BBC BASIC (Z80) FOR THE BRITISH BROADCASTING CORPORATION MICROCOMPUTER PLUS Z80 SECOND PROCESSOR RUNNING CP/M*

Part no 409003
Issue no 2
Date September 1984

First published 1984 Published by Acorn Computers Limited

CONTENTS

Introduction

The Z80 version of BBC BASIC has been designed to be compatible with the 6502 version supplied with the BBC Microcomputer. In general, therefore, the BBC Microcomputer user guide can be used as a reference manual for both versions. There are, however, a number of differences between the two versions that the user should be aware of. Some of these differences result from the characteristics of CP/M, some from the use of the Z80 processor and some from the dual processor architecture

1 Running BBC BASIC

BBC BASIC (Z80) is supplied on a CP/M format floppy disc, suitable only for use with the Z80 second processor. The name of the file is BBCBASIC.COM. To run BBC BASIC type

```
BBCBASIC
```

and press RETURN. The system will reply:

```
Acorn BBC BASIC (Z80) Version 2.00 (0  Copyright
R.T.Russell 1983

>
```

Alternatively, you may type BBCBASIC

<filename>

Where <filename> is the filename of a BASIC program. In this case the introductory message will not appear and the system will proceed as if a CHAIN "filename" command had been issued after initialisation. A default extension of .BBC is used if none is supplied. This can be useful in allowing BBC BASIC programs to be executed in batch mode using the CP/M SUBMIT facility. Such programs should terminate with a *CPM command to return to CP/M and allow the next program in the batch stream to execute.

2 Program flow control

BBC BASIC (Z80) uses a single control stack (the processor's hardware stack) for all looping and nesting operations, whereas the 6502 version uses separate stacks for FOR...NEXT, REPEAT...UNTIL and GOSUB...RETURN operations. The main effects of this difference are as follows.

Loop operations

There is no limit (except memory size) to the level of nesting of FOR...NEXT, REPEAT...UNTIL and GOSUB...RETURN operations. The error messages

Too many FORs Too many REPEATS Too
many GOSUBs

do not exist; instead the

No room

message (not trappable) will be issued if all stack space is used up.

In some circumstances the Z80 version is less tolerant of jumping out of loops. For example, consider the following program:

```
100 PROCtest(5)
110 END
120 DEF PROCtest(x)
130 FOR i=l TO 10
140 IF i=x THEN GOTO 160 : REM Jump out of loop
150 NEXT i
160 ENDPROC
```

If this is run on the 280 version it will result in the error message No

PROC at line 160

because ENDPROC was encountered when NEXT was expected. On the 6502 version the program will run without error, but if PROCtest is called more than ten times the

Too many FORs

error will result. This situation should not be encountered in a properly structured program, but if it is there are three main solutions:

- Eliminate the necessity of jumping out of the loop by rewriting as a REPEAT...UNTIL loop, or otherwise. This is the preferred solution:

```
120 DEF PROCtest(x)
130 i=0
140 REPEAT i=i+1
150 UNTIL i=x OR i=10
160 ENDPROC
```

- Exit the loop prematurely by setting the loop variable to a value greater than the limit value (assuming a positive step) and jumping to the NEXT statement:

```
    120 DEF PROCtest
    130 FOR i=1 TO 10
    140 IF i=x THEN i=11 : GOTO 150
    150 NEXT
    160 ENDPROC
```

Enclose the loop in a dummy outer loop and utilise the BASIC capability to 'pop' an inner FOR...NEXT loop, as follows:

```
    120 DEF PROCtest(x)
    125 FOR duimny=1 TO 1
    130 FOR i=1 TO 10
    140 IF i=x THEN GOTO 155 : REM Jump to an outer NEXT is OK
    150 NEXT i
    155 NEXT dummy
    160 ENDPROC
```

Local variables

Local variables are also stored on the processor's stack, so you cannot make an array LOCAL by using a FOR...NEXT loop. For example:

```
    200 DEF PROCtest
    210 FOR i=l TO 10
    220 LOCAL A(i)
    230 NEXT i
```

will give the message:

```
Not LOCAL at line 220
```

if run on the Z80 version. In this instance it is necessary to fabricate the loop using IF...THEN:

```
    200 DEF PROCtest
    210 i=l
    220 LOCAL A(i)
    230 i=i+l
    240 IF i<= 10 THEN 220
```

Stack pointer control

HIMEM determines the initial value of the stack pointer, which can be changed only when the stack is empty. Therefore, HIMEM cannot be changed within a procedure, function, subroutine, FOR...NEXT loop or REPEAT. . . UNTIL loop .

3 Assembly language and operating calls

Because the language processor is a Z80 rather than a 6502, the assembler and machine code calls (CALL and USR) operate somewhat differently. It is assumed in the following sections that you are reasonably familiar with the Z80 instruction set and the standard Zilog assembly language mnemonics.

The assembler

The Z80 assembler included as part of BBC BASIC (Z80) is accessed in the same way as the assembler in the 6502 version. It includes those facilities provided on Issue 2 of the 6502 version. The main differences to note are as follows:

- The standard Z80 mnemonics are accepted, ie those in the Z80 Assembly Language Programming Manual. 'ADD', 'ADC', and 'SBC' must be followed by 'A' or 'HL' (eg ADD A,C is accepted but ADD C is not), but the brackets around the port number in 'IN' and 'OUT' are optional (ie both OUT (5),A and OUT 5,A are accepted). The instruction 'IN F,(C)' which is not explicitly mentioned in the Z80 programming manual, is NOT accepted but the equivalent object code is produced from the instruction 'IN <HL),(C)'.
- There must always be at least one space between the op-code and the operand(s).
- Comments are introduced by a semi-colon (;) or by a backslash (\).
- Pseudo-op's provided are DEFB (define byte), DEFW (define word) and DEFM (define message). These behave in the same fashion as EQUB, EQUW, and EQUS on the 6502 version. Those used to other Z80 assemblers should note that DEFB and DEFW must be followed by a single value only. OPT may be used to simulate a MACRO capability. There is no equivalent to EQUD.
- Because the Z80 has no privileged zero page instructions the 'Byte' error does not exist; instructions needing an 8-bit operand simply take the least significant byte of the operand value. 'Out of range' is issued if the destination of a relative jump (JR or DJNZ) instruction is too distant, and also if an RST instruction has an illegal operand.
- A few illegal instructions (eg ADD HL,IX) are accepted without an error messsage being issued.

The USR function

This function calls a machine code subroutine whose address is its argument and returns a 32-bit integer value. The processor's A, B, C, D, E, F, H and L registers are initialised to the least significant bytes of the integer variables A%, B%, C», D%, E%, F%, H%, and L% respectively, and the returned value is the 32-bit integer composed of the Z80 registers H, L, H', L', most significant to least significant. USR behaves differently if the argument is in the range &FF00 to &FFFF (see 'Operating System access' overleaf).

The CALL statement

CALL allows a machine code subroutine to be called, with optional passing of parameters. The CALL statement sets up a parameter block which contains details of the parameters (their types and addresses). Parameters are passed by reference and can therefore be altered by the machine code routine. On entry to the subroutine the processor's registers are set up as described for USR above; also the Z80's IX register is set to the address of the parameter block and IY to the address of the subroutine (this can be useful if the machine code routine needs to find out where it is). The parameter block contains the following information:-

```
     number of parameters   - 1 byte  (IX)
     1st parameter type     - 1 byte  (IX+1)
     1st parameter address  - 2 bytes (IX+2, IX+3), LS first
     2nd parameter type     - 1 byte  )  repeated as often
     2nd parameter address  - 2 bytes )    as necessary.
```

Note that parameter type and parameter address are exchanged compared with the 6502 BASIC. Parameter types are as follows:-

```
      0 - 8-bit byte (eg ?X)
      4 - 32-bit integer variable (eg !X or X%)
      5 - 40-bit floating point number (eg V)
    128 - A 'fixed string' (eg $X)
    129 - A string variable (eg A$)
```

Except in the case of a string variable, the parameter address is the actual address at which the parameter is stored. Integer variables are stored least significant byte first and fixed strings as the characters of the string followed by a carriage return (&0D). Floating point variables are stored least significant byte first, the fifth byte being the binary exponent. If the exponent is zero, then the variable is either zero or has an integer value determined by the other four bytes. If the exponent is non-zero then the variable has a floating point value; if the exponent is 127 then 0.5<=value<l, if 128 then 1<=value<2, if 129 then 2<=value<4 etc. The most significant bit of the fourth byte is the sign bit (=1 for negative). Note that an integer value can be represented in two ways, for example the value -5 can be represented as &00FFFFFFFB or as &82A0000000; X=-5 will result in the former and X=-5.0 in the latter.

In the case of a string variable, the parameter address is the address of a 'string descriptor' which gives the current length of the string, the number of bytes allocated to the string and the address of the string (LS byte first) in that order. Note that, once again, the order of these items is not the same as in the 6502 version.

Operating system access

Addresses in the range &FF00 to &FFFF provide access to the machine operating system, as with the BBC Microcomputer on its own. In order to achieve the greatest compatibility between the 6502 and the Z80 versions of BBC BASIC, CALL and USR behave differently when addressing this area of memory. In both cases, the processor's A, H, and L registers are initialised to the least significant bytes of the integer variables A%, Y%, and X% respectively. In the case of USR, the returned 32-bit value is composed of the processor's F, H, L and A registers corresponding to the 6502's P, Y, X and A registers, most significant to least significant. Assuming that the address used is one of the legal OS entry points, the call will be communicated through the Tube to the 6502 processor where it will be executed as normal. Any result returned will be communicated through the Tube back to the Z80, the Z80' s F, H, L and A registers corresponding to the 6502's P, Y, X and A registers. Although the format of the Z80's flags register F is not identical to the 6502's status register P, the carry bit is in the same position, so nearly all OS calls made from BASIC will work correctly.

'Star' commands and the OSCLI statement

As with the 6502 version, operating system commands can be issued from BASIC either by preceding them with a star (eg *FX 4,1) **or** by using the OSCLI statement (eg OSCLI "FX 4,1") However, before being passed to the operating system the command is checked to see if it corresponds to one of the CP/M filing system commands (see overleaf). If it does, the appropriate CP/M function is carried out and control is returned to BASIC; if it does not, the command is passed to the 6502 operating system in the usual way. If you need to pass an operating system command to the 6502 which happens to have the same name as one of the CP/M commands (eg *DIR) then it should be preceded with another star; ie **DIR $ or OSCLI "*DIR $" will cause the Acorn DFS to set its default directory to "$" rather than result in a CP/M disc directory listing.

4 The CP/M filing system

BBC BASIC (Z80) runs under the CP/M operating system. Generally, input/output (eg the console) bypasses CP/M and uses the standard BBC Microcomputer operating system conventions (eg CTRL B switches on the printer) ; this is to maintain the best compatibility with the 6502 BASIC. File operations, however, use CP/M as the filing system and this gives rise to some characteristics of which you should be aware.

Filenames

Filenames accord with the usual CP/M conventions, ie they consist of an optional drive letter, a filename of up to eight letters and an optional 'extension' of up to three letters, eg D:FILENAME.EXT. The colon is part of the drive specification and the full stop is part of the extension. If the drive letter is omitted, the current drive is assumed. If the extension is omitted, .BBC is assumed. Lower case letters within filenames are converted to their upper case equivalents.

Filing system operations

The statements and functions which access the CP/M filing system are SAVE, LOAD, CHAIN, OPENOUT, OPENIN, OPENUP, EXT#, PTR#, BPUT#, BGET#, PRINT#, INPUT#. EOF# and CLOSE#. As CP/M does not provide the facility for opening a file for 'input only', OPENIN and OPENUP have identical effect. Note that CP/M is used unconditionally as the filing system;
it is not possible to change to another (eg TAPE). Access to other BBC Microcomputer filing systems may be achieved by direct calls (from BASIC or machine code) to the various entry points in page &FF (eg OSFIND, OSBGET, OSBPUT etc).

CP/M operating system commands

Those filing system operations which are not provided directly by BASIC statements and functions (eg file deletion and renaming) are made available as pseudo operating system commands. That is, they appear to the user to be normal operating system commands but are actually processed by the Z80 and are not sent to the 6502 I/O processor. These commands are similar to the facilities provided by CP/M's console command processor and use the same syntax:

*ERA filename	Erase (delete) the file 'filename'. If the filename contains a wildcard (? or *) then all files matching the name are deleted. The drive defaults to the current drive and the extension to .BBC unless otherwise specified.
*REN file2=filel	Rename filel to be called file2. The filenames must be unambiguous (wildcards are not permitted). The extensions default to .BBC.
*TYPE filename	Type the contents of the specified text file to the screen.
*DIR	List the disc directory (catalogue). The default drive is the currently logged drive and the default extension is .BBC (ie all .BBC files are listed). The command *DIR *.* will list all files on the disc.

Six more commands, which are not standard CCP commands, are provided:

*RESET	Resets the CP/M disc system. This command must be issued after changing a disc.
*CPM	Return to CP/M (warm boot).
*BYE	Has the same effect as *CPM.
*DRIVE d	Selects drive d as the current drive for subsequent file operations. The colon is mandatory.
*LOAD file aaaa	Loads the named file to memory address aaaa (hexadecimal). Note that this command differs from the normal BBC Microcomputer command in that the load address must always be explicitly given.
*SAVE file aaaa bbbb	Saves an area of memory to a disc file, where aaaa is the hexadecimal start address and bbbb-1 is the end address. Instead of specifying the end address, the length of the data block to be saved may be given, preceded by a plus sign (eg *SAVE file aaaa +1111). Note that this command differs from the normal BBC Microcomputer command in that the saved file is always a multiple of 128 bytes long, and that there is no execute address or reload address associated with the file.

Operating system commands may be entered in lower case, and filenames may optionally be enclosed within quotation marks. Commands may be abbreviated by using a full stop, eg *DR. is equivalent to *DRIVE. Filing system commands other than those listed above (eg *DUMP) are not intercepted by the Z80 and are passed to the current filing system selected on the 6502.

File size, EXT# and EOF#

CP/M (2.2) maintains file lengths only as a multiple of 128 bytes, and this is reflected in the value returned by EXT#. Another effect of this limitation is that the end-of file flag EOF# cannot be used to indicate the precise end of data in the file with any certainty: up to 127 bytes (nulls) may be read after the last data item before the EOF# flag is found to be TRUE. If these additional nulls cannot be tolerated, a deliberate end-of-file marker should be included in the file (eg an illegal data value, if the file's content allows this) or alternatively the file's exact length (the value of PTR# after the last data item has been written) can be written as the first record of the file:

```
 100 file=OPENOUT(filename$)
 110 IF file=Q PRINT "Directory full":END
 120 PRINT# file,0 : REM.  write dummy value to reserve space
  .... REM.  write data to file
 200 length=PTR# file
 210 PTR# file=0
 220 PRINT# file,length : REM.  write length as first record
 230 CLOSES file
```

EXT# performs a directory read operation so is quite slow. If repeated use of the file length is necessary, use EXT# once and save the result in a variable.

Random access files

CP/M supports random access files, and unlike the Acorn DFS such files need not occupy contiguous space on the disc. There is therefore no equivalent to the 'can't extend' error, and it is quite acceptable to lengthen an existing file by opening it for update and writing new data items at its end:

```
 100 file=OPENUP(filename$)
 110 IF file=0 PRINT "File not found":END
 120 INPUT# file,length : REM. length stored as first record
 130 PTRft file=length
 ... REM. write new data at end of file
 200 length=PTR# file
 210 PTR» file=0
 220 PRINT# file,length
 230 CLOSE# file
```

CP/M files may be 'sparse', ie areas of a random access file to which data has never been written need not occupy any disc space. Such unwritten areas are limited by this implementation of CP/M to blocks of 2K bytes in size; if the file pointer (PTR#) is set to a value within an unwritten block, EOF will be set to TRUE. It is possible for the 'virtual length' of a file (returned by EXT#) to be greater than the total capacity of the disc, but is limited by CP/M to 8M bytes (&800000 bytes).

5 Data file formats

Data files written by the PRINT* statement, and read by the INPUT# statement, have a different format from files used by the 6502 BASIC. String items consist of the characters of the string followed by carriage return (and therefore occupy n+1 bytes where n is the length of the string) and numeric items (whether integer or floating point) consist of five bytes of binary data. The stored items are not 'typed', therefore the 'Type mismatch' error is not produced when reading files; if no carriage return is found within 256 characters when reading a string value, a null string is returned.

Because strings are written in a straightforward fashion, it is possible to create a conventional text file (compatible with a Text Editor for example) by using the PRINT* statement. To separate lines of text with carriage-return line-feed it is simply necessary to add the line feed character using BPUT# :

```
 150 PRINT file,text$ : BPUT# file,10
```

Programs which use PTR# to position the file pointer when reading a data file with INPUT will probably need modifying if transferred from the 6502 BASIC to the Z80 BASIC.

6 Other differences between the 6502 and Z80 versions

The following differences are of a minor nature, and will not normally be of importance to the programmer. They are listed here for the sake of completeness:

END, or an untrapped error, closes all open files. STOP leaves files open to aid debugging. File control blocks are contained within the dynamic variable area so all files should be closed before using CLEAR, CHAIN etc or editing the program.

The default prompt from an INPUT statement is '? ' rather than **'?'** (ie there is a space after the query).

REPORT does not issue a new-line preceding the error string, so you can position the message anywhere on the screen.

A program line with a line number of zero cannot be entered (it is treated as an immediate command).

Integer variables do not increment or decrement in a 'circular' fashion; you get "Too big" if you try to increment above &7FFFFFFF or decrement below &80000000.

Leading spaces are automatically stripped from program lines.

The random number generator is initialised by RUN, therefore RND will return zero until RUN (or CHAIN) is first issued.

LISTO defaults to 7.

Characters with an ASCII value of less than 13 have no effect on COUNT.

MODE can be used within a procedure or function, and has no effect on HIMEM. HIMEM can be returned to its initial value with HIMEM=(!6)AND&FF00.

Trapped errors neither restore the DATA pointer nor turn off TRACE mode.

The 'Bad program' error performs a NEW operation.

The numeric range of floating-point numbers is approximately 5.9E-39 to 3.4E38.

INSTR returns the value zero if the string being searched for is a null string.

The exact result of complex calculations may differ from the 6502 version because of differing rounding techniques.

7 Transfer of programs from the BBC Microcomputer DFS

The filing systems used by the BBC Microcomputer alone and with the Z80 second processor differ. The BBC Microcomputer alone uses DFS and with the Z80 it uses CP/M. A utility program DIP has been provided to transfer programs or data between these filing systems.

As is usual for CP/M utilities, DIP can be invoked for a single operation, eg

```
A>DIP FRED=:1.$.BERT
```

Alternatively, DIP can be invoked and will then prompt repeatedly until terminated by CTRL C, eg

```
A>DIP
*FRED=:1.$.BERT
*:1.$.JIM=BILL
*CTRL C A>
```

To copy from CP/M to DFS the valid syntax for a command is:

```
<DFS filename>=<CP/M filename>[,/B][/Ixxxx]
```

and to copy from DFS to CP/M the valid syntax is:

```
<CP/M filename>=<DFS filename>[/B]
```

where [. . .] indicates that the contents of the brackets are optional (the brackets should not be typed). <DFS filename> represents a filename acceptable to the Acorn DFS. The filename must be given in full, ie :drive.dir.name. <CP/M filename> represents a filename following the usual CP/M conventions, eg FRED, B.FRED, FRED.EXT, A:FRED.EXT

The /B switch

6502 and Z80 BBC BASIC store BASIC programs in slightly different formats. The /B switch is used to convert between these formats, eg

```
>DIP DESTN.BBC=:3.B.SOURCE /B >DIP
:1.$.DEST=A:SOURCE.BBC /B
```

Note that there must be at least one space before /B. The /I

switch

Some CP/M assemblers produce object files always starting at 100 (hex). It is therefore sometimes convenient to ignore the first N characters of a CP/M file 'when transferring to DFS. This is done with the ignore characters switch (I). The switch takes an argument representing the hexadecimal number of characters to be ignored, eg

```
>DIP :1.$.THING=THING.COM /I 0F00
```

to ignore the first &F00 characters when transferring THING.COM from CP/M to DFS.

Note that there must be at least one space before /I; the leading zeros are optional.

The /I switch does not affect DFS load and execution addresses. These can be sorted out later if necessary using OSFILE 2 and 3.