BBC Microcomputer System

Technical Specification - Issue 3
September 1981

The present specification does not include details of various parts of the BBC Microcomputer System which are still being developed. These include:

The teletext adaptor
The Prestel adaptor
The single-drive 100 Kbyte disk store
The dual-drive 800 Kbyte disk store
The 6502 second processor expansion
The Z80 second processor expansion
The CP/M-compatible disk system.

It is anticipated that the specification covering these items will be available by 31st March 1982. If you would like to be sent a copy of this specification when it becomes available, please send a large s.a.e. (324 x 229mm and 20p postage) to:

Technical Specification (2),
BBC Microcomputer System,
P.O. Box 7,
London,
W3 6XJ

September 1981

The BBC Microcomputer System

PART I - FIRMWARE SPECIFICATION

Section 1 - The BASIC Language Interpreter

This document specifies the firmware characteristics of the Microcomputer Unit of the BBC Microcomputer System, as agreed between the British Broadcasting Corporation and Acorn Computers Limited. It forms part of the overall specification, firmware and hardware, of the BBC Microcomputer System.

Edited by: R.T.Russell, BBC Engineering Designs Department.

Listed below are commands, statements and functions which form the "common core".

ABS
AND
ASC
ATN
AUTO
CALL
CHAIN
CHR$
CLEAR
CLOSE
CLS
COLOUR
COS
DATA
DEF FN
DELETE
DIM
DRAW
ELSE
END
EOF
EOR
ERL
ERR
EXP
FOR ... TO ... STEP ... NEXT
GET
GOTO
GOSUB
IF ... THEN ... ELSE
INKEY
INPUT
INPUT LINE
INPUT#
INSTR
INT
LEFT$
LEN
LET
LIST
LN
LOAD
LOG
MID$
MOD
MOVE
NEW
NEXT
NOT

ON ERROR
ON X GOTO
ON X GOSUB
OPENIN
OPENOUT
OR
PI
POINT
POS
PRINT
PRINT#
READ
REM
RENUMBER
RESTORE
RETURN
RIGHT$
RND
RUN
SAVE
SGN
SIN
SPC
SQR
STEP
STOP
STR$
STRING$
TAB
TAN
THEN
TO
TRACE
USR
VAL
WIDTH

Extensions.

These extensions should be avoided in simple programs that are intended to be used on a variety of machines.

ACS
ADVAL
ASN
BGET
BPUT
CLG
COUNT
DEF PROC
DEG
DIV
ENDPROC
ENVELOPE
EVAL
EXT
FALSE
GCOL
HIMEM
LISTO
LOCAL
LOMEM
MODE
OLD
OPT
PAGE
PLOT
PTR
RAD
REPEAT ... UNTIL
REPORT
SOUND
TIME
TOP
TRUE
UNTIL
VDU
VPOS

The BASIC interpreter works through one of three filing systems: a cassette filing system (CFS), a disc filing system (DFS) or a network filing system (NFS). The user can easily move between filing systems and can pass complicated commands direct to the filing system while in BASIC or any other language or environment.

The BASIC interpreter contains an Assembler which accepts standard 6502 mnemonics.

Variables:

Variable names may be of unlimited length and all characters are significant. Variable names must start with a letter. They can only contain the characters A..Z, a..z, 0..9 and underline. Embedded keywords are allowed. Upper and lower case variables of the same name are different.

Keywords are recognised before anything else (e.g. both DEG and ASN in DEGASN are recognised, but neither in ADEGASN). Pseudo variables (PI, LOMEM, HIMEM, PAGE, TIME etc.) act as variables in that if PI is a (pseudo-) variable then it does not affect PILE (or if A is a variable, then AB can be). Note that PI%, PI$ etc. are not allowed.

The following types of variable are allowed:

A	real numeric
A%	integer numeric
A$	string

The variables A%..Z% are regarded as special in that they are not cleared by the commands or statements RUN, CHAIN and CLEAR. In addition A%, C%, X% and Y% have special uses in CALL and USR routines and P% has a special meaning in the assembler (it is the program counter). The special variable @% controls numeric print formatting.
The variables @%..Z% are called "static variables", all other variables are called "dynamic variables".

Real variables have a range of approximately +-1E-38 to +-1E38 and numeric functions evaluate to 9 significant figure accuracy. Internally every real number is stored in 40 bits.

Integer variables are stored in 32 bits and have a range of about +-2E9.

String variables may contain up to 255 characters.

All arrays must be dimensioned before use.
All statements can also be used as direct commands.

ABS A function giving the absolute value of its argument.

```
X=ABS(DEFICIT)
LENGTH=ABS(X1-X2)
B=ABSA
```

ACS A function giving the arc cosine of its argument in radians.

```
X=ACS(Y)
```

ADVAL A function which returns the last known value of ADC channel n.

```
X=ADVAL(INTENSITY)
Y=ADVALN
Z=ADVAL2
```

AND The operation of integer bitwise logical AND between two items.

```
IF A=2 AND B=3 THEN 110
A=X AND 3
```

ASC A function returning the ASCII character value of the first character of the argument string. If the string is null then -1 will be returned.

```
X=ASC(A$)
X=ASC("HELLO")
X=ASC"e"
X=ASC(MID$(A$,Z))
```

ASN A function giving the arc sine of its argument in radians.

```
X=ASN(Y)
```

ATN A function giving the arc tangent of its argument in radians.

```
X=ATN(Y)
```

AUTO A command allowing the user to enter lines without first typing in the number of the line. AUTO mode is left with <escape>. Step size range is 1 to 255.

```
AUTO          offers line numbers 10, 20, 30 ....
AUTO 100      starts at 100 with step 10
AUTO 100,1    starts at 100 with step 1
AUTO ,2       starts at 10 with step 2
```

BGET A function which gets a byte from the file whose channel number is its argument.

```
E=BGET #N
```

BPUT A statement which puts a byte to the file whose channel number is the first argument. The second argument's least significant byte is sent.

```
BPUT #E,32
BPUT #STAFF_FILE,A/256
```

CALL A statement to call a piece of machine code. CALL initialises a vector at address hex 0600 in store (actually it's the string input buffer) to contain the number of parameters and pointers to each parameter with an attached type. The processor's A, X, and Y registers and the C carry flag are initialised to the least significant bytes of A%, X% and Y% and the least significant bit of C% respectively, see also USR.
Parameter types are:

```
  0: byte (8 bits) e.g.  ?A%
  4: word (32 bits) e.g.  !A% or A%
  5: real (40 bits) e.g.  A
128: fixed string e.g.  $A% terminated by CHR$13
129: movable string e.g.  A$
```

On entry to the subroutine the parameter block contains the following values:

```
Number of parameters - 1 byte
Parameter type - 1 byte
Parameter address - 2 bytes
Parameter type     ) repeated as often
Parameter address )   as necessary
```

The value given is the absolute address of the item except for type 129 where it is the address of a parameter block containing start address, maximum length and current length.

```
CALL MULDIV,A,B,C,D
CALL &FFE3
CALL 12340,A$,M,J$
```

CHAIN A statement which will load and run the program whose name is specified in the argument. All variables except @% to Z% are CLEARed.

```
CHAIN "GAME1"
CHAIN A$
CHAIN("jim")
```

CHR$ A string function whose value is a string of length 1 containing the ASCII character specified by the least significant byte of the numeric argument.

```
A$=CHR$(72)
B$=CHR$12
C$=CHR$(A/200)
```

CLEAR A statement which clears all the dynamically declared variables, including strings. The variables @% to Z% are static: CLEAR will not alter these at all.

CLOSE A statement used to close a sequential file. CLOSE #0 will close all sequential files.

```
CLOSE #E
CLOSE #0
```

CLG A statement which is equivalent to PRINT CHR$16;. It clears the graphics area on the vdu to the current graphics background colour and moves the graphics pointer to 0,0 (bottom left).

CLS A statement which is equivalent to PRINT CHR$12;. It clears the text area on the vdu to the current text background colour and moves the cursor to 0,0 (top left).

COLOUR A statement which sets the text foreground and background colours of the "soft" displays (modes 0-6). If the argument is >127 the text background colour is affected, otherwise the text foreground colour is affected.
The colours are as follows with the default palette:

0	128	black
1	129	red
2	130	green
3	131	yellow
4	132	blue
5	133	magenta
6	134	cyan
7	135	white
8	136	flashing black
9	137	flashing red
10	138	flashing green
11	139	flashing yellow
12	140	flashing blue
13	141	flashing magenta
14	142	flashing cyan
15	143	flashing white

```
COLOUR 3
COLOUR N+128
```

COS A function giving the cosine of its radian argument.

```
X=COS(Y)
```

COUNT A function returning the number of characters printed since the last newline (this is not necessarily the same as POS).

```
A=COUNT
```

DATA A program object which must precede all lists of data for READ. The string values may be quoted or unquoted as for INPUT, the numeric values may include calculation so long as there are no reserved words.

```
DATA 10,2,HELLO,"THIS IS A COMMA,"
DATA 0,1,2,3.4,jim,"  print",PRINT
```

DEG A function which converts radians to degrees.

```
X=DEG(PI/2)
X=DEG(ASN(1))
```

DEF A program object which must precede declaration of a user function FN or a user procedure PROC. If encountered at execution then the rest of the line is ignored so that single line definitions can be put anywhere in the program. Multi line definitions must not be executed; it is recommended that they are placed at the end of the main program text. There is no speed advantage gained in placing them at the start of the program.

```
DEF FNMEAN ....
DEF PROCJIM ....
```

DELETE A command which deletes a group of lines from the program. Both start and end lines of the group will be deleted.

```
DELETE 10,15    delete 10,11,12,13,14,15
DELETE 0,20     delete up to 20
DELETE 20,32767 delete all lines after 19
```

DIM — A statement which declares arrays. Arrays must be predeclared before use. String arrays may be multi dimensional. After DIM all elements in the array are 0/null. Arrays may not be redimensioned. DIM can also be used to dimension integers to point at an area of memory which the interpreter will not then use. This can be used for positioned strings, data structures and assembler etc. The last item in the example sets A% to an area of memory with bytes A%?0 to A%?67 free for use by the program. The subscript base is 0 so DIM X(12) defines an array of 13 elements.

```
DIM A(2),Ab(2,3),A$(2,3,4),A%(3,4,5,6),A% 67
```

DIV — A binary operation giving the integer quotient of two items.

```
X=A DIV B
```

DRAW — Draw a line to the specified position in the current graphics foreground colour. This statement is exactly equivalent to PLOT5.

```
DRAW X,Y
```

ELSE — A statement delimiter which behaves as follows:
When encountered as a delimiter the rest of the line is ignored. If in an IF statement the boolean is false the statements after ELSE will be executed. This makes the following work:

```
IF A=B THEN B=C ELSE B=D
IF A=B THEN B=C:PRINT"WWWW" ELSE B=D:PRINT"QQQQ"
IF A=B THEN B=C ELSE IF A=C THEN.............
```

But this does not work:

```
IF A=B THEN B=C:IF C=D THEN B=Q ELSE B=P ELSE B=W
```

ELSE also traps exceptions in ON.

END — A statement causing the interpreter to return to direct mode. There can be any number (>=0) of END statements anywhere in a program.

ENDPROC — A statement denoting the end of a procedure. All LOCAL variables and the dummy arguments are restored at ENDPROC and the program returns to the statement after the calling statement.

ENVELOPE — A statement taking 14 parameters which are bytes. The bytes are passed to the Machine Operating System to control the sound generator.

```
ENVELOPE 0,1,2,3,4,5,6,7,8,9,10,11,12,13
```

EOF

A function which will return -1 (TRUE) if the file whose channel number is the argument is at its end.

```
IF EOF#B THEN ....
```

EOR

The operation of bitwise integer logical exclusive-or between two items as 32 bit integers.

```
IF A=2 EOR B=3 THEN 110
X = B EOR 4
```

ERL

A function returning the line number of the line where the last error occurred.

```
X=ERL
```

ERR

A function returning the error code number of the last error which occurred.

```
X=ERR
```

EVAL

A function which applies the interpreter's expression evaluation program to the characters held in the argument string. An easy way to pass a function into a program from a user input.

```
X=EVAL"3"
X=EVAL("X^Q+Y^P")
X=EVAL"A$+B$"
X=EVAL(A$)
```

EXP

A function returning e to the power of the argument.

```
Y=EXP(Z)
```

EXT

A function which returns the length, in bytes, of the file whose channel number is the argument.

```
L=EXT#E
```

FALSE

A function returning 0, i.e. false.

FN A reserved word used at the start of all user declared functions. A function may be defined with any number of arguments of any type, and may return (using =) a string or numeric quantity. A function definition is terminated by '=' used in the statement position. The arguments are passed by value.

```
DEF FNMEAN(Q1,Q2,Q3,Q4)=(Q1+Q2+Q3+Q4)/4
DEF FNFACTORIAL(N) IF N<2 THEN =1 ELSE
=FNFACTORIAL(N-1)*N
DEF FNRIGHT2(A$)=RIGHT$(A$,2)
DEF FNREVERSE(A$) LOCALB$,Z%
FOR Z%=1TOLEN(A$):B$=MID$(A$,Z%,1)+B$:NEXT:=B$
```

FOR A statement initialising a FOR NEXT loop. The loop is executed at least once. GOTO inside a loop is in extended range (which means you can't exit the loop with GOTO). Any numeric assignable item may be used as the control variable. Integer control variables are three times quicker than real variables at the NEXT statement and also much faster when indexing an array.

```
FOR TEMPERATURE%=0 TO 9
FOR A(2,3,1)=0 TO 9
FOR X=1 TO 16 STEP 0.3: PRINT X: NEXT X
```

GCOL A statement to set the graphics foreground and background colours and actions. The first value is the action (0 = store, 1= OR, 2= AND, 3= EOR, 4= invert) and the second is the colour, if >127 then background, else foreground (see COLOUR).

```
GCOL 0,RED
GCOL 2,129
```

GET A function and compatible string function that reads the next character from the input stream, usually the keyboard (it waits for the character). See INKEY.

```
N=GET
N$=GET$
```

GOTO A statement which will go to a line with constant number or calculated value. If a calculated value is used, the program should not be RENUMBERed.

```
GOTO 100
GOTO (X*10)
```

GOSUB As GOTO but allows RETURN. Maximum depth is 25 nested GOSUBs.

```
GOSUB 400
GOSUB (4*ANSWER+6)
```

HIMEM — A pseudo-variable which sets and gives the maximum address used by the interpreter. The user is cautioned to use this facility with care!

```
HIMEM=HIMEM-40
```

IF — A statement.

```
IF LENGTH=5 THEN 110
IF A<C OR A>D GOTO 110
IF A>C AND C>=D THEN GOTO 110 ELSE PRINT "STRUCTURED BASIC"
IF A<Q PRINT"I THINK IT IS LESS"
IF A>Q THEN PRINT"I THINK IT IS GREATER"
IF A>=Q THEN PRINT"GREATER OR EQUAL":END
```

INKEY — A function and compatible string function that will do a GET/GET$, waiting for a maximum of n clock ticks (usually 10ms each). If no key is pressed in the time limit INKEY will return -1 and INKEY$ will return a null string.

```
N=INKEY(0)
N$=INKEY$(100)
```

INPUT — A statement to input values from the current input stream (usually keyboard).

```
INPUT A,B,C,D$,"WHO ARE YOU",W$,"NAME"R$
```

If items are not immediately preceded by a printable string (even if null) then a "?" will be printed as a prompt. Items A, B, C, D$ in the above example can have their answers returned on one to four lines, separate items being separated by commas. Extra items will be ignored. Then WHO ARE YOU? is printed (the question mark comes from the comma) and W$ is input, then NAME is printed and R$ is input. The statement INPUT A is exactly equivalent to INPUT A$:A = VAL(A$). Leading spaces will be removed from input, but not trailing spaces. Strings in "ed form are taken as they are, with a possible error occurring for a missing closing quote.

INPUT LINE — A statement of identical syntax to INPUT which uses a new line for each item to be input. The item input is taken as is, including commas, quotes and leading spaces.

```
INPUT LINE A$
```

INPUT# — A statement which reads data in internal format from a file and puts them in the specified variables.

```
INPUT #E,A,B,C,D$,E$,F$
```

INSTR — A function which returns the position of a sub-string within a string, optionally starting the search at a specified place in the string. If the sub-string is not found 0 is returned.

```
X=INSTR(A$,B$)
Y=INSTR(A$,B$,Z%) :REM START SEARCH AT POSITION Z%
```

INT — A function truncating a real number to the lower integer i.e INT(99.8)=99, INT(-12)=-12, INT(-12.1)=-13.

```
X=INT(Y)
```

LEFT$ — A string function which returns the left n characters of the string. If there are insufficient characters in the string then all are returned.

```
A$=LEFT$(A$,2)
A$=LEFT$(RIGHT$(A$,3),2)
```

LEN — A function which returns the length of the argument string.

```
X=LEN(A$)
X=LEN"fred"
X=LENA$
X=LEN(A$+B$)
```

LET — optional assignment statement.

LIST — A command which causes lines of the current program to be listed out with the automatic formatting options specified by LISTO.

```
LIST            lists the entire program
LIST ,111       lists up to line 111
LIST 111,       lists from line 111 to the end
LIST 111,222    lists lines 111 to 222 inclusive
LIST 100        lists line 100 only
```

Note that L. is a convenient abbreviation for LIST.

LISTO — A command which sets options for formatting a LISTed program. LISTO takes an integer 0 to 7, the bits of which control strings a, b and c:
String a is printed out just after every line number.
String b is printed out n times just after a where n is the level of nested FOR NEXT loops.
String c is printed out m times just after b where m is the level of nested REPEAT UNTIL loops.
The strings are " " (single space), " " (double space), " " (double space).
Note that n and m are set according to the lines actually listed out so far. It is useful to set LISTO 0 when doing a lot of cursor editing, this is the default value.

```
LISTO 7
```

LN — A function giving the natural logarithm of its argument.

```
X=LN(Y)
```

LOAD — A command which loads a new program from a file and CLEARs the variables of the old program.

```
LOAD "PROG1"
LOAD A$
```

LOCAL — A statement to declare variables for local use inside a function (FN) or procedure (PROC). LOCAL saves the values of its arguments in such a way that they will be restored at = or ENDPROC. See FN for an example.

```
LOCAL A$,X,Y%
```

LOG — A function giving the base-10 logarithm of its argument.

```
X = LOG(Y)
```

LOMEM — A pseudo-variable which controls where in memory the dynamic data structures are to be placed. The default is TOP, the first free address after the end of the program. Changing LOMEM causes loss of all dynamic variables.

```
LOMEM=LOMEM+100
```

MID$ A string function which returns N characters of the string starting from character M. If N is not present or if there are insufficient characters in the string then all from M onwards are returned.

```
C$=MID$(A$,M,N)
C$=MID$(A$,Z)
```

MOD A binary operation giving the signed remainder of the integer division. MOD is defined such that A MOD B = A - (A DIV B) * B

```
X=A MOD B
```

MODE A statement which will select the specified vdu screen mode (0-7). The space provided by a 20K screen changing to a 1K screen is automatically recovered by MODE. MODE cannot be used inside a PROC or FN (but VDU22,n could). The display modes are as follows:

0	640x256 2-colour graphics and 80x32 text	(20K)
1	320x256 4-colour graphics and 40x32 text	(20K)
2	160x256 16-colour graphics and 20x32 text	(20K)
3	80x25 2-colour text	(16K)
4	320x256 2-colour graphics and 40x32 text	(10K)
5	160x256 4-colour graphics and 20x32 text	(10K)
6	40x25 2-colour text	(8K)
7	40x25 teletext mode	(1K)

```
MODE 0
MODE A
```

MOVE A statement which moves the graphics pointer to the specified position. It is exactly equivalent to PLOT4.

```
MOVE X,Y
```

NEW A command which initialises the interpreter for a new program to be typed in. The old program may be recovered with the OLD command provided no errors have occurred and no lines have been typed in.

NEXT The statement delimiting FOR NEXT loops. NEXT takes an optional control variable; if present then FOR NEXT loops may be 'popped' in an attempt to match to the correct FOR statement.

```
NEXT
NEXT J,K
```

NOT This is a high priority unary operator (the same priority as unary -).

```
IF NOT(RATE>5 AND TIME<100) THEN .....
```

OLD A command which undoes the effect of NEW provided no new lines have been typed in, and no variables have been created.

ON A statement controlling a multi-way switch or error trapping. The values in the list are skipped without calculation but it will get confused by ASC"," appearing.

```
ON  A+2 GOTO 100,200,300,120,300:ELSE PRINT"Illegal"
ON B-46 GOSUB 100,200,(C/200):ELSE PRINT"What?"
ON ERROR PRINT"Suicide":END
ON ERROR GOTO 100
ON ERROR OFF:  REM sets back normal error handler
```

OPENIN A function which returns the channel number of the file. The file is opened for input and updating. If it does not exist 0 is returned. OPENIN is an abreviation of OPEN FOR INPUT. With the CFS input only is allowed, with the DFS or NFS input and output are allowed (random access).

```
X=OPENIN"jim"
X=OPENINA$
X=OPENIN(A$)
X=OPENIN("FILE1")
```

Example of reading in N(10) and N$(10), a top ten array:

```
A=OPENIN"TOPTEN"
IFA=0 PRINT"No TOPTEN data file":GOTO100
FORZ=1TO10:INPUT#A,N(Z),N$(Z):NEXT:CLOSE#A
```

Example of writing out the top ten array:

```
A=OPENOUT"TOPTEN"
FORZ=1TO10:PRINT#A,N(Z),N$(Z):NEXT:CLOSE#A
```

Example of reading the bytes in TOPTEN backwards:

```
A=OPENIN"TOPTEN"
FORZ=EXT#A-1TO0STEP-1:PTR#A=Z:PRINTBGET#A;:NEXT:CLOSE#A
```

OPENOUT A function which returns the channel number of a file with optional creation of the file. If a file of that name is present it is used, otherwise one is created. The file is opened for output (and updating of already output material in DFS and NFS). OPENOUT is an abreviation of OPEN FOR OUTPUT.

```
X=OPENOUT(A$)
X=OPENOUT("DATAFILE")
```

OPT An assembler pseudo operation controlling the method of assembly. OPT is followed by an expression with the following meanings:-

Value	Action
0	assembler errors suppressed; no listing
1	assembler errors suppressed; listing
2	assembler errors occur; no listing
3	assembler errors occur; listing

Where the possible assembler errors are branch out of range and unknown label. The default OPT is 3.

OR The operation of bitwise integer logical OR between two items:

```
IF A=2 OR B=3 THEN 110
X=B OR 4
```

PAGE A pseudo-variable controlling the starting address of the current text area. With care several programs can be left around in RAM memory without the need for saving them. PAGE is always an integer multiple of 256.

```
PAGE=6*256
PAGE=PAGE+512
```

PI A function returning 3.14159265.

```
X=PI
```

PLOT A statement controlling most of the graphics. The first argument controls whether points or lines will be drawn, how pixels are put on the screen, and whether the coordinates (the other arguments) are relative or absolute.

Currently assigned values:

Bits zero and one
- 00 No change in memory while plotting (e.g. MOVE)
- 01 Plot in graphics foreground colour
- 10 Plot INVERT
- 11 Plot in graphics background colour

Bit 2
- 0 Plot coordinates are relative
- 1 Plot coordinates are absolute

Bits 3 to 6

(a) Bit 6 clear: lines and curves
- Bit 3 clear - plot both endpoints
  set - plot first endpoint twice
- Bit 4 clear - plot continuous
  set - plot dotted
- Bit 5 clear - plot line
  set - plot curve ***FUTURE EXPANSION***

(b) Bit 6 set: plot special objects
Modes &40 to &47 plot point
Modes &50 to &57 plot and fill triangle
Example PLOT&41,X,Y plots a point at X,Y
PLOT K,X,Y

POINT A function returning the colour of the point on the screen or -1 if off screen.
PRINT POINT(100,100)

POS A function returning the horizontal position of the cursor on the screen; left hand column is 0. See COUNT for printer compatible version.
X=POS

PRINT A statement. PRINT has a pair of local variables PRINTS and PRINTF (print size and print flag) which can be set or unset by various delimiters. The variable @% controls numeric print format and spacing of field output by PRINT, the bytes of @% from lsb upwards will be referred to as f1, f2, f3 and f4. For a description of the action of f4 see STR$. f3 selects the numeric format mode:
0 general (G) mode
1 exponent (E) mode
2 fixed (F) mode
The initial mode is G mode.
f2 controls the number of digits in a mode. If it is out of range then 9 is assumed. In G mode it specifies the maximum number of digits to be printed between 1 and 9. In E mode it specifies the exact number of digits to be printed between 1 and 9. In F mode it specifies the number of digits to follow the decimal point, between 0 and 9.
E mode will print an optional - sign, one digit, a decimal point, f2-1 digits, an E and an exponent field in 3 characters, padded with trailng spaces if necessary. The G mode will print integral values as integers, numbers in the range 0.1 to 1 as 0.1 etc, numbers less than .1 or greater than 1Ef2 with an exponent of as few characters as possible. F mode will print numbers with f2 digits after the "." unless the total number would then have more than nine digits in which case G mode is used.

Examples (all shown right justified)

number	G2	G9	F2	E2
.1	0.1	0.1	0.10	1.0E-1
.01	1E-2	1E-2	0.01	1.0E-2
.001	1E-3	1E-3	0.00	1.0E-3
.005	5E-3	5E-3	0.01	5.0E-3
1	1	1	1.00	1.0E0
10	10	10	10.00	1.0E1
100	1E2	100	100.00	1.0E2

Initially, and every time a "," is encountered, PRINTS is set to f1. Initially and every time either "," or ";" is encountered, PRINTF is set to false. When a "~" is encountered PRINTF is set to true. When a "'" is encountered a new line is generated. When a "," is encountered, then 0 or more spaces are output so that the current print position is a multiple of f1. When a ";" is encountered PRINTS is set to zero. When a numeric value is to be printed, it is printed in hexadecimal notation if PRINTF is true, and in decimal notation otherwise. A numeric value is printed right justified in a field of size PRINTS; if there is not enough space in the field (e.g. PRINTS is 0) then it is printed in the minimum number of characters possible (trailing spaces in E mode are printed). A string value is printed with no extra characters at all. If the last non-space character in the PRINT statement is not a ";" then a new line will be generated.

Examples	Result
@%=&00090A	
PRINT 1,2	1 2
PRINT10,200	10 200
PRINT10;200	10200
PRINT"Answer ";A	Answer 42
PRINT"Answer "A	Answer 42
PRINT"Answer ",A	Answer 42
PRINT"Hello","Hello"	Hello Hello
PRINT"Hello";"Hello"	HelloHello
PRINT1'20	1 20

PRINT# A statement which writes the internal form of a value out to a sequential file. All numeric constants are written as five bytes of binary real data, all strings are written as the bytes in the string plus a carriage return.

```
PRINT#E,A,B,C,D$,E$,F$
```

PROC — A reserved word used at the start of all user declared numeric procedures. Any number of parameters, including zero, may be passed; they are passed by value. The procedure does not have to be declared before it is called.

```
INPUT"Number of discs "F
PROCHANOI(F,1,2,3)
END
DEF PROCHANOI(A,B,C,D) IFA=0 ENDPROC
PROCHANOI(A-1,B,D,C)
PRINT"Move disk ";A" from pile ";B" to pile ";C
PROCHANOI(A-1,C,B,D)
ENDPROC
```

PTR — A pair of statement and function which allow the programmer to select the next byte to be transferred to/from a file and thus enables random access.

```
PTR#A=PTR#A-20
```

RAD — A function which converts degrees to radians.

```
X=RAD(Y)
X=SINRAD(90)
```

READ — A statement which will assign to variables values read from the DATA statements in the program. Strings must be enclosed in double quotes if they have leading spaces or contain commas.

```
READ C,D,A$
```

REM — A statement that causes the rest of the line to be ignored.

RENUMBER — A command which will renumber the lines and correct the cross references inside a program. Options as for AUTO. RENUMBER produces messages when a cross reference fails.

```
RENUMBER
RENUMBER 1000
RENUMBER 1000,5
```

REPEAT A statement which is the starting point of a REPEAT...UNTIL loop. These loops may be nested up to a maximum depth of 15. Leaving a loop with GOTO brings you into Extended Range, the loop can be reentered. UNTIL TRUE will 'pop' the stack. A single REPEAT may have more than one UNTIL.

```
REPEAT A=A+1:UNTILA>B
REPEAT
X=X+10
PRINT "What do you think of it so far?"
UNTIL X>45
```

REPORT A statement which prints out a newline followed by the error string associated with the last error which occurred.

RESTORE RESTORE can be used at any time in a program to set the place where DATA comes from. The optional parameter for RESTORE can specify a calculated line number.

```
RESTORE
RESTORE 100
RESTORE (10*A+20)
```

RETURN A statement causing a RETURN to the statement after the most recent GOSUB statement.

RIGHT$ A string function which returns the right n characters of the string. If there are insufficient characters in the string then all are returned.

```
A$=RIGHT$(A$,2)
A$=RIGHT$(LEFT$(A$,3),2)
```

RND A function with optional parameter. RND(1) returns a real number in the range 0.0 to .99999999. RND returns a random integer 0 - &FFFFFFFF. RND(n) returns an integer in the range 1 to n. If n is negative the pseudo random sequence generator is set to a number based on n and n is returned. If n is 0 the last RND(1) random number is returned.

```
X=RND(1)
X%=RND
N=RND(6)
```

RUN Start execution of the program. RUN is a statement and so programs can reexecute themselves. All variables except @% to Z% are CLEARed by RUN.

SAVE — A statement which saves the current program area to a file.

```
SAVE "FRED"
SAVE A$
```

SGN — A function returning -1 for negative argument, 0 for 0 argument and +1 for positive argument.

```
X=SIN(Y)
```

SIN — A function giving the sine of its radian argument.

```
X=SIN(Y)
```

SOUND — A statement taking 4 parameters, considered by BASIC to be 16 bit words. A provisional meaning is channel, envelope, frequency (or period), duration. The values are passed straight to the Machine Operating System.

```
SOUND 1,0,MIDDLE_C,QUAVER
```

SPC — A function which can only be used in PRINT or INPUT. SPC outputs n MOD 256 spaces where n is the argument.

```
PRINT DATE;SPC(6);SALARY
```

SQR — A function returning the square root of its argument.

```
X=SQR(Y)
```

STEP — Part of the FOR statement, this optional section specifies step sizes other than 1.

STOP — Syntactically identical to END, STOP also prints a message to the effect that the program has stopped.

STR$ — A string function which returns the string form of the numeric argument as it would have been printed. The msb of @%, if non zero, lets STR$ use the current @% description for printing numbers, if zero (the initial value) then G9 format (see PRINT) is used.

```
A$=STR$(PI)
```

STRING$ — A function returning N concatenations of a string.

```
A$=STRING$(N,"hello")
B$=STRING$(10,"-*-")
C$=STRING$(Z%,S$)
```

TAB A function available in PRINT or INPUT. TAB(X) will attempt to make COUNT equal to X by printing a newline if required plus some spaces. TAB(X,Y) will perform a cursor move on the VDU screen to character cell X,Y if possible. The leftmost column is column 0.

```
PRINT TAB(10);A$
PRINT TAB(X,Y);B$
```

TAN A function giving the tangent of its radian argument.

```
X = TAN(Y)
```

THEN Part of the IF... THEN ... ELSE statement.

TIME A pseudo-variable which reads and sets the lower four bytes of the computational elapsed time clock.

```
X=TIME
TIME=100
```

TOP A function which returns the value of the first free location after the end of the current text. Thus the length of the program is given by TOP-PAGE.

```
PRINT TOP-PAGE
```

TO Part of the FOR ... TO ... STEP statement.

TRACE TRACE ON causes the interpreter to print executed line numbers when it encounters them. TRACE X sets a limit on the size of line numbers which may be printed out, only those less than X will appear. Thus a careful user with all subroutines at the end of the main program can prevent traced output of subroutine calls. TRACE OFF turns trace off. Note that the interpreter does not execute line numbers very often:-

```
  Program                TRACE output
     10 FORZ=0TO100
     20 Q=Q*Z:NEXT       [   10] [   20] [   30]
     30 END
```

But things would be different were NEXT on line 25, TRACE output would be [10] [20] [25] [25] [25] [25] [25] [25] [25] [25]

TRUE A function having a true value (-1).

UNTIL A part of the REPEAT ... UNTIL structure.

USR A function allowing machine code to return a value directly for things which do not require the expense of CALL. USR calls the machine code subroutine whose address is its argument with the processor's A, X and Y registers and the C carry flag initialised as in CALL and returns a 32 bit integer composed of PYXA msb to lsb.

```
X=USR(LIFT_DOWN)
```

VAL A function which converts a character string representing a number into numeric form. Unless the argument is a unary signed constant VAL returns 0.

```
X=VAL(A$)
```

VDU A statement which takes a list of numeric agruments and sends them to OSWRCH. A word can be sent if the value is followed by a ";". The bytes sent DO NOT contribute to the value of COUNT, but may well change POS and VPOS (e.g. VDU 30 would home the cursor).

```
VDU28,0,31,32,0:REM define window
VDU27,0;0;1280;1024;
```

VPOS A function returning the vertical cursor position. The top of the screen is line 0.

```
X=VPOS
```

WIDTH A statement controlling output overall field width. The initial value is WIDTH 0 when the interpreter will not attempt to control the overall field width. WIDTH n will cause the interpreter to force a newline after n MOD 256 characters have been printed.

```
WIDTH 60
```

Operators and special symbols.

!	A unary and binary operator giving 32 bit indirection.
?	A unary and binary operator giving 8 bit indirection.
"	A delimiting character in strings. Strings always have an even number of " in them. " may be introduced into a string by the escape convention "".
#	Indicates "immediate" operation in assembler; precedes reference to a file channel number (and is not optional).
$	A character indicating that the object has something to do with a string. The syntax $<expression> may be used to position a string anywhere in memory, overriding the interpreter's space allocation. As a suffix on a variable name it indicates a string variable.
%	A suffix on a variable name indicating an integer variable.
&	Precedes hexadecimal constants e.g. &EF
'	A character which causes newlines in PRINT or INPUT.
( )	Objects in parentheses have highest priority.
=	"Becomes" for LET statement and FOR, "result is" for FN, relation of equal to on integers, reals and strings.
-	Unary negation and binary subtraction on integers and reals.
*	Binary multiplication on integers and reals; statement indicating operating system command.
:	Multi statement line statement delimiter.
;	Suppresses forthcoming action in PRINT or INPUT.
+	Unary plus and binary addition on integers and reals; concatenation between strings.
,	Delimiter in lists.
.	Decimal point in real constants; abbreviation symbol on keyword entry; introduce label in assembler.
<	Relation of less than on integers, reals and strings.
>	Relation of greater than on integers, reals and strings.
/	Binary division on integers and reals.

<= Relation of less than or equal on integers, reals and strings.

>= Relation of greater than or equal on integers, reals and strings.

<> Relation of not equal on integers, reals and strings.

[ ] Delimiters for assembler statements. Statements between these delimiters will need to be assembled twice in order to resolve any forward references. The pseudo operation OPT (initially 3) controls errors and listing.

Example:

```
10 oswrch=&FFF4
20 FORZ=1TO3STEP2:P%=TOP+1000
30 [ opt Z : .start lda # ASC"!"
40 ldx # 40
50 .loop jsr oswrch
60 dex:bne loop
70 rts:] NEXT
80 CALL start
90 END
```

^ Binary operation of exponentiation between integers and reals.

~ A character in the start of a print field indicating that the item is to be printed in hexadecimal.

Expression Priority:

```
(1) variables, functions () ! ? &, unary + - NOT
(2) ^
(3) * / MOD DIV
(4) + -
(5) = <> <= >= > <
(6) AND
(7) EOR OR
```

Examples of use of the priority

```
IF A=2 AND B=3 THEN          IF ((A=2)AND(B=3))THEN
A=?C AND &FFF                A=((?C)AND(&FFF))
IF A=1 OR C=2 AND B=3 THEN   IF((A=1)OR((C=2)AND(B=3)))THEN
IF NOT(A=1 AND B=2) THEN     IF(NOT((A=1)AND(B=2)))THEN
```

Characters permitted in variable names

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9 _
```

ERROR MESSAGES AND ERROR CODES

Direct Mode Only (error code 0):

Silly!	LINE space
RENUMBER space	

Disastrous and untrappable:

Bad Program	No room

Trappable:

1 Out of range	2 Byte
3 Index	4 Mistake
5 Missing ,	6 Type mismatch
7 No FN	8 $ range
9 Missing "	10 Bad DIM
11 DIM space	12 Not LOCAL
13 No PROC	14 Array
15 Subscript	16 Syntax error
17 Escape	18 Division by zero
19 String too long	20 Too big
21 -ve root	22 Log range
23 Accuracy lost	24 Exp range
25 Bad MODE	26 No such variable
27 Missing)	28 Bad hex
29 No such FN/PROC	30 Bad call
31 Arguments	32 No FOR
33 Can't match FOR	34 FOR variable
35 Too many FORs	36 No TO
37 Too many GOSUBs	38 No GOSUB
39 ON syntax	40 ON range
41 No such line	42 Out of DATA
43 No REPEAT	44 Too many REPEATs

The following words cannot be used in upper case as the start of a variable name (anything may be used in lower case):

ABS	ACS	ADVAL	AND	ASC
ASN	ATN	AUTO	CALL	CHAIN
CHR$	COLOUR	COS	DATA	DEG
DEF	DELETE	DIM	DIV	DRAW
ELSE	ENVELOPE	EOR	ERROR	EVAL
EXP	FN	FOR	GCOL	GET
GET$	GOTO	GOSUB	IF	INKEY
INKEY$	INPUT	INSTR(	INT	LEFT$(
LEN	LET	LINE	LIST	LN
LOAD	LOCAL	LOG	MID$(	MOD
MOVE	NEXT	NOT	OFF	ON
OPENIN	OPENOUT	OR	PLOT	POINT(
PRINT	PROC	RAD	READ	REM
RENUMBER	REPEAT	RESTORE	RIGHT$(	SAVE
SGN	SIN	SOUND	SPC	SQR
STEP	STR$	STRING$(	TAB(	TAN
THEN	TO	TRACE	UNTIL	USR
VAL	VDU	WIDTH		

The BBC Microcomputer System

PART II - HARDWARE SPECIFICATION

Section 1 - The Microcomputer Unit

This document specifies the hardware characteristics of the Microcomputer Unit of the BBC Microcomputer System, as agreed between the British Broadcasting Corporation and Acorn Computers Limited. It forms part of the overall specification, firmware and hardware, of the BBC Microcomputer System.

Written by: R.T.Russell, BBC Engineering Designs Department.

THE BBC MICROCOMPUTER SYSTEM

HARDWARE SPECIFICATION

Section 1 - The Microcomputer Unit

GENERAL

The machine will be designed to the highest engineering standards and all component parts will be used within the manufacturers' recommended limits. Particular care will be taken with printed-circuit layout and supply decoupling. Unused gate inputs will not be left open-circuit.

All items in this specification will apply to the Model A BBC Microcomputer unless specifically indicated otherwise. With the sole exception of the Econet interface, integrated circuit sockets will be fitted in all unequipped positions.

CASE

Dimensions: 400mm wide by 400 mm deep by 60mm high approx.
Material: Injection moulded thermoplastic.
External appearance: Colour scheme and markings to be agreed by BBC.

The case will be designed to enclose the printed circuit board(s) and mains power supply and to provide a rigid support for the keyboard. It must be sufficiently strong to withstand rough handling in transit, particularly with regard to the mounting of heavy items such as the mains transformer.

The case will be of two-part construction, the upper shell supporting the keyboard PCB and the lower shell the main PCB and power supply. Electrical connection between the two parts will be by flexible ribbon cable of sufficient length to allow easy access for servicing or modification. Precautions will be taken to prevent malfunction resulting from electrical coupling between ribbon cable conductors.

All printed circuit boards will be copper-on-fibreglass and will have screen-printed component legends. The main logic board will be through-hole-plated and have a solder resist.

Apertures will be provided for expansion connectors plus sufficient ventilation slots to allow for continuous use at a local ambient temperature of 35C without subjecting the internal components or case material to a temperature in excess of their rated maximum (with all expansion options fitted).

The two parts of the case will be secured together by four or more metal screws (not self-tapping). The case will be provided with non-slip non-scratch feet commensurate with its use in a domestic environment (on polished furniture etc.).

ELECTRICAL SAFETY

The unit will meet the appropriate BEAB, BSI and European standards and will be so marked. It will be constructed in accordance with BSI Class 1 requirements, i.e. all exposed metalwork is to be connected to earth via the earth lead of the 3-core mains cable.

Due regard must be given to the fact that the owner will be encouraged to open the case in order to fit I.C.'s and connectors; it is therefore essential that all points which could be at mains potential when power is applied be inaccessible to the "standard finger".

MECHANICAL SAFETY

There must be no sharp edges or corners which could cause damage to person or clothing when the covers are in place. It would be preferable for no sharp edges to be exposed even when the case is opened.

KEYBOARD

The keyboard will be in standard QWERTY format using a pitch between keys and between rows of 0.75", with a conventional row-stagger of 0.375" and 0.1875". There will be four rows of alphanumeric keys plus a space-bar and an additional top row of ten software-definable keys, 73 keys in all. The two SHIFT keys and the RETURN key will be at least 1.5 times the normal key size and the space bar will be at least 5.5" long with an appropriate mechanism to allow it to be pressed with equal ease anywhere along its length. The keyboard will be fixed at a convenient operating angle.

There will be four cursor-control keys at the right-hand end of the main keyboard; LEFT, UP, RIGHT and DOWN. It must be possible to generate all ASCII codes (0/0 to 7/15) by using the SHIFT and CTRL keys in conjunction with the other keys. The space bar and zero key must not be affected by depressing the SHIFT key.

The keys will have positive action with a total travel of approximately 0.2", the keypress being detected at approximately 50% total travel. The action must be acceptable to a professional typist. The keyboard will have two-key rollover on all keys except SHIFT, CTRL, SHIFT LOCK and CAPS LOCK. It must be possible under software control to determine whether a key is held depressed or has been released.

A RESET (or BREAK) key will not be provided on the main keyboard but there will be a press-button RESET switch accessible at the rear of the machine and mounted so as to minimise the possibility of being accidentally depressed.

The legend on the keys will be achieved by two-shot moulding; both pound sign and number symbol (hash) must be included (the detailed keyboard layout showing legends and codes produced is

attached). The keyboard should, as far as possible, be resistant to liquid spills and cigarette ash. The RETURN key will be a different colour from the rest.

The keyboard will be software scanned but this will be made transparent to the user, and processing overhead minimised, by interrupting the processor and scanning the keyboard only when a key is pressed.

POWER SUPPLY

The power supply will accept a mains input of between 220 and 260 volts rms at a frequency of 47-63 Hz. The total consumption will not exceed 50VA. The output will be +5v at sufficient current to power the machine with all on-board expansions, continuously rated, with preferably some excess capacity. The power supply will withstand an overload indefinitely and will protect itself from damage through overheating, even at 260v mains. The power supply will NOT be a switched-mode type.

There will be sufficient filtering to reduce electromagnetic interference conducted into the mains wiring to acceptable proportions (the exact maximum level to be agreed). There will be no exposed mains terminals within the case; if an auxiliary mains outlet is provided this must be of high quality and shuttered. A three core flying lead will be provided, terminated and anchored to the case during manufacture, having standard colours and meeting all electrical regulations governing domestic equipment. This lead is to be at least three metres in length. A high quality square-pin plug, complete with a 3 amp fuse, will be supplied ready fitted.

COMPOSITE VIDEO OUTPUT

A 1v pk-pk (75 ohms) composite video (PAL coded) output will be provided on a 75 ohm BNC socket accessible at the rear of the machine. The socket will not be fitted in the Model A machine but all circuitry associated with the video output will be included as standard. The TV standard will be 625-lines, 50-field interlaced, PAL, with a field-sync pulse consisting of a single broad pulse of 128us duration. The PAL subcarrier need not be line-locked but must be in the range 4.43361875 MHz +/- 100Hz over the temperature range 5-35C (ambient). The signal must be capable of being displayed in colour on a typical PAL (baseband input) monitor and preferably of being recorded and replayed in colour on an appropriate video cassette recorder (VHS, Beta, VR2000 and U-Matic formats).

UHF OUTPUT

The UHF output will be from an Astec modulator fixed tuned to approximately channel 36, for feeding to a domestic TV set. This will be negative modulated by a video signal as defined in the previous section, and must be capable of being displayed in colour on a typical domestic receiver. A flying lead will be provided, terminated by a standard Belling Lee plug, at least 2m in length. This lead will either be terminated inside the machine or will be supplied loose with a Phono plug for connection to the modulator through a suitable aperture in the rear panel of the machine.

RGB OUTPUTS

Four outputs at TTL levels will be provided, being red, green & blue video signals and a composite (mixed) sync pulse signal. These will be suitable for driving a high input impedance RGB monitor. The TV standard will be as previously defined. The connector for these outputs need not be fitted in the Model A machine.

AUDIO CASSETTE INTERFACE

A cassette modem will be incorporated to allow storage of programs and data on a standard audio cassette recorder, mono or stereo. The format will be asynchronous serial data with one start bit (space), eight data bits (LSB first) and a minimum of one stop bit (mark) per byte at one of two alternative speeds, 300 baud and 1200 baud. In the low-speed mode a MARK bit (logic 1) will be encoded as 8 cycles of 2400Hz tone and a SPACE (logic 0) as 4 cycles of 1200Hz tone. In high-speed mode a MARK will be 2 cycles of 2400Hz and a SPACE will be 1 cycle of 1200Hz tone. It must be possible to switch between low-speed (CUTS) mode and high-speed mode under software control, without internal modification.

The demodulator will be insensitive to input level variations of up to +6dB or -12dB and must recover the UAR/T clock from the tape in order to track short and long-term speed variations. It must cater for an instantaneous speed error of at least 10% WITHOUT relying on the inherent insensitivity to speed of asynchronous data, i.e. bit-centre sampling must be maintained. The demodulator must be insensitive to phase of the played-back signal.

Input and output levels will be standard DIN. Appropriate low-pass filtering will be incorporated in the output to avoid subjecting the cassette recorder to high-frequency components which could cause overloading or other forms of distortion. The input circuitry will incorporate band-pass filtering to reduce the sensitivity of the demodulator to high and low frequency noise, hum etc.

The cassette connector will be a seven-pin DIN socket wired so that if a stereo recorder is used both channels are recorded but

replay is from the left channel only. A DIN to DIN connecting lead will be supplied.

A relay will be included in the basic machine which can be used to switch the cassette recorder tape-transport motor on and off. This relay will be adequately rated taking into account that the load may be highly inductive. The relay will be driven by the operating system so as to allow, in particular, the easy storage and recall of cassette data files. Connections to the relay contacts will be brought to pins 6 & 7 of the DIN connector.

RS423 SERIAL INTERFACE

When the cassette interface is not in use, the serial port will be available to provide a bi-directional RS423 (+/-5v) interface for driving a serial printer etc. Provision will be made on the main PCB to include the necessary interface i.c's but these need not be fitted in the Model A machine. Baud rate will be selectable under software control to any value from the group 75, 150, 300, 1200, 2400, 4800, 9600 or 19200 bauds; the speeds will be accurate to within 0.2%. A simple handshaking input will be provided which will inhibit the serial output when negative and will enable it when positive (RS423 levels). It must be possible for the user to provide his own printer driver software to implement, for example, form feed or handshake using the reverse serial channel.

The RS423 connector will be a 6-pin DIN type and a suitable adaptor lead to a 25-way D-type socket will be required as part of the RS423 option. It may be necessary to have more than one variety of adaptor lead to suit various types of printer. In the standard version the following pins on the D-type will be wired: 2 - serial input, 3 - serial output, 7 - signal ground, 20 - handshake. In addition pin 6 will be wired to a single-bit RS423-level output, capable of being set high or low under software control, which is normally held high (+5v).

PARALLEL PRINTER INTERFACE

A parallel printer output to Centronics specifications will be provided in the Model B Microcomputer, using a 6522 input/output device plus buffers.

It must be possible for the user to intercept the normal driver software for the parallel printer so as to add special features such as software form-feed.

FLOPPY DISC INTERFACE

Provision will be made on the main PCB to fit a floppy disc controller plus data separator and buffer devices to allow interfacing to one or two mini-floppy or 8" floppy drives. Neither these devices nor the connector will be fitted in the Model A machine.

The disc connector will be a 34-way type to SA400 specification, but all control signals must be present to allow an SA800 specification interface to be implemented by means of an adaptor cable. A link on the main PCB will select the appropriate data rate, i.e. 125 kbits/sec for mini-floppy and 250 kbits/sec for 8" floppy. Both hardware and software must be capable of supporting 8" discs to the IBM 3740 specification, although this will necessitate fitting an additional ROM which will not be present in the basic machine (Model A or Model B). There is no requirement for double-data-density operation.

ANALOGUE INPUTS

Provision will be made on the main PCB to fit a four-channel twelve bit analogue to digital converter device to which external X-Y joystick controls can be connected. The ADC and connector will not be fitted in the Model A machine. The connector will be a 15-way D-type and will be common with the light-pen input (q.v.).

ECONET INTERFACE

Provision will be made on the main PCB to fit a 6854 I/O device plus buffering to implement a standard Econet interface. These devices will not be fitted in the Model A or the Model B machine.

PROCESSOR BUS INTERFACE ("TUBE")

Un-buffered address, data and control signals will be available on a 40-way connector to provide a high-speed interface to an external language processor (e.g. the Z80 CP/M option). The interface will be suitable only for this purpose and will use a short length of ribbon cable as interconnection between the two units. The connector will not be fitted in the Model A machine.

TELETEXT RECEIVER INTERFACE

A buffered processor bus interface will be provided for connection to the teletext receiver/data grabber. Interconnection between the main machine and the teletext receiver will be by means of ribbon cable having alternate ground conductors, allowing a cable length of at least 30 cm. The connector will be fitted as standard in the basic machine, but when adding the teletext receiver option it will be necessary to fit two buffer i.c.'s and a read-only-memory device containing the teletext and telesoftware firmware.

SOUND GENERATOR

A loudspeaker will be fitted in the Model A machine as standard. This will be fed from a three-voice sound-generator device capable of producing tones and music under software control.

ELAPSED TIME CLOCK

An elapsed-time clock will be included in the Model A machine, having a resolution of 10ms. It will be possible to set and read the clock time under software control.

USER INPUT/OUTPUT

A TTL-compatible 8-bit port plus 2 control bits will be provided in the Model B machine for user input/output.

START-UP OPTIONS

A set of internal links will define the input, output and storage channels, and the screen display option, to be used on machine start-up.

Default options will be as follows:

Console input - Keyboard.
Console output - VDU display in mode 7.
File input - Cassette recorder, high speed.
File output - Cassette recorder, high speed.

LIGHT-PEN INPUT

An input will be provided which allows the connection of a simple light-pen, fed to the 6845 CRT controller. This input will use the same connector as the analogue inputs. The light pen itself will not be supplied.

VDU SCREEN FORMATS

There will be 8 selectable display formats as follows (RAM requirements in bytes are shown in brackets):

0.	640*256 2-colour graphics & 80*32 text.	(20K)
1.	320*256 4-colour graphics & 40*32 text.	(20K)
2.	160*256 16-colour graphics and 20*32 text.	(20K)
3.	80*25 2-colour text	(16K)
4.	320*256 2-colour graphics & 40*32 text.	(10K)
5.	160*256 4-colour graphics & 20*32 text.	(10K)
6.	40*25 2-colour text.	(8K)
7.	40*25 teletext-compatible display.	(1K)

The alphanumeric characters displayed in modes 0 to 6 will be generated in high-resolution graphics with the "character generator" being part of the operating system ROM. These characters will be on an 8*8 matrix (with one-line descenders). In display modes 3 and 6 there will be forced gaps between character rows and therefore continuous vertical lines (on forms etc.) will not be possible. The character set is subject to agreement with the BBC but in general will be ASCII with the exception that code 6/0 will give a pound sign. For consistency between codes, incoming telesoftware will be code-converted so that 2/3 (pound) becomes 6/0, 5/15 (hash) becomes 2/3 and 6/0 (wide bar) becomes 5/15. The complementary conversion will take place in the VDU driver for display mode 7 (teletext).

In modes 0 to 6 the "colours" will be selectable from a palette of 16 effects being black, red, green, yellow, blue, magenta, cyan, white and the same eight colours automatically flashing.

Mode 7 will use a teletext character generator which will provide a full teletext/prestel format display including character rounding, double-height, 8 display colours, 8 background colours, non-contiguous graphics, held graphics, flash etc.

The circuitry to generate all these display modes will be present in the Model A machine, but it will not be possible to use display modes 0 to 3 unless the 16K RAM expansion option has been fitted.

All display modes will use nominally 40us active horizontal display period and 256 lines per field active vertical period, i.e. 77% and 89% respectively of the nominal TV display area.

Processor access to the display refresh RAM will be totally transparent and the display will therefore be completely glitch-free.

CENTRAL PROCESSOR

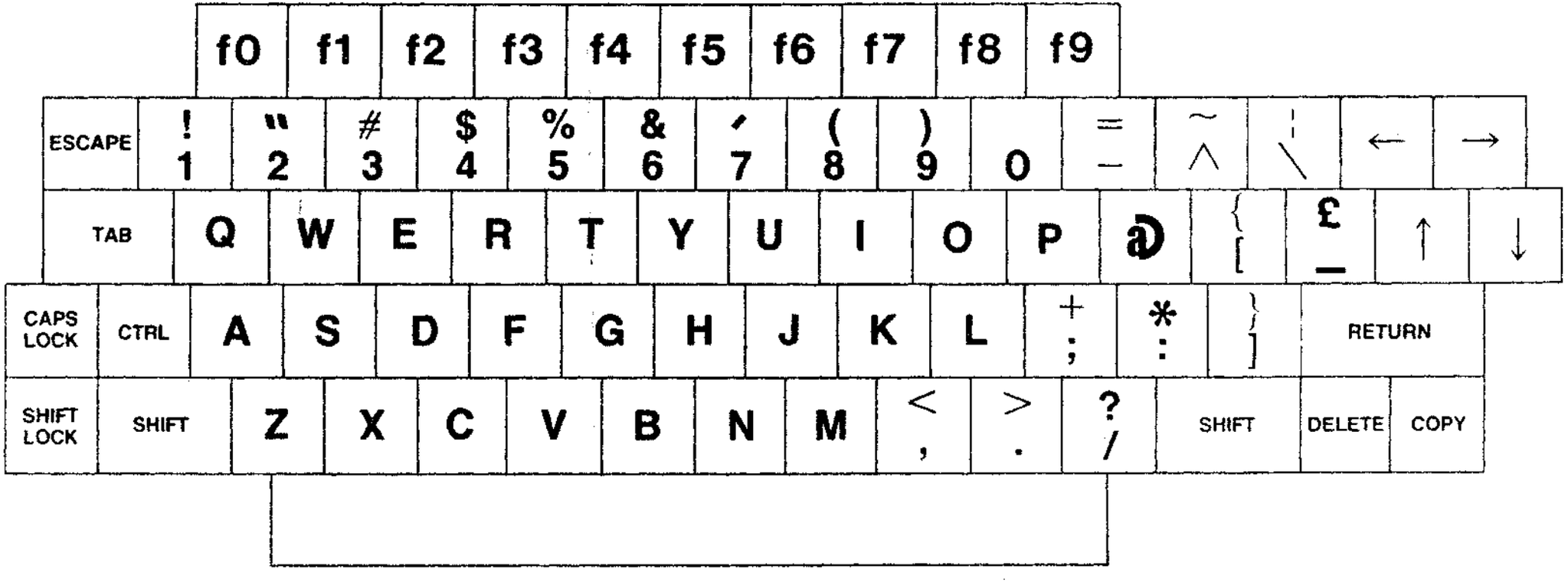
The CPU will be a 6502A running at a 2MHz clock rate except when accessing some input-output devices, when the effective clock frequency will be reduced to 1MHz.

MEMORY

The Model A machine will be equipped with 16 Kbytes dynamic random-access-memory (5v type), 15 Kbytes operating system ROM (character generator, input/output handlers, cassette operating system etc.) and 16 Kbytes language ROM (BASIC interpreter). The RAM will be capable of expansion to 32 Kbytes and the ROM to 80 Kbytes (being another 16 Kbyte language ROM - e.g. Pascal - plus up to 32 Kbytes of ROM/EPROM containing applications software etc. Only one of the language ROMs, or the ROM/EPROM, can be enabled at any given time under software control).

RADIO FREQUENCY INTERFERENCE

The level of R.F. radiation from the machine is to be minimised. The exact maximum level is to be agreed by the BBC.



BBC MICROCOMPUTER KEYBOARD LAYOUT

CBT & RR 01-04-81
24-09-81