The Sol Assembler Manual

The Sol assembler is an editor + assembler made for the ZX Spectrum Next computer. It creates and imports source files (text files with the .asm file name ending) and assembles them into machine code that may be stored in an executable file (a .tap file including an auto loader) to be launched from the browser menu of the Spectrum Next.

The Command Line

The bottom line of the screen is reserved for input of commands and for system messages to the user. You can switch between the command line and the editor area by pressing 'EDIT'.

Available commands are as follows.

Memory: Displays the amount of used and free memory. Be aware that used memory includes some free memory reserved for ensuring fast editor response, and is measured in terms of whole 8k banks.

Find [string]: Go to next instance of given string in the source code. *Find* with no argument will use the string from the last *find* command if any.

Save [filename]: Saves the source code to an .asm file of given name. If the file name is omitted the source code will be saved using the same name used in previous load or save if such exist, to the current directory. Any existing file of the same name is overwritten. File names may be 10 characters in length, letters or numbers allowed. Do not include the .asm ending. If a file name is specified a file system browser is opened for selecting where to store the file.

Saveas [filename]: Always brings up the file system browser. Otherwise works as the *save* command.

Load [filename]: Loads a source code file into memory. The document is modified to fit the requirements of the assembler, i.e. only control code allowed is 13 (enter), and no code values above 127 either. All illegal codes are simply removed, as are line endings if lines exceed 79 characters (no column scrolling features). If a file name is specified loads the file from the current directory. Otherwise the file system browser is opened for the user to select a file to load.

New: Starts a new project. Be aware that any unsaved source code in the editor will be lost, as the command simply clears the memory for a new project.

Assemble [filename] [-nonext]: You may use the short form *A* as an alternative command. Assembles the code and optionally outputs to given file. If a file name is not specified the code is only assembled and not saved to file. Otherwise the file system browser is opened for selecting where to store the output file. The optional '-nonext' removes support for Spectrum Next specific instructions, i.e. extended instruction set.

Exit: Reboot.

The Editor

The arrow keys are used for getting around in the document. Notice that left + right arrow do not go to new line, and delete will only delete characters on the current line. If you want to delete a line, use line delete (symbol shift + delete).

Symbol shift + arrow keys will scroll up + down as well as page up/down.

Extend + left/right will jump 500 lines back/forward in source code.

Caps shift + arrow keys are used for going to start/end of line as well as start/end of source code.

True Video toggles color theme.

Variable Assignments

All variable assignments (not labels) must be placed at the beginning of the document, following a 'DEFINE:' label. A variable assignment goes like

variable = *value*

or

variable EQU *value*

A variable name must start with a letter and contain only letters and numbers. It must be of at least 3 characters in length. The value may be a decimal, hexadecimal or binary integer. The value specification defines the legal uses of the variable, i.e. a variable assigned the value 345 may not be used where a byte is expected. Use '$', '#' or '0x' prefix for hex and '%' prefix for binary numbers.

I use the term 'variable' here, although they are in fact constants. The real variables are the registers and the memory.

ORG Statements

The source document must contain at least one ORG statement. The first ORG statement must be located right after the variable assignments. The syntax of an ORG statement is

ORG *start address*

where the start address is the address at which the code should be loaded into prior to execution. It defines the values of labels.

Labels

Labels are variables that are assigned a value corresponding to the address at which they appear in code. A label must start at the beginning of a line and ended with a ':'. Nothing else may be on the line except for spaces and comments. The variable naming rules follow that of variable assignments.

Data Statements

DEFB $byte_1$, $byte_2$, …, $byte_n$: A sequence of bytes are stored. A character or a string may also be included in the sequence. In this case the code of the character(s) are stored.

DEFW $word_1$, $word_2$, …, $word_n$: A sequence of 16 bit values are stored.

DEFS *number of bytes* [SET *value*]: Skips a given number of bytes (reserves). The 'SET' addition is optional and allows for setting all skipped bytes to a given value.

The File System Browser

A simple file system browser for selecting file or directory. Move up and down with arrow keys and select with enter. In the case files and subdirectories do not fit on screen, right arrow loads another screen full, and left arrow starts over at the top of the list. This is to compensate for the lack of a scroll feature. Break exits without selecting.

Alternative Key Combinations for Old Spectrum Keyboards

CAPS + 1 : EDIT
CAPS + 2 : CAPS LOCK
CAPS + 3 : TOGGLE COLOR THEME (coming feature)
CAPS + 4 : BACK 500 LINES
CAPS + 5 : LEFT
CAPS + 6 : DOWN
CAPS + 7 : UP
CAPS + 8 : RIGHT
CAPS + 9 : FORWARD 500 LINES
CAPS + 0 : DELETE
CAPS + SPACE : BREAK
SYMBOL + Q : PAGE UP
SYMBOL + E : PAGE DOWN
SYMBOL + I : LINE DELETE
SYMBOL + W : ©