OCP™

ADDENDA TO O.C.P. FULL SCREEN EDITOR/ASSEMBLER MANUAL

48K ONLY

FOR STANDARD 32 COLUMN & PLUS 80
80 COLUMN PROGRAMS

As from October '84, O.C.P.'s Full Screen Editor/Assembler will support 32 column printers, ZX microdrives, the RS232 port in ZX Interface 1 as well as a number of 80 column interfaces from various manufacturers. This insert is designed to supplement both the 'standard' Editor/Assembler manual as well as the +80 Editor/Assembler manual.

N.B. ADDENDUM 1 SUPERSEDES
APPENDIX 2 IN THE PLUS 80 MANUAL

ADDENDUM I — CONFIGURING +80 FULL SCREEN EDITOR/ASSEMBLER

N.B. THE FOLLOWING SUPERSEDES APPENDIX 2 IN THE OLD +80 MANUAL

When you first load +80 EDITOR/ASSEMBLER into your Spectrum, you will be asked a number of questions about the interface and the printer you are using. You will then be given the opportunity of saving your personal configuration of +80 EDITOR/ASSEMBLER to a fresh cassette tape or microdrive cartridge. Thereafter, whenever you load this copy into your machine, you will find that everything is set up for your particular requirements.

To load +80 EDITOR/ASSEMBLER, enter LOAD "", insert the supplied cassette tape into your tape recorder and press play. The program will take about 3 minutes to load and will auto-start.

You will then be presented on screen with a menu of 15 different Centronics and RS232 interfaces. Simply type in the number alongside the interface you have and press ENTER. If you are using a 32 column ZX or Alphacom printer, enter a 0 as your selected interface, and then follow the on-screen prompts to save your copy of +80 EDITOR/ASSEMBLER to tape or microdrive. The rest of this addendum is only relevant to users of 80 column printers.

If you select one of the RS232 interfaces, you will be asked some questions about the serial characteristics of your printer; specifically the baud rate, the number of data bits, the number of stop bits and whether the printer expects odd or even parity. This information should be contained in your printer's manual, but if you can't find it, don't worry. The defaults (pressing ENTER on its own) are more than likely to work.

The next question you will be asked relates to control codes sent to the printer at the START of each line. You can enter up to 8 control codes, separated by commas. Press ENTER on its own if you don't wish to use this facility.

Control codes are not printable characters as such, but affect the way characters are printed. Each make of printer has its own set of control codes and a list of these codes will generally be contained in the printer's manual. These codes can be used to instruct your printer to change the print mode (to emphasised, condensed, italics etc. depending on the facilities available). For example, to obtain condensed characters from Epson MX-80 printers, use code 15. It is quite instructive to try a few of the control codes given in your printer's manual to see what it can do when asked.

The next question relates to control codes sent to the printer at the END of each line. Again you can enter up to 8 codes, separated by commas.

Printers usually have internal DIP switches that can be set to force an automatic Line Feed (code 10) with each Carriage Return (code 13) received. If your printer is adjusted to do this, then the only control code you will need at the end of a line will be a 13 (in actual fact, pressing ENTER on its own defaults to this). If your printer is not adjusted to force automatic Line Feeds, then you will need to enter both codes 10 and 13.

Finally you will be asked some questions about the desired format of assembler printouts; (i) the number of blank lines at the top of each page, (ii) the number of actual printed lines in each page, and (iii) the number of blank lines at the bottom of each page. Note that the first value must be 1 or more, and that the sum of the three values must equal the total number of lines in a sheet of the paper you are using. Defaults are, respectively, 3, 59 and 4.

Before proceeding any further, +80 EDITOR/ASSEMBLER will list your parameters to the screen and will ask you if this information is correct. Enter 'n' and the above sequence of questions will be repeated, enter 'y' and you will be committed to your selection.

If everything is correct, all that remains to be done is to save your personalised copy of +80 EDITOR/ASSEMBLER to tape or microdrive. Simply follow the on-screen prompts to accomplish this. In the case of a microdrive save, note that the BASIC loader part of +80 EDITOR/ASSEMBLER is saved with the filename "run" to simplify the loading procedure (i.e. typing RUN and ENTER after NEW'ing the Spectrum).

IMPORTANT NOTICE

ADDENDUM II — NEW COMMANDS

The following extended commands are available to control the microdrive.

(i) Save text buffer to microdrive :-
S*<drive number><filename>
e.g. S*1testfile saves the text buffer to the cartridge in drive 1 with the filename "testfile". Note that if a file with this filename already exists on the cartridge, the user is given the option of overwriting the old file or aborting the operation.
Default drive number is drive 1 and default filename is "SourceCode".

(ii) Load text buffer from microdrive :-
L*<drive number><filename>
e.g. L*2program1 loads the text buffer with the file "program1" from the cartridge in drive 2.
Default drive number is drive 1 and default filename is "SourceCode".

(iii) Append to text buffer from microdrive :-
X*<drive number><filename>
e.g. X*scratch appends the file "scratch" to the present contents of the text buffer.
Default drive number is drive 1 and default filename is "SourceCode".

(iv) Verify text buffer or object code buffer from microdrive :-
V*<drive number><filename>
e.g. V*1prog verifies either the text buffer or the object code buffer with the file "prog" from the cartridge in drive 1. The editor/assembler is able to distinguish between the two types of files and knows what to verify with what.
If the verification fails, a message to that effect is displayed. If the verification is successful, no special message is displayed and the user is returned to edit mode.
Default drive number is drive 1 and default filename is "SourceCode".

(v) Save and verify text buffer to microdrive :-
SV*<drive number><filename>
e.g. SV*3testfile saves the text buffer to the cartridge in drive 3 with the filename "testfile" and automatically verifies it. If the verification fails, the bad file is erased and another attempt is made to save the buffer. This is repeated up to 5 times until a good copy is made.
Default drive number is drive 1 and default filename is "SourceCode".

(vi) Assemble text buffer to microdrive :-
A*<drive number><filename>/s1/s2/s3
e.g. A*1mcode/nl/ns assembles the text buffer with the 'no listing' and 'no symbol table' switches active, and saves the contents of the object code buffer to the cartridge in drive 1 with the filename "mcode".
Default drive number is drive 1 and default filename is "ObjectCode".

(vii) Assemble and verify text buffer to microdrive :-
AV*<drive number><filename>/s1/s2/s3
e.g. AV*2prog-oc/lp assembles the text buffer with a listing sent to the line printer, saves the contents of the object code buffer to the cartridge in drive 2 with the filename "prog-oc", and automatically verifies it. If the verification fails, the editor/assembler 'retries' up to 5 times.
Default drive number is drive 1 and default filename is "ObjectCode".

NEW PSEUDO-OPCODES

(i) LIST OFF and LIST ON - This pseudo-op toggles the output of hardcopy to the line printer when the /LP switch is active. A LIST OFF statement causes subsequent assembly lines to appear only on the screen. A LIST ON statement causes subsequent assembly lines to appear on the line printer (if /LP set) as well as the screen.

Most assembler programs contain a number of EQU statements to define the values of any constants used. These are usually grouped together, either at the start of a program, or at the end. A typical use for the LIST pseudo-op would be to suppress the printout of these EQU'ates when a hardcopy of an assembly is taken.

(ii) WAIT - This pseudo-op pauses an assembly, prints the message "PRESS ANY KEY TO CONTINUE THE ASSEMBLY" on the bottom line of the screen, and then waits for a key to be pressed. It takes no argument. Note that a WAIT statement takes effect on both passes of the assembler.

(iii) INCL <drive number><filename> - This powerful pseudo-op instructs +80 Editor/Assembler to begin assembling from a named microdrive file. In effect, the specified file is INCL'uded in the text buffer at the specified point. The real advantage of this is that much larger programs can now be assembled than could ever be contained at any one time in the text buffer. For example, the source code of a large program could be split up into a number of individual modules with, say, filenames "module_1", "module_2", and "module_3". An 'executive' program such as the one below could then be used to assemble the entire program in one operation with all cross-references between the modules fully resolved (unlike the /OS switch which can only be used for 'backward' references).

```
00010         ORG  8000H
00020         INCL module_1
00030         INCL 1module_2
00040         INCL 2module_3
00050         END
```

Note that each file is loaded on each pass of the assembler, and that any microdrive error will abort the assembly.

Remember that END pseudo-ops are used to mark the end of assembly source code. If the individual modules of programs contain END statements, be warned that these may lead to the premature termination of an assembly. As a general rule, therefore, END statements should be confined to the executive level.

If the modules of a program are spread across several microdrive cartridges, WAIT pseudo-ops can be used to facilitate the assembly of the whole program in one operation. For example;

```
00010        ORG  8000H
00020        WAIT ;first cartridge
00030        INCL Source1
00040        WAIT ;second cartridge
00050        INCL Source2
00060        END
```

The user must insert the correct cartridges into the microdrive in the correct sequence in response to the "PRESS ANY KEY ..." prompts. In the above example, this will involve four cartridge 'swops' in all (two on each pass).

Default drive number for INCL statements is drive 1, but note that ther is no default filename. Also note that comments should not be used on INCL lines as they are treated as part of the filename. Finally note that INCL files cannot be nested.

Users of 80 column printers and interfaces should note that the assembler automatically paginates its hardcopy output, and prints a page number at the top right hand corner of every sheet. The following two pseudo-ops are available to these users (N.B. they have no effect if a 32 column ZX printer is connected).

PAGE - This pseudo-op causes the line printer to print blank lines to the top of the next page. It takes no argument.

HEAD - This pseudo-op takes a string surrounded by single quotes (') as an argument, i.e. similar to the DEFM pseudo-op. The string operand is printed at the top of each page, and thus can be used to title a program. Note that HEAD statements can appear anywhere in the source code.

In addition, +80 Editor/Assembler patches MCTT to enable it to support an 80 column printer. Pressing the SYMBOL SHIFT and the F or G keys at any time will copy the screen to the line printer. The F key prints on the left hand side of the paper while the G key prints on the right hand side.

Note that the BREAK key (CAPS SHIFT and SPACE) can be used at any time to cancel a printout, even if a print function is invoked and no printer is connected.

Statements containing any of the above new pseudo-ops do not appear in the line printer listing but only in the screen listing.

CUSTOMISING FOR DIFFERENT INTERFACES

+80 EDITOR/ASSEMBLER has been designed to work with a wide range of Centronics and RS232 interfaces and contains all the software to drive these devices. But what if you have an interface that is not included in the screen menu - can you still use +80 EDITOR/ASSEMBLER ? The answer is yes, but a guarded yes. You will need to write some short machine code routines (see below) which can control your particular interface, and POKE them into memory - consult your interface manual for technical details.

Three machine code routines are required:-

(i) the first routine initialises the interface. If your interface needs no special initialisation, this routine can just be a RET instruction.
(ii) the second routine checks the status of the printer. It returns with the zero flag set (Z) if the printer is ready, but with the zero flag reset (NZ) if the printer is busy.
(iii) the third routine is fed with an ASCII code in the accumulator, and outputs this code to the printer.

Note that each of these routines must be terminated with a RET instruction and that any of the registers AF, BC, DE and HL can be used within them. See below for examples of the sort of routines needed.

The interface 'driver' is completed by preceding these routines with 5 'vectors'. These must contain, respectively, the start address of the driver, the entry address of the initialisation routine, the entry address of the 'busy' routine, the entry address of the 'print byte' routine and the length of the driver as a whole (length of the 3 routines plus 10 bytes for the 5 vectors). Assemble the routines with origin 40000 decimal (this will also be the value of the first vector). Note that the length of the driver must be less than 256 bytes.
This may sound complicated, but study of the example below should clarify the construction of the driver.

The procedure for entering your driver into memory is as follows :-

(a) Load the +80 EDITOR/ASSEMBLER tape as usual.
(b) Break into the program by pressing the CAPS SHIFT and the 6 keys simultaneously.
(c) POKE your driver into memory from address 57344 decimal (0E000H) onwards.
Please note that this is not the run-time location of your driver.
This can be made easier by preparing an object code tape of the driver in advance and then loading it with a LOAD ""CODE 57344 command.
(d) Re-start +80 EDITOR/ASSEMBLER with a RUN 100 command and enter 255 (user-own driver) in response to the first question.
(e) Proceed as above.

```
                ;
                ; DRIVER FOR THE KEMPSTON 'S' CENTRONICS INTERFACE.
                ;
9C40                    ORG     40000
                ;
9C40            START   EQU     $
                ;
                ; THE FIRST PART OF THE DRIVER CONSISTS OF 5 VECTORS.
                ;
9C40 409C               DEFW    START
9C42 4A9C               DEFW    INIT
9C44 569C               DEFW    BUSY
9C46 5E9C               DEFW    LPRINT
9C48 2D00               DEFW    LENGTH
                ;
                ;
9C4A            INIT    EQU     $
                ;
                ; THIS ROUTINE INITIALISES THE INTERFACE.
                ;
9C4A 01BFE3             LD      BC,0E3BFH
9C4D 3E81               LD      A,81H
9C4F ED79               OUT     (C),A
9C51 3E0F               LD      A,0FH
9C53 ED79               OUT     (C),A
9C55 C9                 RET
                ;
                ;
9C56            BUSY    EQU     $
                ;
                ; THIS ROUTINE CHECKS THE STATUS OF THE PRINTER.
                ; RETURNS WITH THE ZERO FLAG SET IF THE PRINTER IS READY.
                ;
9C56 01BFE2             LD      BC,0E2BFH
9C59 ED78               IN      A,(C)
9C5B CB47               BIT     0,A
9C5D C9                 RET
                ;
                ;
9C5E            LPRINT  EQU     $
                ;
                ; THIS ROUTINE SENDS THE CHARACTER IN THE ACCUMULATOR
                ; TO THE PRINTER.
                ;
9C5E 01BFE0             LD      BC,0E0BFH
9C61 ED79               OUT     (C),A
9C63 06E3               LD      B,0E3H
9C65 3E0E               LD      A,0EH
9C67 ED79               OUT     (C),A
9C69 3C                 INC     A
9C6A ED79               OUT     (C),A
9C6C C9                 RET
                ;
                ;
9C6D            END     EQU     $
                ;
002D            LENGTH  EQU     END-START       ; LENGTH OF DRIVER
                ;
```