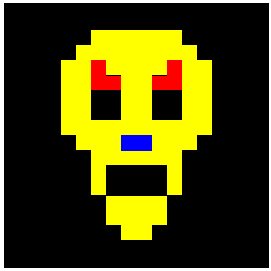

8

BASIC I, BASIC II AND ELECTRON BASIC

There are, at the time of writing, two versions of BBC BASIC available for the BBC Microcomputer, known officially as BASIC and BASIC II. In this book, however, they are referred to as BASIC I and BASIC II respectively, and the name BASIC is used as a general term to cover either. This chapter looks at the differences between the two which affect the assembler programmer and provides BASIC I versions of the new directives and keyword.

The BASIC provided on the Electron can be assumed to be BASIC II. The BASIC routines given elsewhere in this book will all work on machines containing BASIC II and can be adapted to work with BASIC I as well.

8.1 Distinguishing BASIC I from BASIC II

To find out if you have BASIC I or BASIC II press BREAK and then type REPORT. BASIC II will give the message:

```
(C)1982 Acorn
```

Whereas BASIC I will produce:

```
(C)1981 Acorn
```

8.2 The main differences between BASIC I and BASIC II

The main differences between the two BASICs which concern assembler programmers are:

OPT

In BASIC I only the lowest two bits in the OPT statement are significant; if the lowest bit is set then the machine code is listed and if the next bit is set error messages are reported. The other bits are ignored. However, in BASIC II the third bit is significant as well; it is used to produce code which will execute somewhere other than the assembled position. If the third bit is set (OPTs 4 to 7) then the code is assembled at the value of O% (the code origin), not at P%. However, all the JMP's etc. will be set up as if it is going to execute at P%, and so it is an easy matter to relocate the code. This is particularly useful if, for example, you had written a routine which had to work in ROM, or some other space which is not normally accessible. Note that if this option is used then as the code is produced both O% and P% are incremented, otherwise just P% is incremented.

EQUB, EQUW, EQUD and EQUS

Four assembler directives have been introduced in BASIC II which are not present in BASIC I. These new directives are EQUB, EQUW, EQUD and EQUS which stand for '**equ**ate **b**yte', '**equ**ate **w**ord' (2 bytes), '**equ**ate **d**ouble word' (4 bytes) and 'equate string' (0 - 255 bytes). These each take a single argument, and put its value into the assembly code at P% (also incrementing P% by the correct amount), e.g.

```
EQUB &FE                Put '&FE' at P%

EQUW oswrch             Put contents of 'oswrch'
                        at P% and P%+1

EQUD 0                  Set the next four bytes
                        to zero

EQUS "Fred"+CHR$(13)    Put 'Fred'+Carriage Return
                        in memory starting at P%
```

OSCLI

A new kevword, OSCLI, has been introduced which takes as its argument an expression, which it then passes to the operating system command line interpreter. This is not directly useful in assembler source code, but is useful when saving or loading variable amounts of data, or when sending variable FX commands. For example the following routine sets up soft key 0 to contain the string ' LIST+ERL+ [RETURN]' :

```
PROCkey (0, "LIST "+STR$ERL+"|M")
.
.
DEFPROCkey(number, A$)
OSCLI("KEY " + STR$number + " " + A$)
ENDPROC
```

Another example is to SAVE a BASIC program by typing

```
OSCLI("SAVE <filename> " STR$~PAGE + " " + STR$~TOP)
```

Note: this is exactly the same as the BASIC command SAVE <filename>. Note also the use of ' STR$~'here to convert a hexadecimal number to a string.

8.3 BASIC I versions of EQUB, EQUW, EQUD and EQUS

Macros can be set up in BASIC I to emulate these directives:

EQUB

```
DEF FNequb(byte)
?P% = byte
P% = P% + 1
= pass
```

EQUW

```
DEF FNequw(word)
?P% = word AND &FF
P%?1 = word DIV &100
P% = P% + 2
= pass
```

EQUD

```
DEF FNequd(doubleword)
!P% = doubleword
P% = P% + 4
= pass
```

EQUS

```
DEF FNequs(string$)
$P% = string$
P% = P% + LEN(string$)
= pass
```

Note that ' FNequs'will put the string into memory from ' P%'on, and will also set the byte following the string to a &D byte (RETURN character), although the next mnemonic assembled will overwrite this. In the event that this is a problem, ' FNequs'could be rewritten so that only the string is put into memory, and nothing more. This is left as an exercise for the reader.

8.4 BASIC I version of OSCLI

The following routine can be used in exactly the same way as OSCLI is used in BASIC II, e.g.

```
PROCoscli("SAVE <filename> " + STR$~PAGE + " " +
STR$~TOP
```

Note that ' cli'is an ATOM string which should be DIMensioned at the start of the program, e.g. DIM cli 64

```
DEFPROCoscli($cli)
LOCAL X%, Y%
X% = cli AND &FF
Y% = (cli AND &FFOO) DIV &100
CALL oscli                          oscli is at &FFF7
ENDPROC
```