Monitor
Assembler

# ASTRUM+

instruction manual

# ASTRUM+ Editor, Assembler and Monitor.

Most of the assemblers for the Sinclair Spectrum were written a few years ago and are now looking quite dated. They have, without exception, either clumsy or totally inadequate editors: They are based around storage of source and object code on tape cassettes, with microdrive extensions patched in as an accessory and, until now no facility for using the Opus disc drive unit: They have very unhelpful error reporting and many are tortuous in use.

Astrum+ aims to rectify most of these short-comings. After using many of the other assemblers on the market, all the features that were most liked have been brought together, the irritations of the old programs discarded and many enhancements added. The outcome is a program which has a superb full screen editor which supports lines of up to 256 columns a width (perfect far those of you who lake to comment their source code so that it can be understood next week), all the usual search and replace functions, and excellent block handling facilities. All of these are front-ended by a system of pull down menus.

The editor creates source files of unlimited length, handled intelligently in pages which provide the input to a powerful assembler which features full textual error reporting and a wide range of built in macro commands to aid your programming and make the tent file more readable. The editor and assembler are not co-resident in memory, but chain each other from drive so that they form an integrated machine code development environment.

Astrum+ also provides you with a monitor program, a powerful static and dynamic de-bugging tool, with witch you can disassemble and examine machine code, run and single step through routines whilst inspecting the contents of the Z80 registers and patch, modify and alter object code.

The tape cassette system is used only as a security back-up facility for the files and Astrum+ will only run on microdrives or the Opus Discovery disc drive. This frees you from the tedium of tapes and enables the program to make full use of the file handling potential of the more advanced storage devices.

If you have purchased your Astrum+ on cassette, the first program which loads will automatically transfer all the files necessary onto a formatted cartridge or disc in drive 1, leaving the cassette a security back-up; just type LOAD "" and follow the instructions as they appear on screen.

Your copy of Astrum+ consists of the following files:
Astrum+ editor (ED).
Astrum+ assembler (ass, CODE).
File copier (copy, COPY).
Text converter (exchange, EX).
Microdrive/Disc header reader (header, CATCODE).
Definer (define).
Monitor (M36, M48, M56).
Creator (C36, C48, C56)
Text file (LIBRARY).

All the files will be discussed in this manual. First of all, however, the obligatory copyright message.

**The ASTRUM+ Editor**

This is the default entry point when you type RUN after re-setting the computer. On the top line is the menu bar. The cursor appears in inverse, so is initially a black space. The bottom two lines of the display are the status lines, black on cyan.

The Astrum+ editor is a full word processor style screen editor; anything that is displayed anywhere on the screen can be altered by moving the cursor and using the editor function keys; unlink the Spectrum BASIC system it is not necessary to call lines into a separate editing area.

The screen is divided up into three logical portions. The top line is the menu bar which displays the titles of the 7 pull down menus which control the selection of the advanced editing functions, file handling and chaining of the assembler.

The bottom two lines are the status and prompt lines, The status line is used to display certain parameters regarding the state of the editor; the current state of the caps lock (initially ON), the cursor mode, (initially OVER), the column position of the cursor and the name of the current text file. The prompt line is used by the program to report errors and to request input or actions by the user.

The third, central 21 lines are devoted to the source text. Movement of the cursor is via the arrow keys on the Spectrum+ or the C/S 5,6,7 & 8 keys on the ordinary Spectrum. The delete key works in the usual way, deleting the character under the cursor and moving the remainder of the line after the cursor to the left. The source text is divided into three fields or columns. First the label field, starting in column 0, then the instruction field, the actual Z80 instruction which starts in column 8 and finally the operand field, the data, register or address to be used by the instruction. This formatting is accomplished automatically by Astrum+. Try typing in a line of assembler;

```
_LD_A,0  (_=space).
```

Press ENTER and you will see that the cursor moves to the next line down and the text has been formatted so that the label field is blank, the instruction field contains LD and the operand field contains A,0. All the text typed in is immediately formatted unless preceded by a semi-colon to denote a comment line.

If the first character of a line is not a space, then the editor treats the characters up to the first space as a label and puts the word into the label field e.g.

```
LOOP RST #10
```

When the screen is full i.e. the source teat is more than 21 lines in length, then the text will be scrolled off the top as you type in more. Movement of the cursor past the top or bottom of the screen will cause the display to scroll by one line, but more rapid movement through the file may be accomplished by using the C/S 3 and C/S 4 keys which scroll a screen at a time in an upwards or downwards direction respectively.

Entire lines may be deleted using S/S I, but if any line that you have been editing is not to your liking, then the EDIT key, C/S I, offers an undo facility a it will restore the line to the state it was in before you started to edit it.

Astrum+ positively encourages you to annotate you source code by providing long lines. The screen acts as a window to a text file which may be up to 256 characters wide; continue typing over the right margin of the screen and the window will scroll right to accommodate your line. All the functions; search and replace, delete, insert etc. work on the full 256 columns, not just the area on screen.

In addition to the editing keys outlined above, the following functions are also available from within the editor program. A full list for easy reference is included in Appendix 1 at the end of the manual.

C/S ENTER

Has the same effect as enter but the blank line will be inserted above the current line, not after it.

S/S ENTER

Moves the cursor to the last character of the current line.

S/S SPACE

Moves the cursor to the next tab position. Tab stops are set every 8 characters. The width of the label, mnemonic and operand fields are all 8 characters.

C/S SPACE

Moves the cursor to the next tab position, but also deletes all characters from the current cursor position up to the new position.

| | |
|---|---|
| S/S Y | Produces the character [ |
| S/S U | Produces the character ] |
| S/S A | Produces the character ^ |
| S/S S | Produces the character \| |
| S/S F | Produces the character { |
| S/S G | Produces the character } |

**Extended Mode Editor Commands.**

If you press SYMBOL SHIFT and CAPS SHIFT together, or the extended mode key on the Spectrum+, the border will turn red. This signifies that Astrum+, is waiting for another keypress, the effects of which are listed below.

| | |
|---|---|
| P | Produces the character (c). |
| E | Moves the cursor to the last column on the line i.e. 255. |
| S | Moves the cursor to the start of the line i.e. column 0. |
| DELETE | This deletes forward; the character under the cursor is removed and the text to the right is moved one position left. The program remains in extended mode, so the key may be repeated. |
| S/S Q | Deletes the characters from the start of the line up to, but not including, the cursor position. |
| S/S E | Deletes from, and including the character at the cursor position, to the end of the line. |
| S/S R | Use with care! This key will reset Astrum+; It erases all text files from memory and initialises all options. |
| L | Lists 21 lines, starting at the current line. |
| T | Toggles auto tab mode. Lines of source code are usually formatted on-screen when you press the ENTER key. In auto tab mode the space key inserts a tab to the next field, so the text is formatted as you type it in. It is purely a matter of preference which mode you use. Note that a semi-colon in column 0 denotes a comment line and the tab function is turned off for the entire line. |
| F | Toggles the auto-formatter on and off. The auto-formatter determines what Astrum+ does to every line after you have typed it in. Normally it splits (or formats) each line into label/opcode/operand fields, but by turning this off you can use the Astrum+ editor as a simple word processor. When the formatter is turned off, the bottom two lines will become reversed (i.e. Black paper and cyan ink). |

NOTE: Files produced unformatted cannot be assembled. You are not given access to the COMPILE menu while auto-format is turned off.

B

Returns you to BASIC. If you enter RUN then Astrum+ is restarted with all text files intact; nothing is lost.

**ASTRUM+ Menus.**

Astrum+ has a series of pull down menus, 7 in all, by which you can select some of the more complex functions and utilities. The selection as shown by the bar at the top of the screen; the current menu is denoted by a bright background. S/S W pulls down the current menu, displaying the options available with the menu cursor shown in inverse attributes. To select one of the options, press any key (except ENTER) to move the menu cursor over the option of you choice and then press ENTER. This selects the option and Astrum+ proceeds as directed.

The currant menu is changed using the S/E E key; you will see the highlighted position move one menu to the right. Further presses of S/S E circulate around the 3 menus on the bar. Access to the second bank of three menus via the S/S Q key which toggles between the two sets of menus. The following section describes the 7 menus and the options they contain.

**MODE Menu.**

Over:

Sets the cursor mode to over; any character typed over-writes the character at the cursor position. To insert single characters, use graphics (C/S 9) to insert a single space, then the key for the character to be inserted.

Insert:

Sets the cursor mode to insert; characters typed are inserted at the cursor position.

Extended:

Deals with multi-page files, details of which will be covered in a later section.

**FILE Menu.**

Create New File:

In order to correctly handle large files, Astrum+ requires that a file is created before any text is written to it. This is done via the create new file option. You will be prompted for a file name, which may be up to 10 characters in length, but it is preferable to restrict your Astrum+ source files to 9 characters, for reasons that will become apparent.

If a file of the same name exists, you will be asked whether it is to be erased; if the answer if no, then the create option is aborted, else it is erased and a new file is created. The create file command does not save the file; this must be done explicitly.

Save File:

Will save the current file if it has been created as a new file or loaded from an existing file. Any existing copies a automatically erased, the text is saved, verified and any errors reported; this system ensures maximum security of the source text.

Load File:

Loads a named file from drive and makes the file name the current file.

Catalogue

Catalogues a microdrive cartridge or disc; Astrum+ waits for a keypress before returning control to the editor.

Print Out Text:

Produces a hard copy of the source text, or portion of it. Printing starts at the current cursor line and prompts for the end; three responses to this prompt are valid:

| | |
|---|---|
| n | A number; that number of lines is printed. |
| * | A '*' character; printing continues to the end of the text file. |
| - | A ‘-‘ character; text is output until the next block marker is found. |

A block marker is a ‘-‘ character in column 0 of a line. Thus, to print out a section of the text, insert a block marker at the end line of the block of text you wish to print, move the cursor to the first line of the required block, select the print out option and specify ‘-‘ for the end of the text.

All output to the printer is via stream 3 so you must first interface to initialise it; include the BASIC commands to load and run the interface routine in line 5 of the program "run" in your Astrum+ suite. If this software is less than 256 bytes in length, then locate it in the printer buffer area at 23296. If you do not have such an item of software, a suitable driver can be supplied by Bradway Software.

Mistake:

Allows you to exit this menu without carrying out any of the options.

Whilst executing any of the above options it is possible that a microdrive or disc problem may drop you back into BASIC with an error report. If this occurs do not worry, just type RUN and Astrum+ will restart with all files intact. If the error occurred whilst trying to load a file, then the portion of text loaded prior to the fault is available in the editor. Contrast this to the usual situation where an incomplete file is most unhelpfully erased.

FIND Menu.

Find String:

Astrum+ does not need and does not have line numbers, so you will use the find facility very often to jump to places in the text file. The search string is requested and search made from the current cursor position to the end of the text. If the string is found, the cursor is positioned at the start of the string and the relevant area of the text displayed on screen. Trailing spaces are ignored.

Find Again:

Searches for the next occurrence of the string last defined in the find string option. This provides a method of moving down through the text to examine multiple entries of the same string.

Replace String:

Replaces the first string you type with the second string. All occurrences of the first string from the current cursor position to the end of the text are replaced with the second string.

Top of Text:

Moves the current line to the first line of the text file.

Bottom Of Text:

Moves the cursor to the bottom line of the text file.

**COMPILE Menu.**

Why is it called compile and not assemble? You try to write a capital 'S' four pixels high! The assembler and editor occupy the same work space and so only one can exist in memory at any one time. This is, however, not a problem as the Astrum+ control program loads in the editor or assembler at the appropriate time. The options available from the compile menu will be described later in the section dealing with the Astrum+ assembler.

**INFO Menu.**

A rag-bag of options that would not fit anywhere else!

Memory Left:

The amount of memory available for a text file resident in memory is 10K. This option displays the number of bytes used and the number remaining. If you attempt to exceed the 10K limit your file may become corrupted; Astrum+ gives you advance warning of this by turning the border colour yellow.

Numbers:

A useful utility which allows you to convert a number between the four bases; binary, decimal, octal and hexadecimal. All you have to do is to type in the number and its value will be computed and displayed in the four bases. The number should be in the format;

|  |  |
|---|---|
| nn | for decimal |
| #nn | for hexadecimal |
| _nn | for octal |
| @nn | for binary. |

The same conventions are used when the values are displayed on screen, and throughout the Astrum+ system.

Change Drive:

If you have more than one drive it is often useful to be able to load and save on different drives. The drive can be changed by just typing in the number and pressing return. Numbers from 1 to 8 only are acceptable in the microdrive version and 1 or 2 in the Opus version.

Tape Backup:

This option is to maintain backups of your source files for reasons of security and cheapness. The following options are available:

|  |  |
|---|---|
| 1 | Save the current memory to tape |
| 2 | Load a file from tape and appends it to any text already in memory |
| 3 | Return to the editor. |

If you press BREAK whilst loading or saving you will be returned to BASIC; Astrum may be re-entered by typing RUN. Any text file which was loading when you broke into the routine will be available in memory, but will probably terminate unsatisfactorily.

If you have purchased your copy of Astrum+ on cassette the last file on side 2 is a library of useful system addresses and numbers for inclusion into your programs. This should be loaded in using the tape backup facility and then saved to microdrive or disc drive as required.

**BLOCK Menu.**

The last choice from the first menu bar deals with the manipulation of blocks of program. Block markers are the '-' character in column 0 of a line. The block marker should occupy a line of its own; any other text on the block marker line will be excluded from the block operation selected. The first options require only s single black marker as the current line is taken as the beginning of the block.

Delete:

Lines can be deleted singly by pressing S/S I. This is not convenient for more than just a few lines. There are 3 choices from the block delete option, all of which take the current cursor position as the first line:

| | |
|---|---|
| nn | A number. Deletes nn lines from the cursor line; nn may be prefixed by any of the base operators @, _ or # as mentioned previously. |
| * | A '*' character. Deletes all the text from the current line to the end of the file. |
| - | A '-' character. Deletes up to and including the next block marker in the text. |

Write:

Prompts for a filename and writes out a block of text delineated by a pair of block markers to the file. If the file already exists, you are asked to confirm whether it is to be deleted. If the block is being written from a named file, then it is saved with a comment detailing the source file name. In the Opus version the block of text is deleted from the source file as it is written out.

The next 3 options require that a block be identified by two block markers, one at the beginning and one at the end of the area of text to be manipulated.

Move:

Mark the text to be moved using a pair of block markers, move the cursor to the destination and select this option. The block will be moved from its initial position and will be inserted immediately before the destination line. The block markers are left in their source position for reference.

Copy:

The same as block move, but the text is not deleted from the source position. This enables program chunks to be replicated without retyping, although you may have to alter the names of any local labels used.

Join:

Prompts for the name of a block of text previously written out to drive and joins it onto the end of the current text file. It may easily be moved from there to the current cursor position using the block move option. The source text may have been written using the above block write option or may be a text file in its own right.

Mistake:

Abandons the block menu without any action.

EXTENDED Menu.

In order that Astrum+ can manipulate source files of greater than 10k in length, the assembler uses a series of files of the same name each with a suffix, which is always the tenth character of the filename, starting at 1 and proceeding through the entire ASCII character set. These are referred to as pages of the file, if you have already created a file of Astrum+ text you will have noticed in the status line the message "PAGE 0" next to the filename; this will always tell you which page you have in memory. Each page is a file in its own right as well as being a part of the main file. Thus, if you have 2 files on your cartridge or disc, "testfile" and "testfile 1", then testfile is page 0 and testfile is page 2 of the same source file.

It is possible to create new pages to extend a file by using the create new text file option but there is a menu devoted to the manipulation of multi-page source files. Select the MODE menu and then the Extended option; another menu will pull down in place of the MODE menu. The options it contains are as follows;

Add on Text:
: This option is specifically to create new pages in a multipage file; upon selection it tries to create a file with the same name as the current one, but with the page number incremented by 1. The option clears any existing text from memory. It is sensible to create new pages when the current one contains about 8K of text so that there is some space left for alterations, and for workspace.

Up File:
: Will attempt to load in a page 1 less than the current one. If the page is not found, then the current page number is not altered.

Down File:
: The opposite of up file, it attempts to load in the page 1 greater than the current one. Similarly if the page is not found, the number is not altered.

Goto Page:
: Typing the number of the page of the current file you require causes it to be loaded. If you type '-' then the current page is reloaded, a useful function if you have made same gross error in editing the current page and wish to restore it to the state in which it was last saved.

Mistake:
: As usual, this abandons the menu without any action.

The current page may be saved by going to the FILE menu and selecting Save File. In addition, all four options ask if you wish to save the current page before proceeding with the requested option.

## The ASTRUM, Assembler.

Assembly is achieved from the COMPILE menu which, after asking you to confirm that the assembler cartridge or disc is in drive 1, automatically chains in the two pass assembler, The first pass checks the syntax and forms the symbol table, the second pass generates the object code. Three options are available from within the compile menu:

Assemble File:

Assembles the file whose name appeared in the editor status line as the current file.

Assemble Memory:

Assembles the text file in memory.

Mistake:

Does nothing but return you to the command line.

Before assembly commences a series of options are presented: press the key corresponding to the option to cycle through the choices available.

List:

Controls the output of the assembly listing which may be;

| | |
|---|---|
| None | -No listing produced. |
| Screen | -Assembly listing to the screen. |
| Printer | -Assembly listing directed to stream 3. |

Symbol Table:

Controls the output of the assembler symbol table, the list of all the labels and their values;

| | |
|---|---|
| None | -No symbol table is displayed. |
| Screen | -The symbol table is displayed on screen. A feature of Astrum+ is that any labels which appear but are not used are displayed in inverse video. You will be surprised how these accumulate during the development of a program and they all waste space and assembly time. |
| Printer | -Symbol table listing is sent to stream 3. |

Position on Error:

The action taken by the assembler when an error is encountered;

| | |
|---|---|
| On | -Any error causes assembly to terminate, reloads the editor and text file and positions the cursor at the error line, in this way a fully interactive assembly and editing system is achieved. |
| Off | -The assembler will just list the errors which occur and continue with the assembly. |

Code To Microdrive/Disc:

Selects the destination of the object code.

| | |
|---|---|
| On | -Sends the object code to drive. Type in the name of the file when prompted; if the file already exists the previous version is erased. The code is also written to the memory as described in the next paragraph. |
| Off | -Code is put in memory; the code can be run as long as the ORG is 56000 and it is less than 2000 bytes in length. |

Numbers:

The numerical output from Astrum+ may be printed in decimal or hexadecimal.

After the desired options have been selected using keys 1 to 5, the ENTER key starts assembly. If errors are detected in the first pass then assembly will not proceed to the second pass.

Astrum+ is configured ouch that when it is resident and space has been allocated for the various text file buffers and the symbol table, then the area of memory from 56000 to about 58000 is available for the object code. If no ORG directive is included in the source, then an origin at 56000 is assumed and assembled code less than 2000 bytes in length can be run without saving it to drive and reloading it. Larger programs, or those occupying memory outside the range 56000 to 58000, must be assembled to drive, the assembler deleted from memory and the machine code reloaded and run.

After assembly has completed some information about the run is produced; this includes the length of the object code, the number of assembly errors generated and the length of the symbol table. The space allocated for the symbol table is about 10.5k; if more labels are generated than well fit into this space, then it is likely that the assembler will crash. However, a symbol table of this size is unlikely to be encountered in a program of less than 20k, or so object code size.

As with the editor, any time that the assembler has to access the drives you are prompted to confirm that the correct cartridge or disc is in the appropriate drive. This is a fail safe device designed to stop Astrum+ trying to load or save onto the wrong medium.

## Assembly of Multiple Page Files.

Selection of the Assemble File option from the COMPILE menu loads the assembler, allows you to set the assembler options, then starts at page 0 and assembles through each subsequent page until the assembler directive END is encountered in the text. A similar process is repeated for pass 2 and then the editor is reloaded. The page remaining in the editor workspace will be the final page of the text file and not necessarily the one in memory when you started assembly.

## Astrum+ Source Code

Astrum+ will assemble all the standard Zilog Z80 mnemonics, lists of which are easily available in various standard text books on the chip. Some of the other Spectrum assemblers depart from these standards; a list of the main deviations that you may encounter are listed here:

| Astrum+ | Variant | | |
|---|---|---|---|
| ADC A,r | ADC r | | |
| ADC A,n | ADC n | | |
| ADD A,r | ADD r | | |
| ADD A,n | ADD n | | |
| ADD A,(IX+0) | ADD A,(IX) | | |
| SBC A,r | SBC r | | |
| SBC A,n | SBC n | | |
| EX AF,AF' | EX AF,A'F' | EX AF,AF | EX AF,AF" |
| IN A,(n) | IN A,n | | |
| OUT (n),A | OUT n,A | | |

Astrum+ will accept mnemonics typed in either upper or lower case letters, but the former is the customary style.

Many Z8O instructions have numerical operands; Astrum+ allows you to substitute the number with an arithmetic expression which is evaluated during assembly. For example, LD A,2 could be written as LD A,1+1. Do not leave any spaces between the numbers and their operators. The following operators are recognised by the assembler; the expressions can be of any length and are evaluated strictly from left to right; there is no inbuilt operator precedence.

| | |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Modulo division; the remainder after an integer division |
| & | Logical AND |
| ? | Logical OR |
| ! | Logical XOR |
| = | Equality; returns 1 if the test is true, 0 if false |
| > | Greater than; returns 1 if the test istrue, 0 if false |
| < | Less than; returns 1 if the test is true, 0 if false |
| >= | Greater than or equal; returns 1 if the test is true, 0 if false |
| <= | Less than or equal; returns 1 if the test is true, 0 if false |

In addition to the operators, there is a range of different numerical formats which are recognised;

| | |
|---|---|
| nn | Decimal number |
| #nn | Hexadecimal number |
| @nn | Binary number |
| _nn | Octal number |
| $ | Location counter; the address of the start of the current instruction |
| “ | The ASCII value of the character in quotes |
| label | A value contained in the symbol table |

**Labels.**

Labels are identifiers in the first field of the source code and are the means by which reference to lines is made in the absence of line numbers. Astrum+ labels may be 8 characters in length, rather more generous than most other assemblers, which makes for more meaningful names and hence more easily read and maintained source code.

Using the EQU and DEFL directives, the value of labels may be defined so that they act as variables and constants within the program.

**Assembler Directives.**

The following directives are not Z80 opcodes but are instructions to the assembler, some of which modify the code produced whilst others just make the assembly process more powerful. They are typed in just as the opcodes are, with the directive in the instruction field and the argument in the operand field. The following directives are recognised by Astrum+: Firstly those which directly influence the object code;

ORG nn

ORiGin; sets the assembly address to the value nn. This enables code to be assembled to run at any location in the memory. It is advisable to put only one ORG instruction in a file, although this rule may be relaxed if you are sure of what you are doinq.

LOAD nn

When code is assembled to drive, the default load back address is to the assembler buffer area at 56000, irrespective of the ORG address The load back address can be set with the lead directive; the usual format of this command is likely to be;

```
            ORG 60000
            LOAD $
```

which will set the load address to the same as the ORG address.

DEFB n1,n3,n3...

DEFine Byte; allows you to insert specified bytes into the machine code. The syntax is very flexible as a space is treated as a zero data item e.g.

| Instruction | Values inserted |
|---|---|
| DEFB | 0 |
| DEFB 10,10 | 10 10 |
| DEFB 10, | 10 0 |

Astrum+ will also accept DB as an abbreviated form.

DEFW n1,n3,n3...

DEFine Word; similar to DEFB except that a two byte word is inserted, using the standard Intel format of low order byte first e.g

| Instruction | Values inserted |
|---|---|
| DEFW | 0 0 |
| DEFW 10 | 10 0 |
| DEFW 256, | 0 1 0 0 |
| DEFW 529, 6, | 17 2 6 0 |

Astrum+ will also allow the shortened form of DW.

DEFM "string"

DEFine Memory; Inserts ASCII values into memory. The codes for the characters between the quotes will be inserted into the machine code e.g.

| Instruction | Values inserted |
|---|---|
| DEFM "Hello” | 72 101 108 108 111 |

Astrum+ will also allow the shortened DM directive.

DEFI "string"

DEFine memory with Invert; this is almost identical to DEFM except in that the last character has bit 7 set. This is particularly useful for using the ROM routine PO_MSG at #0C0A, which prints a message to the current channel and determines the end of the message by looking at bit 7 of each character as it prints it.

| Instruction | Values inserted |
|---|---|
| DEFI "Hello” | 72 101 108 108 239 |

DEFS nn(,n)

DEFine Space; DEFS allows the reservation of a block of memory and, optionally, fills it with a specified byte. In the absence of a fill character 0's are used, e.g.

| Instruction | Values inserted |
|---|---|
| DEFS 5 | 0 0 0 0 0 |
| DEFS 5, “.” | 46 46 46 46 46 |

Astrum+ will also allow the shortened form of DS.

<LABEL> EQU nn

EQUate; a particularly useful directive which sets a label with a constant value for use within the program e.g. LINEFEED EQU 10; from this point all references to LINEFEED will insert the value 10.

Astrum+ will also allow the symbol = to represent EQU

<LAEEL> DEFL n

reDEFlne Label; allow. the re-definition of a previously declared label or symbol. Its old value is discarded and replaced with nn. Only the last assigned value for any label will appear in the symbol table at the end of assembly.

Astrum+ also allows the directive DL.

The remaining directives are Astrum+ control commands:

END

Must be present on the last page of a multi-page source file, or the last page that you wish to assemble. During assembly the source code is processed to the end of the page containing the END directive.

LIST

Controls the output of the assembly listing, enabling you to turn it on or off or direct it to different channels. The following options are available;

| | |
|---|---|
| None | -Turns off assembly listing. |
| Screen | -Starts to send the assembly listing to the screen. |
| Printer | -Sends an assembly listing to stream 3, usually to the printer unless you have specifically diverted it elsewhere for some reason. |
| #n | -Sends a listing to a designated stream, where n is a number in the range 1-16. Stream 1 is the equivalent of LIST NONE. |

TEXTP

If assembling a long file you may not always need the Insert Cartridge or Disc prompts from the program. The two instructions TEXTP ON and TEXTP OFF allow you to control the prompts, This is useful when you have more than one drive and the source file is one and the object code is being sent to another; there is no need to change cartridges during assembly in this case, so the redundant prompts may be turned off.

CODEP

Similarly controls the production of prompts for the object code cartridge or disc.

TYPE

Enables you to send a message to the screen when the assembly has reached this point in the file. This will probably be used in conjunction with the STOP directive, described next.

STOP

Stops the assembly until a key is pressed. In conjunction with the TYPE directive it allows you to keep different sections of files on different cartridges or discs, if so desired; e.g.

```
TYPE Change cartridge now
STOP
```

**Predefined Macros.**

The Inclusion of a series of predefined macro commands within the specification for Astrum+ was not to economise on the size of source code files, although it does so as each individual macro instruction substitutes for, typically, four times the length of text which it replaces. Instead it was primarily to make the source code more easily readable, understandable and hence maintainable. The microdrive version of Astrum+ has 36 inbuilt macros to cover many of the microdrive handling hook codes and many more of the commoner and more useful ROM calls, whilst the Opus disc version has all but the Interface 1 hook code macros. The following list details all the macros, shows examples of their use and lists the CPU registers which are altered by the calls.

```
CHAN n
             Inserts the code
                          LD A,n
                          CALL #1601
             Opens stream n for reading or writing.
             Exit condition: AF, BC, DE, HL altered.

VDU n1{,n2...}
             Inserts the code
                          LD A,n
                          RST #10
             Sends the characters n1, n2 etc. to the current stream.
             Exit condition: AF altered.

CLS
             Inserts the code CALL #OD6B
             Identical to the BASIC instruction CLS.
             Exit condition: All registers except IX, IY altered.

PAGE nn
             Inserts the code
                          RST #10
                          DEFW nn
             This routine is used to call the main ROM when the Interface l ROM is paged
             in. It causes the shadow ROM to page out, the main ROM routine at address nn
             to be called and then, on return, the shadow ROM to be paged back in again.
             Not a routine to be used by the novice!
             Exit condition: Shadow ROM paged in, register usage depends on the main ROM
             routine called.

STKA
             Inserts the code CALL #2D28
             The value in the A register is put onto the calculator stack.
             Exit condition: A11 registers are altered.
```

STKBC

Inserts the code CALL #2D2B
The value in the BC register pair is put on the top of the calculator stack.
Exit Conditions: All registers are altered.

PRFP

Inserts the code CALL #2DE3
The value on the top of the calculator stack is printed to the current stream.
Exit condition: All registers altered e.g.

```
         CHAN 2
         LD A,1
         STKA
         RST #28
         DEFB #26,#38
         PRFP
         RET
```

This routine prints out the natural log of 1 i.e. e.

PRBC

Inserts the code

```
             CALL #2020
             CALL #2DE3
```

Prints out the value in the BC register pair to the current stream, using the calculator.
Exit conditions: All registers altered.

FPTOA

Inserts the code CALL #2DD5
This compresses the floating point number on the calculator stack into the A register.
Exit conditions: Carry set if the number >255; Z if the number=negative; NZ if the number =positive. All registers altered.

FPTOBC

Inserts the code CALL #2DA2
Similar to FPTOA, except that the number is put into the BC register pair.
Exit conditions: Carry set if the number >65535; Z if the number negative; NZ if the number positive. All registers altered.

MULT

Inserts the code CALL #30A9
This has the effect of let HL=HL*DE, a useful and very fast routine.
Exit conditions: Registers AF, BC, HL altered.

PRINT

Inserts the code CALL #203C
Print, out a string of characters to the current stream. DE must hold the start of the characters, and BC the number of characters to be printed.
Exit conditions: DE holds the address of the byte after the message, BC=0 and A=0 e.g.

```
                CHAN 2
                LD DE,MESSAGE
                LD BC,MESSEND-MESSAGE
                PRINT
                RET
MESSAGE         DEFM "Sample messages"
MESSEND
```

POMSG

Inserts the code CALL #0C0A
Prints any message from within a table of messages. DE holds the start of the table and A the message number. The last character of each message must have bit 7 set and the table must start with a number >127.
Exit conditions: All registers are used e.g.

```
                LD AE,TABLE
                LD A,1
                POMSG
                RET
TABLE           DEFB 128
                DEFI "Message 0"
                DEFI "Message 1"
```

Prints out the string <Message 1>

INKEY

Inserts the code CALL #15E6
This routine tries to read a character from the current stream and put it in the A register.

BEEP

Inserts the code CALL #0385
Produces the wonderful, diminutive Spectrum bleep. DE contains the length of the note and HL the pitch.
Exit conditions: All registers affected, interrupts enabled.

The remaining macros all deal with interface 1 hook codes and so are not available in the Opus disc drive version. Reference is made to variables in the microdrive channel area, such as DSTR 1. For further details of these, see Andy Pennel's microdrive book. The code inserted during assembly is;

```
                RST #8
                DEFW #n
```

where n is the hook code number.

OPENM

Hook code #22
Opens a microdrive channel.
Entry conditions; D_STR1 contains the drive number, T_STR1 contains the address of the file name, N_STR1 contains the length of the file name.
Exit conditions: IX contains the start address of the channel, HL contains the stream displacement from IX.

CLOSEM

Hook code #23
Closes a microdrive channel. If the channel was open for writing, then the contents of the buffer are first sent.
Entry conditions: IX points to the beginning of the Channel.
Exit conditions: Drive off, interrupts on.

MOTOR

Hook code #21
Turns microdrive motor on or Off
Entry Conditions: If A=microdrive number (1-8), turns drive on, if A=0, turns all motors off.
Exit conditions: A=microdrive number; Interrupts off, A=0: interrupts on.

ERASEM

Hook Code #24
Erase a microdrive file.
Entry conditions: D_STR1 contains drive number, T_STR1 contains address of file name, N_STR1 contains length of file name.
Exit conditions: File erased:

RECLAM

Hook code #2C
Reclaim a microdrive channel.
Entry Conditions: IX.start of Channel.

READP

Hook code #27
Read from a print type file; this reads a specific record number.
Entry conditions: CHDRIVE contains drive number, CHNAME contains file name, CHREC contains record number.
Exit conditions: Motor on, interrupts off.

READNP

Hook code #25
Read the next record from a print file.
Entry conditions: CHDRIVE contains drive number, CHNAME contains file name.
Exit conditions: Motor on, interrupts off.

READS

Hook code #29
Reads in the next record; if the file is not a print type file, then the buffer is cleared.
Entry Conditions: CHDRIVE contains the drive number.
Exit conditions: Motor on, interrupts off, Carry reset if print file, set if a non print file.

READR

Hook code #28
Reads in a numbered record; if the file is not a print type file, then the buffer is cleared.
Entry conditions: CHDRIVE contains the drive number, CHREC contains the number of the record to be searched for.
Exit conditions: Motor on, interrupts off, Carry reset if print file, set if a non print file.

WRITES

Hook Code #26
Write buffer to next available space.
Entry conditions: IX=start of channel, CHBYTE contains record length, CHREC Contains record number, CNNAME contains filename. CHDRIVE contains file number.
Exit conditions: CHBYTE=0, CHREC=CHREC+1, motor off, interrupts on.

WRITER

Hook code #2A
Write buffer to a specified sector number. Entry conditions: IX=start of channel, CHDRIVE=drive number, CHNAME=filename, CHREC=sector to be written to.
Exit conditions: Motor off, interrupts on.

232IN

Hook code #1D
Read a byte from RS232 port.
Entry conditions: none.
Exit conditions: Byte in A, carry reset on error.

232OUT

Hook code #1E
Write a byte to the RS232 port.
Entry conditions: Byte in A.
Exit conditions: Interrupts on.

OPENR

Hook code #34
Open an RS232 channel. NOT version 1 shadow ROM!

OPENN

Hook code #2D
Open a temporary network channel.
Entry conditions: DSTR__1=destination station number, NSTAT=base station number.
Exit conditions: IX =start of channel (CURCHL).

CLOSEN

Hook code #2E
Close a network channel; if it is a write channel, send the buffer first.
Entry conditions: (CURCHL)=start of channel.

READN

Hook code #2F
Read a packet from the network. Does NOT Work on version 1 of the shadow ROM!
Entry conditions: IX contains start of the channel.

WRITEN

Hook code #30
Write a packet to the network.
Entry conditions: IX= start of the channel, A=0 for data, A=1 for EOF, NCNUMB= block number, NCIRIS= destination station number.
Exit conditions: A= destination number.

NEWVAR

Hook code #31
Create new system variables if they do not exist.

SHADOW

Hook code #32
Call a shadow ROM routine.
Entry conditions: HD_11 = the address in the shadow ROM to be called.

**ASTRUM+ Error Reporting.**

Astrum+ does not leave you floundering with terse error numbers, but has a comprehensive error reporting system. Although the errors are in plain English, here is a list of them with some extra explanation.

8 bit overflow:
An instruction requiring a single byte operand is out of range e.g. LD A,757

Bad exchange instruction:
Illegal register exchange has been attempted: e.g. EX A,B

Bad load instruction:
Incorrect syntax in an LD instruction e.g. LD A,

Bad port:
Syntax error in an IN or OUT instruction e.g. IN B,(D)

Bad opcode:
The assembler has encountered an instruction that it does not recognise.

Bad ORiGin:
Syntax error in the operand of an ORG statement e.g. ORG !

Badly formed index address:
Syntax error in an instruction using IX or If registers e.g. ADD A,(IX+!)

Bit number not valid:
Attempt to operate on a bit in excess of 7 e.g. SET 8,A

Division by Zero:
An attempt to divide or take the zero modulus of a number is not meaningful e.g. SUB 256/0

Expression does not make sense:
Usually reflects a gross syntax error e.g. BIT .1 ,A

Index displacement out of range:
The displacement using the IX or IY registers is outside the range -128 to +127 e.g. SUB (IX+130)

Invalid stream:
The Spectrum supports streams 0-16 only, but a attempt has been made to LIST to a stream outside this range.

Interrupt mode not valid:
An operand other than 0, 1 or 2 has been used in an IM instruction e.g. IM 4

Invalid ReStarT address:
Probably caused by a missing # symbol in a hexadecimal RST instruction e.g. RST 30

Invalid register:
An invalid register identifier has been encountered e.g. BIT D,G

Invalid register pair:
An instruction requiring one or more register pairs has been incorrectly formed with a single register e.g. ADD HL,A

Jump out of range:
A relative Jump in a JR or DJNZ instruction exceeds 127.

Label already defined:
An attempt to use a label more than once.

Label not defined:
Reference has been made to a label which does not appear in the symbol table.

Missing bracket:
Brackets are missing or unmatched e.g. IN A,(254

Missing information:
Insufficient information in this instruction e.g. BIT 1

Missing ":
One or both quote marks are missing from a DEFM or DEFI instruction.

Operand not compatible:
The instruction and operand are incompatible e.g. ADD NC

Registers inconsistent:
IX, IY and HL registers have been mixed in one instruction e.g. ADD IX,HL

## Utility Programs.

In addition to the main assembler and editor programs you will find a small suite of utility programs on your tape, cartridge or disc.

### Library.

This is not a program but a source file containing a set of labels to cover the Spectrum system variables and the channel maps. It may be loaded or joined into an Astrum+ file and used in your programs.

### Define.

This program allows you to alter some of the default settings for Astrum+. The options are selected by pressing the number next to them and include;

Text file drive:
Selects the drive on which the text files are normally stored.

Object code drive:
Similarly alters the drive number for saving the assembled code.

Code/Listing separator:
Changes the character used to separate the code values and the mnemonics when an assembly listing is sent to the printer. A 0 switches off the separator.

End of line separator:
Alters the end of line character when output is sent to the printer.

When you have customised Astrum+ then press 5 and the new version is saved to the disc or cartridge in drive 1.

### Exchange.

If you are converting to Astrum+ from a different assembler, this program changes all your old text files into Astrum+ format so that they need not be re-typed.

The program is simple to use; just press the key adjacent to the name of your old assembler and follow the screen prompts for the file names. You are also able to select the drive number for the source text and for the Astrum+ format text. Text may be loaded from tape but not saved to tape.

Pressing 'D' takes you into a utility program which allows the cataloguing, erasing or formatting; do not worry, you cannot accidentally format a cartridge or disc, try selecting the format option to see!

## Copy.

This utility allows you to copy Astrum+ files from cartridge to cartridge, or from disc to disc, with the minimum of effort. You are prompted for the filename to be copied and its new name; just hit the return key to retain the same name. If you want the program to pause for conformation at every drive access, then answer 'Y' to the Prompts? enquiry. Finally, specify the number of pages in the file; if you have a 3 page file, for example, then the program will copy all three pages across automatically.

If you specify a page number with the initial file name, then parts of a multi-page text file may be copied. This makes the copy program a useful utility for inserting pages into a multi-page file. For example, if you have a file 'Fred' with pages 0 to 5 and wish to insert a new page 3. then copy 'Fred     3' to 'Jim' and specify 3 pages (i.e. 3-5). Next copy 'Jim' to 'Fred     4' with 3 pages again; this will, in effect, move pages 3 to 5 of 'Fred' up one place, allowing page 3 to be used for the new text. Finally, erase file 'Jim'.

As with the exchange program the copy program allows you to change the source and destination drives, catalogue or format a cartridge or erase a file. The file copier will copy any print type files, providing that they are less than about 19k, in length.

## Header.

One of the things that neither interface I nor the Opus Discovery do is to tell you the type, length, start address etc. of files on cartridge or disc. This sort of information is essential if you want to back up your software. Header allows you to get this information; all you have to do is to type in the name of the file you wish to investigate and press enter; the file's vital statistics will then be printed lot.

The input stream in initially attached to drive 1 and output is directed to the screen, but both these defaults can be changed by altering the BASIC in lines 2 and 3.

## Creator.

This utility program produces a disassembly file from object code in a form which can be loaded or joined into Astrum+ assembler.

Just load the program and follow the prompts for the output filename, the start address of the code (which may be specified in decimal or hexadecimal) and the end address (the location after the last instruction to be decoded).

Data areas may next be defined. These are areas within the object code which you know to be data rather than opcodes; these will be translated into DEFB pseudomnemonics. Type in an asterisk to the prompt 'Start of data area' if no data exist, or if you have input all the areas.

The text will be created and saved to file on drive 1. References within the text are handled by the prefix L (for label) followed by the PC at that instruction. For example, the text in the left column below, if assembled to #8000 and then processed by Creator, would look something like the right hand column:

```
            LD      B,8                         LD      B,#8
OUTLOOP     LO      A,"*"               L8002   LD      A,#2A
            RST     16                          RST     #10
            DJNZ    OUTLOOP                     DJNZ    L8002
```

There is just one slight difference between the microdrive and Opus versions of the Creator program. A microdrive file produced by Creator may be loaded or joined directly into Astrum+ assembler, whereas a Creator file on disc is in the wrong format and must be taken into Astrum+ by Pressing S/S A from extended made whilst in the editor.

There are three versions of the Creator program which load and run at address 26000, 48000 and 58000, to cope with the different locations of the object code you wish to disassemble. To run the program, select the version you require and type LOAD *"m";1;"C26" (or C48 or C58):RANDOMIZE USR 26000 (or 48000 or 58000). All versions of Creator are less than 4k in length.

## ASTRUM+ MONITOR.

This program is a complimentary package to the Astrum+ assembler, written in the same friendly style to increase your programming pleasure. Three versions of the monitor are supplied which run at 26000, 460000 and 58000, named M26, M46 and M58 respectively. As most machine code is configured to load and run high in RAM, then the M26 version is likely to be used most frequently as stand alone program,

However, the M46 monitor may be loaded in and used to debug code assembled at 56000 without corrupting the assembler; the symbol table normally exists at 46000. This enables code to be easily debugged, modified and re-assembled with the maximum convenience, although you will need to reload the monitor after each assembly as it will have been corrupted by the symbol table.

To run the monitor you should LOAD *”m”;1;”M26” (or M46 or M58); care should be taken not to overwrite the stack i.e. RAMTOP should be more than 6.75k from the load address of the code. Type RANDOMIZE USR 26000 (or 46000 or 580000 to run the monitor and put you into command mode.

The Astrum+ monitor is a command driven program; instead of remembering terse, one or two character mnemonics for the facilities provided, meaningful command words are expected. The cursor appears as an inverse block at the bottom of the screen, marking the current position in the command line editor. The following summarises the editing commands available;

| | |
|---|---|
| DELETE | Deletes the character to the left of the cursor and closes the gap thus produced. |
| C/S 9 | (Graphics) Inserts a space at the cursor, moving subsequent characters to the right. |
| C/S 5 | (Left arrow) Moves the cursor to the left. |
| C/S 8 | (Right arrow) Moves the cursor to the right. |
| C/S 1 | (Edit) Clears the current line and moves the cursor to the start of the line. |
| C/S 2 | (Caps lock) Toggles the caps shift; when In CAPS mode the cursor turns magenta. |
| S/S Q | Moves the cursor to the start of the line. |
| S/5 E | Moves the cursor to the end of the line. |
| S/S ENTER | Moves the cursor to the last character on the line. |
| S/S Y | [ |
| S/S U | ] |
| S/S I | (c) |
| S/S A | ~ |
| S/S S | \| |
| S/S D | \ |
| S/B P | { |
| S/B G | } |

The editor allows the input of text into the bottom Line, which is then processed by the monitor command line interpreter. The following commands are recognised by the program; they may be typed in upper or lower case. The parameters should be separated from the command and from each other by a space.

POKE nn m

Identical in effect to the BASIC keyword, loads the address nn with the number m. All numbers may be entered in the same format as described for the Astrum+ assembler i.e.

| | |
|---|---|
| Nn | decimal |
| #nn | hexadecimal |
| _nn | octal |
| @nn | binary |
| "a" | ASCII |

```
e.g, poke 16389 215
poke #fffe "Z"
```

PEEK nn

Has the same effect as PRINT PEEK nn in BASIC; the contents of the address are displayed.

DOKE nn mm

Double poke; loads the address nn and nn+1 with the two byte value mm in the standard Intel format of low order byte first.

DEEK nn

Double peek; displays the 16 bit number contained at locations nn and nn+1.

DEC

Directs the monitor to print out all numbers in decimal.

HEX

Directs the monitor to print out all numbers in hexadecimal.

BASIC

Exits the monitor and returns to BASIC.

CLS

Clears the screen of all previous displays.

FILL st n m

This starts at address st and loads the subsequent n addresses with the number m e.g.
fill 16384 32 255 fills the addresses 16384 to 16413 Inclusive, the top line of the screen, with 255.

DIS st end

The dis command invokes the disassembler. starting at address st and continuing until address end. If end is not specified then disassembly will continue until #FFFF. When the screen area is filled with the disassembled text it will automatically scroll. The movement can be halted at any time for examination by holding down any key except Q or S; scrolling will resume when the key is released.

The action of the S key is to stop the listing, but this pause continues until another key is pressed. The Q key stops the display and quits the dis option.

DUMP st end

Produces a memory dump from st to end, Each line starts with the address of the first byte of the line; then follows the contents of the next 6 bytes of memory in hexadecimal and finally the ASCII representation of the 6 bytes (in inverse for clarity). Any non printable character is represented by a dot. Control of scrolling and abandonment of the operation is as for the dis command.

ALTER st

Allows you to alter and patch blocks of bytes quickly and easily. The address, starting at st is printed, followed by its content, and the ASCII representation and a cursor. The options available at this point are as follows.

| | |
|---|---|
| n | -typing a number will put the number at that location. |
| ENTER | -advances the display one byte through memory. |
| : | -moves the display back one byte in memory. |
| /nn | -moves to the address nn; equivalent to the sequence q alter nn. |
| q | -quits the alter option. |

COPY src dest len

Copies len bytes from src to dest. e.g. copy 0 16384 32 copies the first 32 bytes from ROM into the first line of the screen area.

FIND st ad n

Searches the memory from st to end and displays all occurences of the byte n.

WFIND at ad nn

Searches the memory for a 16 hit number.

IN port

Fetches the 8 bit number from the I/O address port and prints the value first in hexadecimal or decimal, then in binary. This routine continues cycling until S/S Q is pressed and so invaluable for understanding and debugging I/0 routines, e.g. Try typing 'in 63486' then press the keys 1 to 5.

OUT port n

Puts the byte n to the I/O address port e.g. out 254 4

PRINTER

The output of any of the monitor functions may be echoed to stream 3 by using the command printer at any time. Typing the command again toggles it back to the off position.

**The Front Panel**

The commands so far listed allow you to alter and examine sections of the memory or the I/O. In order to examine the state of the Z80 registers during running of programs, the front panel must be enabled;

PANEL

This command toggles the display of the front panel of the monitor; the screen is split into two windows, the upper section containing the front panel, the lower contains the rest of the monitor output. A typical front panel and the details of its display are as follows;

```
PC=#0000     DI
SP=#nnnn     TOS=#0202B
A=#BO        F=P   Z   I   PE  0   NC
HL=#2D2B     (HL)=#FD
BC=#93BO     (BC)=#C3
DE=#5CDB     (DE)=#00
IX=#03D4     (IX+0)=#04
IY=#5C3A     (IY+0)=#FF
```

PC=#0000 DI

The address of the program counter and the Z80 opcode at that address

| | |
|---|---|
| SP=#nnnn | TOS=#2D2B<br>The address of the stack pointer and the top entry on the stack, the last number put there by a push or call instruction etc. |
| A=NHO | F=P Z 1 PE 0 NC<br>The first number is the contents of the accumulator register, the string is the status of the flags, the positive, zero. half carry, overflow/parity, subtraction and carry flags. Note that, in contrast to some monitors, flags are shown if they are reset as well as set e.g. both C and NC states of the carry flag are displayed. |
| HL=#2D2B | (HL)=#FD<br>The first is the value in the HL register pair, the second is the number contained at the address pointed to by the HL register pair. |
| BC==#93B0 | (BCI=#C3<br>The value in the BC register pair and the contents of that address. |
| DE=#5CDB | (DE)=#00<br>The value in the DE register pair and the contents of that address. |
| IX=#03D4 | (IX+0)=#04<br>The value in the IX registers followed by the contents of the address pointed to. |
| IY=#5C3A | (IY+O)=#FF<br>The value in the IY registers followed by the contents of the address pointed to. |

The contents of individual registers may be manipulated. Next to the PC line is an asterisk; this is the register cursor which responds to the cursor up and down keys, C/S 6 and 7. Several commands are available to alter the registers;

| | |
|---|---|
| MAKE nn | Sets the value in the current 16 bit register -the one next to the register cursor- to the value nn. |
| ADD nn | Adds the value nn to the contents of the current register. |
| SUB nn | Subtracts the value nn from the contents of the current register. |
| HI n | Sets the high byte of the current register pair to n. |
| LO n | Sets the low byte of the current register pair to n. |

The next series of commands enable code to be run whilst inspecting the registers.

JUMP nn

Starts execution at the address nn and is comparable to the Z80 instruction JP nn. It is sat nehessary to set up the PC register prior to this command as this is taken care of. The contents shown on the front panel are loaded into the other registers prior to executing the code.

CALL nn

Starts execution at the address nn. When the routine has finished, however, control is returned to the monitor program and the front panel this show the exit condition of all the registers. Because this is a call, the return address is first pushed onto the stack, so on execution, the stack pointer and the TOS contents will not he the same as on the front panel. On return from the call the return address into the monitor will be popped from the stack.

**Breakpoints.**

Astrum, monitor has two types of breakpoint to facilitate debugging; soft breakpoints, which cause the execution of the program to halt and control to return to the monitor command level whenever the breakpoint is reached, and hard breakpoints, which may be set in RAM or ROM and cause execution to break when the code has been executed a given number at times. The two types will be described in detail and their differences explained.

Soft breakpoints.

Up to 8 Individual soft breakpoints may be set within any piece of program code. The operation of setting a breakpoint stores the instruction at that address and substitutes a call to the monitor. When the call is encountered the original code is replaced, the breakpoint is reset and control is returned to the command mode of the monitor so that further running, single stepping etc, can commence. The commands for setting and clearing the soft breakpoints are as follows;

SET b nn

Sets breakpoint b (in the range 0-7) at address nn.

RESET b

Disables the breakpoint number b; If previously set then the code is changed back to the program code at that point.

STATUS

Lists all the 8 breakpoints and, if they are set, the address of the breakpoint.

The provision of soft breakpoint is usually all that is necessary to debug a short routine. There are, however, times when it would be helpful to move through the code, watching the registers change until you suddenly realise what you have done wrong! There are 3 commands which allow de-bugging at this level:

OBEY

Executes the instruction resident in the program counter. Only one instruction is executed and then control to returned to the monitor command level.

KSTEP

This is equivalent to typing in 'obey' repeatedly; the command is left using S/S Q. The sequence of events is,

Execute the instruction at the PC and update the PC. Display the results (if the front panel is on). Wait for a keypress. If the key is not S/S Q then loop round to the first step. Else finish.

If the instruction at the PC Is a CALL then pressing the E key will execute the instruction and return to the monitor.

STEP [n]

This command is identical to kstep except that it does not wait for a keypress before continuing. If the panel is not switched an then execution speed is in the order of 600 instructions per second, but the constant updating of the front panel if it is active slows this down. The optional parameter n specifies a delay time of n/50 seconds after each instruction; no pause occurs if the parameter is omitted.

**Hard Breakpoints.**

As the single step commands run code in slow motion, soft breakpoints do not work, so up to 8 hard breakpoints are also provided. These can be set in either RAM or in ROM and do not alter any of the code. As the monitor is controlling the flow of the machine code, this allows the provision for a count down breakpoint. By setting this to a number greater than 1, the monitor is instructed not to break execution until the address has been reached for a specified number of times. For example, if the breakpoint was within a loop and had a count value of 19, then the execution would not stop until the loop had been traversed 18 times and was in its 19th execution. Hard breakpoint commands are:

HSET b nn [count]

Set a hard breakpoint number b at address nn. The count down is set to count, or 1 if this parameter is defaulted.

HRESET b

Disables breakpoint number b.

HSTATUS

Displays the hard breakpoints together with their relevant addresses and associated countdowns.

**ACKNOWLEDGEMENTS.**

The writing of Astrum+ was at times a difficult process. As a result I can recommend the following pieces of software and books:

M-Doc by Seven Stars Software.
TransExpress by Romantic Robot.
Extended Art Studio by Rainbird.

ZX Spectrum Programming Manual by Sinclair.
Master Your ZX Microdrives by Andy Pennel, Sunshine Books.
The Complete Spectrum ROM Disassembly by Ian Logan; Melbourne Rouse.

I Would Also like to thank the following people for their constructive criticism and ability to find bugs;

Dr Richard Walker.
Mr Andrew Hewitt,
Mr Phillip Sprotson,
Miss Lisa Chalmets.
Mr Barren Beatson.
Mr Dennis Dryden,

Robert Chafer, 1986.

Bradway software is still an extremely small software house and as such is very user friendly. If you have any problems with a copy of ASTRUM+ that you have bought, or any suggestions for improvement, or even if you find any bugs, please get in touch with Richard Walker by phoning 0742 350225 at any reasonable time (but best chance at the weekend), by letter (SAE please), or via Prestal mailbox number 742350225.

Bradway Software,
33, Conalan Avenue,
Sheffield.
S17 4PG

APPENDIX 1.

**Astrum+ Editor Command Summary.**

| Key | Action. |
|---|---|
| C/S 8 | (right arrow) Cursor right. |
| C/S 5 | (left arrow) Cursor left. |
| C/S 6 | (up a owl cursor up. |
| C/S 7 | (down arrow) Cursor down. |
| C/S 0 | (delete) Deletes character at cursor. |
| C/S 9 | (graphics) Insert space at cursor. |
| C/5 2 | (caps lock) Toggle caps lock. |
| C/S 1 | (edit) Restore current line to previous state. |
| C/S ENTER | Insert line above current line. |
| S/S ENTER | Move cursor to last position on line. |
| TRUE VIDEO | Move screen up 21 lines. |
| INV VIDEO | Move screen down 21 lines. |
| S/S SPACE | Move cursor to next tab position. |
| C/S SPACE | Delete from cursor position to next tab position. |
| S/S I | Delete line. |
| S/S Q | Toggle menu bar. |
| S/S E | Select next m |
| S/S H | Pull down selected menu. |
| S/S Y | [ |
| S/S U | ] |
| S/S A | ^ |
| S/S S | \| |
| S/S F | { |
| S/S G | } |
| EMODE | Enter extended editor commands. |

| Extended Key. | Action. |
|---|---|
| S | Move cursor to the start of line. |
| E | Move cursor to the end of line. |
| L | Move cursor to the top of screen. |
| T | Toggle auto tab. |
| E | Toggle auto format. |
| P | (c) |
| B | Return to basic. |
| S/S A | Join Astrum+ file (Opus only). |
| S/S D | Join Devpac file (microdrive only). |
| S/S R | Reset Astrum+. |
| DELETE | Delete forwards. |
| S/S Q | Delete from start of line to cursor. |
| S/S E | Delete from cursor to the end of line. |

## APPENDIX 2.

**Astrum+ Monitor Command Summary.**

| Key. | Action. |
|---|---|
| C/S B | (right arrow) Cursor right. |
| C/S 5 | (left arrow) Cursor left. |
| C/S 6 | (up arrow) Cursor up. |
| C/S 7 | (down arrow) Cursor down. |
| C/S 0 | (delete) Deletes character at cursor. |
| C/S 9 | (graphics) Inserts a space at the cursor, |
| C/S 1 | (edit) Clears the current line and moves the cursor to the start of the line |
| C/S 2 | (caps lock) Toggles the caps shift when in CAPS mode the cursor turns magenta. |
| S/S Q | Moves the cursor to the start of the line. |
| S/S E | Moves the cursor to the end of the line. |
| S/S ENTER | Moves the cursor to the last character on the line. |
| S/S Y | [ |
| S/S U | ] |
| S/S I | (C) |
| S/S A | ~ |
| S/S S | \| |
| S/S D | \ |
| S/S F | { |
| S/S G | } |

## APPENDIX 3.

**Astrum+ Monitor Command Summary.**

| Command. | Action. |
|---|---|
| PEEK | As BASIC PEEK. |
| POKE | As BASIC POKE. |
| DEEK | Two byte PEEK. |
| DOKE | Two byte POKE. |
| DEC | Numbers printed in decimal. |
| HEX | Numbers printed in hexadecimal. |
| BASIC | Returns to BASIC. |
| CLS | Clears the screen |
| FILL | Fills an of memory with a specified value. |
| DIS | Disassembles an area of memory. |
| DUMP | Produces a hexadecimal and ASCII dump of an area of memory. |
| ALTER | Enables simple patches to be made to a block of code. |
| COPY | Copies code from one location to another. |
| FIND | Finds a given byte within a specified block of code. |
| WFIND | Finds a two byte sequence as above. |
| IN | Polls a port and prints the data from that address. |
| OUT | Writes a byte to an addressed Port. |
| PRINTER | Echos the monitor functions to stream 3. |
| PANEL | Toggles the front panel display. |
| MAKE | Sets the current register to a value. |
| ADD | Adds a value to the contents of the current register. |
| SUB | Subtracts a value from the current register. |
| HI | Sets the high byte of the current register pair to value. |
| LO | Sets the low byte of the current register pair to value. |
| JUMP | Starts execution of code from a specified address. |
| CALL | Calls the routine at a specified address. |
| SET | Sets a soft breakpoint. |
| RESET | Resets a soft breakpoint. |
| STATUS | Displays the status of the soft breakpoints. |
| OBEY | Executes the instruction at the program counter. |
| KSTEP | Single step on each keypress. |
| STEP | Step through the code with optional speed setting. |
| HSET | Sets a hard breakpoint. |
| HRESET | Resets a hard breakpoint. |
| HSTATUS | Displays the status of the hard breakpoints. |