GC31](factual) MODE 5 requires 10K of memory. (HIMEM = &5800)

Page: 137
[GC32](info) 8000 bytes are on display at any one time; the other 192 form part of a circular buffer and are overwritten when the computer performs hardware scrolling. (HIMEM = &6000)

Page: 137
[GC33](factual) MODE 7 requires 1K of memory. 1000 characters are on display at any

one time; the other 24 form part of a circular buffer and are overwritten when the computer performs hardware scrolling. (HIMEM = &7C00)

Page: 141
[GC34](factual) Read “logical colour 1”. The foreground and background colours share the same palette and so actual colour assignments are made equally to both of them.

Page: 141
[GC35](factual) Read “logical colour 0”, reason as above.

Page: 141
[GC36](typo) Should read, “Logical colour 1 is green”.

Page: 147
[GC37](typo) For correct function insert a space to read:

```
50 PRINT TAB(X,10) ; " "
```

Page: 147
[GC38](grammar) “Altering” is meant, rather than swapping lines 30 and 50.

Page: 147
[GC39](info) Use of FOR…NEXT loops as a time delay is not recommended as this will give different delays on different hardware (due to speed and/or optimisation issues.) Use a construct such as:

```
T% = TIME : REPEAT UNTIL TIME - T% >= 15
```

Page: 150
[GC40](typo) Insert a zero to read:

```
38 GCOL0,1
```

Page: 151
[GC41](typo) The incorrect comma (,) at the end of line 30 should be deleted.

Page: 153
[GC42](typo) Replace burnttime with burntime – the former variable does not exist.

Page: 154
[GC43](typo) Missing first parameter to GCOL – line 5045 should read:

```
5045 GCOL 0, RND(4)
```

Page: 156
[GC44](typo) The third number should be 52 to produce middle C; an error of 1 would be noticeable, for example in a three-note chord.

Page: 157
[GC45](info) When AR=0 the envelope will not finish by itself. See p.219.

Page: 171
[GC46](typo) Delete the double quotes (”) as they create a string constant expression instead.

Page: 171
[GC47](info) The statement may include parameters, for example PRINT “error”, GOSUB 1200 or PROCfallback, and/or consist of multiple statements separated by colons (:).

Page: 171
[GC48](info) This symbol is used only for procedure and function names. Unlike the names of variables, these can include BASIC keywords, so PROCTOPofFORM and FNINSTR are valid (but not recommended). BASIC does not tokenise <variable name>s.

Page: 189
[GC49](typo) Should read “the least significant byte”.

Page: 204
[GC50](typo) Line 25 should read:

25 COLOUR 128+X

Page: 205
[GC51](factual) Replace both occurrences of "<string>" with "<string-var>" and "<numeric>" with "<num-var>", since these will be the formal parameters when the function is defined.

Page: 217
[GC52](info) When AR=0 the envelope will not finish by itself. See p.219.

Page: 220
[GC53](info) EOF# returns TRUE iff the next read operation would result in an error. This suits the BBC which lacks a WHILE…ENDWHILE loop and has weaker error recovery than a typical C environment. Conversely C's feof() function returns nonzero iff a 'bad' read has already taken place. So on the BBC a REPEAT…UNTIL EOF#X construct will read exactly all the records in file X.

Page: 228
[GC54](typo) Should read "the statement X=20".

Page: 230
[GC55](typo) For <str-var> read <string-var>.

Also, only a FN definition syntax is given, so here is the syntax for a FN call:

<num-var>|<string-var>=FN<variable name>[(<numeric>|<string>{,<numeric>|<string>})]

Page: 233
[GC56](info) This statement means, 'Exclusive-OR the existing pixels with logical colour 1, in all future graphics background operations (such as CLG or PLOT 3,x,y.)' The graphics foreground colour and mode-of-action are unchanged.

Page: 235
[GC57](typo) Missing line number – begin line with 2110 PRINT…

Page: 242
[GC58](info) Except when assigning to pseudo-variables. See keyword THEN for details.

Page: 245
[GC59](typo) Replace apostrophe (') with comma (,).

Page: 245
[GC60](info) Despite the incorrect ordering, each key is paired with its correct code.

Page: 245
[GC61](typo) For letter S see above letter M.

Page: 250
[GC62](info) Be aware of a serious bug in BASIC 1 INSTR. For details see p.476.

Page: 250
[GC63](typo) Delete dollar ($) – LEN is the correct name of the function.

Page: 254
[GC64](typo) Change variable name "Length" to read "length".

Page: 258
[GC65](typo) The correct syntax is:

LIST[<num-const>][,[<num-const>]]

Page: 266
[GC66](typo) Change S to dollar ($), that is:

RESTofLINE$=MID$(main$,10)

to specify a string variable.

Page: 286
[GC67](factual) FN$ is not a valid variable name since FN is a BASIC keyword. Replace with, for example, FILE$.

Page: 286
[GC68](factual) Replace the invalid variable name FN$ with the name you used in line 10, for example FILE$.

Page: 293
[GC69](typo) The space between POINT and the opening parenthesis (should be deleted.

Page: 294
[GC70](typo) Insert space between double quotes to read:

100 REPEAT PRINT" ";

Page: 295
[GC71](factual) In the illustration that follows, the commas are absent and only spaces have been used.

Page: 296
[GC72](info) In the normal output format, fractional numbers between –1 and +1 will overflow into the next field. This is because B2 limits only the significant figures of a number (not the leading zero) and the single (B1–B2 = 10–9 = 1) extra space is taken by the decimal point.

The default value of @% was chosen for best all-round results but a more robust setting will prevent overflow. In decimal format allow for three extra characters (B1–B2 >= 3, e.g. @%=&070A) but very large or small numbers will fall back to exponential format which may have six extra characters.

Page: 296
[GC73](typo) If the command line is correct then this line will read 2.25 with the 5 aligned in column 10.

Page: 296
[GC74](typo) Insert a % after the @, namely @%=&01020903.

Page: 298
[GC75](typo) The rest of the row will be correct if this value is 0.0001.

Page: 298
[GC76](typo) For both question marks (?) substitute a vertical bar (|) to indicate alternation. (info) "<string>|<numeric>" can be enclosed in square brackets [] as these elements are optional.

Page: 298
[GC77](typo) Delete hash (#) and replace with dollar ($) to read STR$.

Page: 299
[GC78](typo) Replace question mark (?) with vertical bar (|) to indicate alternation.

Page: 300
[GC79](typo) For <variable-name> read <variable name>. Insert closing parentheses)) so that both lines end …<num-var>})]

Page: 305
[GC80](typo) This should be 1000.

Page: 319
[GC81](info) Channel 1 need not make any sound at all – an amplitude of zero can be used just as well.

Page: 319
[GC82](info) If you want channel 1 to play a silent envelope to drive channel 0, then the last six numbers should be 0,0,0,-1,0,0.

Page: 343
[GC83](typo) Add a semicolon (;) to the end of the line, that is:

```
VDU 24,0;0;1279;830;
```

Page: 348
[GC84](factual) VDU 19 requires 5 extra bytes.

Page: 348
[GC85](factual) VDU 20 requires no extra bytes.

Page: 352
[GC86](factual) Both foreground and background logical colours, for both text and graphics, are reset by this code. VDU 20 does not affect the logical colours of text and graphics already on the screen.

Page: 359
[GC87](factual) The screen in MODE 7 is 25 lines long, so the statement

```
VDU 31,20,12
```

will bring the cursor nearest to the centre of the screen.

Page: 362
[GC88](typo) Insert double quotes (") so the line reads:

```
LOAD“PROG”
```

Page: 364
[GC89](typo) Insert double quotes (") so the line reads:

```
CHAIN“PROG”
```

Page: 368
[GC90](typo) X=3 will fail; use X=2.

Page: 368
[GC91](info) The load address will be printed first, followed by the execution address. The *SAVE command takes these parameters in the opposite order.

Page: 368
[GC92](typo) Missing first parameter – the command will fail. It should be shown as: *OPT0,1

Page: 369
[GC93](info) A logical 1 is represented by two 2400Hz sinusoidal cycles, and a logical 0 by one 1200Hz sinusoidal cycle. In 300 baud mode there are eight and four cycles respectively. If recording the signal to magnetic tape the frequency should change at zero volts, otherwise it should change at the peak or trough. Each byte is written with one start bit (0), eight data bits from lowest to highest, and one stop bit (1).

Page: 369
[GC94](info) On loading, the file name is stored in memory at locations &3B2 to &3BB. The end marker is written between &3B3 and &3BC inclusive.

Page: 369
[GC95](info) The rest of the header including the CRC is stored in contiguous locations from &3BE onwards. The load address is identical in every block of the file.

Page: 369
[GC96](info) All blocks except the last must contain 256 bytes of data. There are 257 possible lengths of data in the last block.

Page: 369
[GC97](info) The data CRC is omitted if there is no data. In all cases a trailing high tone immediately follows the block.

Page: 369
[GC98](info) If bit 0 of the block flag is set then the file is locked – the user is prevented from accessing the contents, and the file can only be CHAINed or *RUN.

Page: 369
[GC99](info) The initial contents of H and L are &00, &00.

Page: 369
[GC100](info) This algorithm can be derived from the PKZIP CRC as follows: the shift register is 16 bits wide, the initial value is &0000, the final value is not exclusive ORed with &FFFF, and the 'magic number' is (&0810 * 2 + 1) = &1021. PKZIP's shift register in turn is a mirror image of the one in the POSIX 'cksum' algorithm.

Page: 370
[GC101](info) *DISK is also accepted by most (if not all) Disc Filing Systems.

Page: 385
[GC102](info) Virtually all filing systems accept *. as a synonym for *CAT.

Page: 385
[GC103](info) This is the *TAPE command with a parameter; there can be a space between *TAPE and 3.

Page: 386
[GC104](factual) Interlace is on by default.

Page: 386
[GC105](info) Two further undocumented commands are *CODE and *LINE, which call code pointed to by USERV (locations &200 and &201.) The */ method of loading and running machine code programs by name, invokes a different paged ROM call from *RUN. *| is documented in this guide as beginning a comment.

Page: 388
[GC106](factual) This seems to contradict the previous pages. The shadow screen is managed by the MOS, not by BASIC; by using the VDU code to change MODE, BASIC is not informed and HIMEM does not change. The MOS almost certainly enters shadow mode, however; this will need to be checked on a model B+.

Page: 393
[GC107](factual) This line should read:

```
20 DIM registers 3
```

if it is to allocate four bytes of memory.

Page: 400
[GC108](factual) Should say "off". This is because the VDU driver is disabled, and so cannot feed the printer stream by the VDU 2 mechanism.

Page: 400
[GC109](factual) Should say "off", reason as above.

Page: 400
[GC110](factual) This range should be 8 to 11.

Page: 402
[GC111](factual) *FX6 (equivalent to *FX6,0) actually filters out ASCII character &00 (NUL). This affects the printing of graphics when the printer is enabled with *FX3,8 or similar commands. To send character &00 to the printer either use VDU 1,0 (after a VDU 2 command has been issued) or temporarily change the value of *FX6.

Page: 405

[GC112](factual) In this table the value of OSHWM is the one returned by OSBYTE 131 when the character set is imploded (on power-up or after issuing *FX20,0). After any *FX20 command OSHWM is updated and a call to OSBYTE with A=131 will give the new value of PAGE without needing to add any offset (although the user's program may be lost.) For example:

```
>A%=131:PRINT ~(USR(&FFF4) AND &FFFF00) DIV 256

      E00

>*FX20,6

>A%=131:PRINT ~(USR(&FFF4) AND &FFFF00) DIV 256

     1400

>*FX20,0

>A%=131:PRINT ~(USR(&FFF4) AND &FFFF00) DIV 256

      E00
```

Page: 406

[GC113](info) Do this before changing the *FX20 setting, reason as above.

Page: 408

[GC114](typo) Should say X=2. Using X=3 is virtually the same as X=1.

Page: 411

[GC115](typo) Delete greater than sign (>). OSBYTE 13 and 14 use the same meanings for X (and Y), namely 0=disabled.

Page: 413

[GC116](info) "This feature is invalid when called from a Second Processor." – Acorn Application Note 020, (for full reference see later comments.)

Page: 413

[GC117](info) "This feature should not normally be called from a Second Processor." – Acorn Application Note 020.

Page: 418

[GC118](info) "This call has been expanded to allow a character to be inserted into any buffer" – Acorn Application Note 020. This implies that earlier versions of the MOS allowed this call to modify only certain buffers.

Page: 419

[GC119](factual) Interlace is on at power up.

Page: 422

[GC120](info) This location accesses the Video ULA control register.

Page: 422

[GC121](info) This location accesses the Video ULA palette register.

Page: 422

[GC122](typo) Missing offset address – this register is at &64.

Page: 423

[GC123](factual) Specifically, the user 6522 data direction register.

Page: 427

[GC124](info) Acorn have also documented the following OSBYTE calls. The full reference is: Acorn Computers Limited, Support Group Application Note 020: "*FX/OSBYTE calls not documented in the BBC Micro User Guide"; Issue 1, 8th July 1992.

Note: How to use the Read/Write calls is detailed on page 424 (page 438 of the Model B's User Guide.) In short, new value = (old value AND Y) EOR X; the old value is returned in X.

A=141: select the sideways ROM filing system, exactly equivalent to *ROM.

A=153: Insert character X into input buffer Y (0=keyboard, 1=RS423 input) generating an ESCAPE condition if X is the interrupt character.

A=156: Read/Write the 6850 control register. This allows the user to change parity and data word length settings. The following values are derived from the Motorola MC6850 data sheet, p. 8:

A=156, X=0, Y=227: 7 data bits, even parity, 2 stop bits (7,E,2).

A=156, X=4, Y=227: 7 data bits, odd parity, 2 stop bits (7,O,2).

A=156, X=8, Y=227: 7 data bits, even parity, 1 stop bit (7,E,1).

A=156, X=12, Y=227: 7 data bits, odd parity, 1 stop bit (7,O,1).

A=156, X=16, Y=227: 8 data bits, 2 stop bits (8,N,2).

A=156, X=20, Y=227: 8 data bits, 1 stop bit (8,N,1) (this is the default setting on the BBC Micro).

A=156, X=24, Y=227: 8 data bits, even parity, 1 stop bit (8,E,1).

A=156, X=28, Y=227: 8 data bits, odd parity, 1 stop bit (8,O,1).

A=241: Read/Write the *FX1 value.

A=245: Read/Write the *FX5 value.

A=246: Read/Write the *FX6 value.

A=252: Read the ROM socket number of the current language (not necessarily meaningful on a second processor.)

A=254: Read/Write the amount of available RAM on the I/O processor (128 = 32K, 64 = 16K), can be called from a second processor.

Page: 430
[GC125](factual) The value written to the program counter is the &3500 mentioned in line 100, and never P% except by explicit instruction (USR(P%) or CALL P%). The MOS is not aware of the variable P%, and machine code is not automatically called when it is assembled.

Page: 433
[GC126](factual) This should read, "the most significant byte".

Page: 433
[GC127](info) A bug in the 6502 microprocessor affects instructions of the form JMP (&xxFF). For example JMP (&23FF) will fetch the new address from locations &23FF and &2300, not from &23FF and &2400.

Page: 433
[GC128](factual) Only indirect JMP is recognised.

Page: 433
[GC129](info) Absolute indirect addressing is not affected by 'wrap around', that is the address being accessed need not be in the same page as the base address.

Page: 436
[GC130](typo) Change this line to read:

```
10 OSWRCH=&FFEE
```

Page: 437
[GC131](factual) This should read “the first two iterations, or turns, of the machine code loop” to avoid confusion with the discussion overleaf.

Page: 437
[GC132](factual) 101 bytes are reserved. See keyword DIM.

Page: 437
[GC133](factual) 29 memory locations are reserved. See keyword DIM.

Page: 443
[GC134](info) OSWRSC is not available in OS 1.20.

Page: 445
[GC135](info) At the end of a file, a call to OSBGET sets an internal ‘warning given’ flag and returns with C=1 as shown. When BASIC calls OSBGET and finds C=1, it calls OSBGET a second time to produce an error.

Page: 448
[GC136](typo) This byte is 0A, the ASCII line feed character.

Page: 458
[GC137](typo) Missing ampersand – &FE

Page: 464
[GC138](factual) This error only occurs if the keyword NEXT is followed by one or more variable names.

Page: 467
[GC139](factual) Replace with FOR.

Page: 469
[GC140](info) Before this line insert: STOP, 0, A STOP statement had been encountered.

Page: 501
[GC141](factual) VDU 19 requires 5 extra bytes.

Page: 501
[GC142](factual) VDU 20 requires no extra bytes.

Page: 506
[GC143](typo) Part of address missing – this cell should read 218.