# StenoLite manual

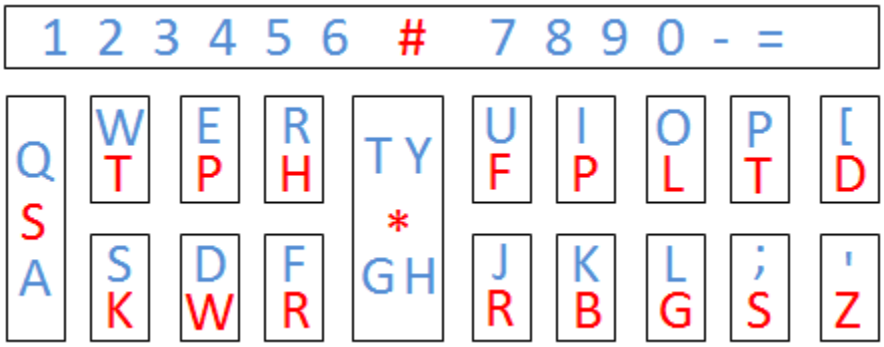
## Table of Contents

## 1. Getting started

There are two ways to get started using StenoLite. The first, and simplest, is to run the StenoLite installer. This will put the StenoLite program into your program files folder, it will create a start-menu shortcut to the program, and it will create a directory for dictionaries in your documents folder. However, if you don't want to install StenoLite (or can't, for whatever reason) it is also possible to run StenoLite without installing it. However, to do so StenoLite must be launched from a folder containing its settings file and a folder entitled "dicts" (no capital letters) where it will search and load dictionaries from. You can find a .zip archive on the same website as the installer that is set up for this way of launching StenoLite.

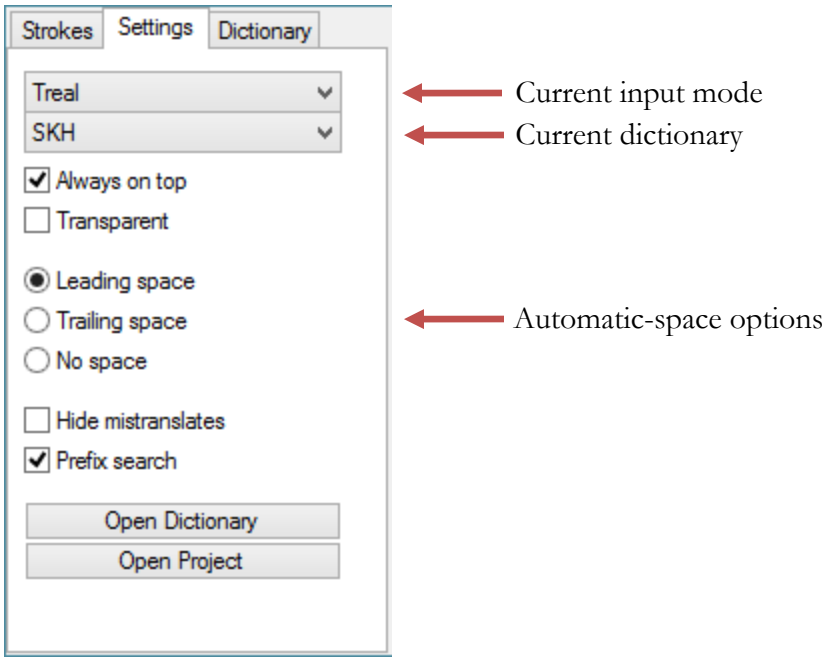
To make sure that StenoLite is functioning correctly, run the program, select the SKH dictionary and Keyboard input options, and then, in a program such as Notepad, press the 'W' key on your keyboard. Under the default settings you should see the word "it" appear. Note that this output will be sent to whichever window is active, and if you have the StenoLite program window itself selected, nothing will appear since StenoLite is not a text editor.

The following image demonstrates the default mapping between qwerty keys, in blue, and stenograph keys, in red.




## 2. Settings and Features

Upon launching StenoLite you should see a window that contains three tabs. The second tab, labeled "Strokes," records what keys you have recently pressed and will display a status message when the program has finished loading. The second tab, labeled "Settings," should display something like the following:




**Input mode**: The current-input-mode selection box shows what type of input StenoLite is currently configured to use. In the above image, StenoLite is set to Off, meaning that it will not respond to any keystrokes. Currently, the other options include keyboard, which will cause StenoLite to process keystrokes from your regular keyboard; Treal, which will cause StenoLite to read keystrokes from a Treal stenograph machine over USB; and the untested serial modes: TX Bolt, Gemini, Passport, and Stentura. The serial modes expect to connect to your machine over a com port (either real or emulated). By default they look for your machine on COM1, but you can change this by adjusting the PORT setting while StenoLite is not running (see section 7 for more on changing the PORT setting).

**Dictionary selection**: The current-dictionary selection box shows which dictionary StenoLite is currently using to translate strokes into text. See the section on adding and modifying a dictionary for more information about how dictionaries work in StenoLite.

**Always on top**: The always-on-top check box will, if checked, keep the StenoLite window above all other windows. This is especially useful if you are using the dictionary tab as you write.

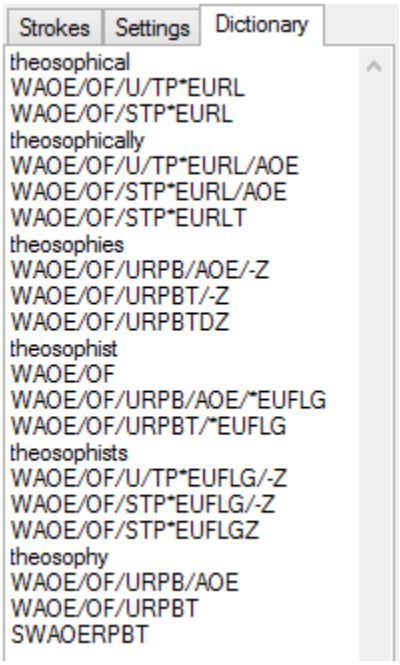
**Transparent**: The transparent check box will, if checked, turn the main StenoLite window partially transparent, allowing you to see what is happening under it. This is most useful in combination with the always-on-top option when, due to monitor space constraints, there is no place to put the StenoLite window that is out of your way.

**Automatic-space options**: The three automatic-space options determine how StenoLite inserts spaces in order to separate words. If the leading-space option is selected, StenoLite generates a space before the words it produces when necessary. If the trailing-space option is selected, StenoLite appends a space after the words is produces when necessary. And, if the no-space option

is chosen, StenoLite will not make any attempt to separate words with spaces, placing that burden upon the user and her dictionary. The author recommends the leading-space option since, at the moment, that option has received the most rigorous testing.

**Hide mistranslates**: The hide-mistranslates checkbox will, if checked, hide any strokes that do not produce words. Normally, a stroke such as STPR*FL would probably be displayed as such, since it is unlikely any word corresponds to it. If the hide mistranslates checkbox were selected, however, no text at all would be produced as a result of such a stroke. The author does not recommend this option, as it can make noticing when you have made a typo much more difficult.

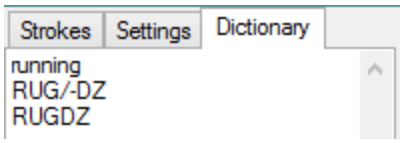
**Prefix search**: Finally, the prefix-search checkbox changes what information is displayed in the dictionary tab. When prefix search is checked, the dictionary tab displays a list of words found in the dictionary beginning with the longest, most recent continuous stretch of text produced* (continuous meaning that it does not contain any spaces, tabs or carriage returns). For example, finger spelling out "theos" results in the following output from the SKH dictionary†:




When the prefix search is not selected, the dictionary tab displays entries in the dictionary that would produce the same text that has been previously output. For example, if "running" is produced using the SKH dictionary, the following result is shown in the dictionary tab:

* If that text is three characters or more in length; the shorter the text the more entries will begin with it, and the search will take longer. Thus, one or two character searches are not performed as they are deemed to be too resource intensive to be worthwhile.

† As these examples reveal, the SKH dictionary is rather "unique." But it is what the author has the rights to distribute free of charge for non-commercial uses.




Note that searches, under both modes, are only conducted while the dictionary tab is selected. If the strokes or settings tabs are selected, the contents of the dictionary tab will not be updated.

The prefix-search option is most useful for finding out how to produce a word you don't know by finger spelling out the first part of it. However, the other option (complete text search) can help you find briefs that can shorten your writing. For example, if you typed out "all of" as two words, and your dictionary included a single entry for "all of," the complete text search mode would display that entry for you, while the prefix search would only display words starting with "of."

# 3. Adding and modifying a dictionary

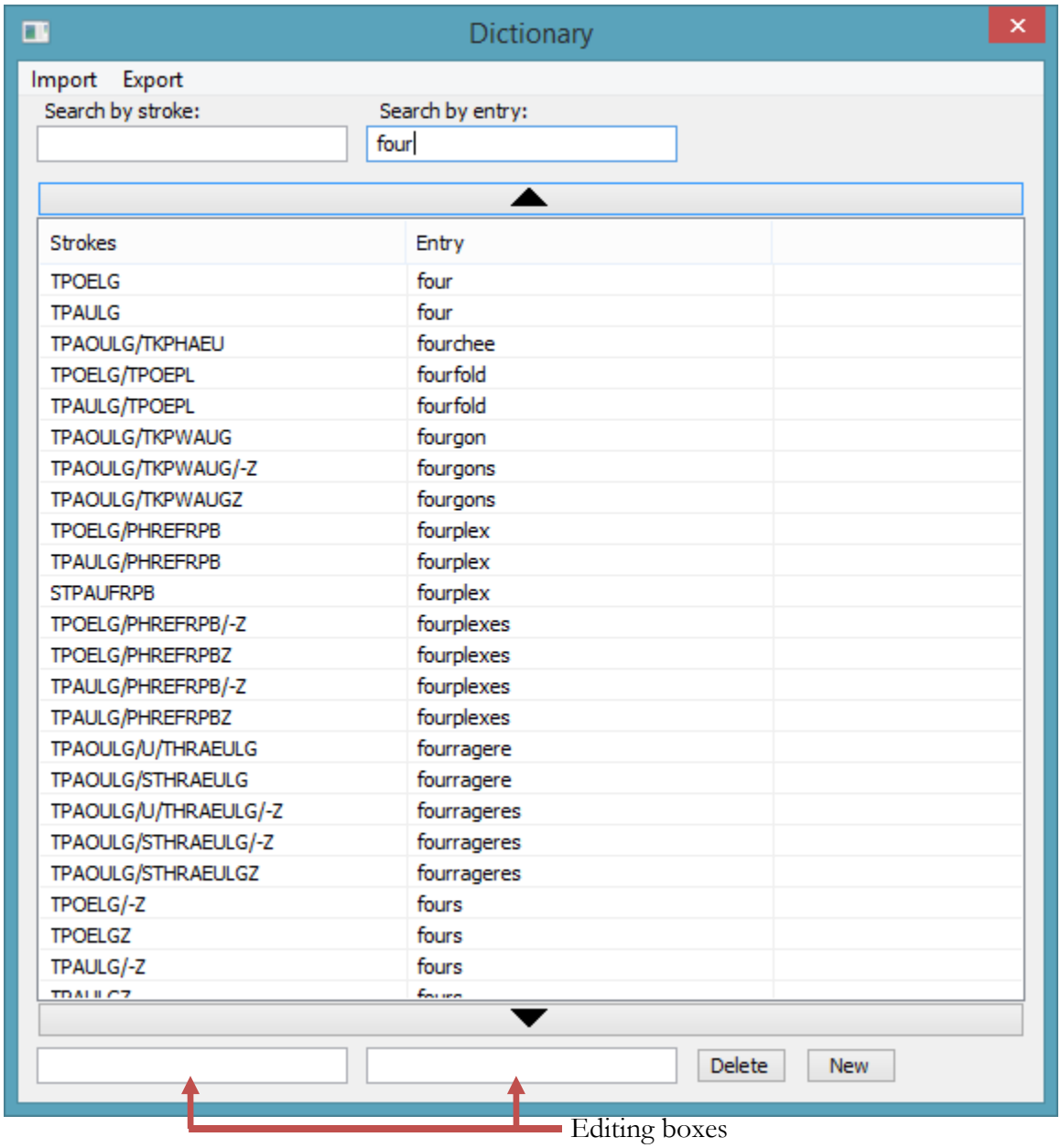
## Adding a dictionary

There are two ways to add a new dictionary to StenoLite. Let's start with the user friendly way. First make a new folder in your dictionary directory with the name you would like your new dictionary to have. Then start StenoLite (if it is already running you will need to close it and then reopen it) and select your new dictionary from the settings tab. Now you can click the open-dictionary button at the bottom of the settings tab, which will open the dictionary editor showing your new, empty dictionary. From there you can add new entries (see below for how to use the dictionary editor) or you can import one or more existing dictionary files in either .json or .rtf format.

Another way to create a new dictionary starts, as with the previous method, by creating a new folder in your dictionary directory. However, before you restart StenoLite, fill that directory with the .json and .rtf files that you would like the create the new dictionary from. When you start StenoLite, the program will begin compiling those files into a new dictionary. However, this process may take some time (the "Loading Finished" message will be displayed in the strokes window when it is completed) and StenoLite will not produce any output until this loading process is finished (users running Windows 7 or above will see the task bar icon show a per-file progress indicator). This is not the preferred way to create a dictionary. It exists for the following reason: if you keep the .json and .rtf files that your dictionary is created from in its directory, then if your dictionary becomes corrupted for any reason, you can simply delete the various other files (everything except settings.txt, user (no extension) and your .json and .rtf files) and your dictionary will be recreated from its original sources plus a list of any new entries you have added from its "user" file (created automatically when you add new entries).

## The dictionary editor

You can launch the dictionary editor by selecting the dictionary you wish to edit and then clicking the open-dictionary button under the settings tab. The following image shows what the dictionary editor looks like




From that top menu you can access the import and export commands. There are two ways to import from an .rtf or .json file. You can choose either "Import new," which will add any items in that file associated with a combination of strokes not currently in the dictionary, or you can choose "Import all" which will overwrite dictionary entries with those from the file when they are associated with the same sequence of strokes. Note that the import capabilities are, at the moment, subject to some limitations. The dictionary editor expects entries in an .rtf file to be in the following format, with each line separated by a carriage return:

`{\*\cxs` ...strokes...`}`...entry contents...

.json files are expected to contain their entries in the following format, with each line separated by a carriage return.

```
"...strokes..." :  "...entry contents...",
```

If you have an existing .json dictionary containing special commands for Plover, such as {&a} as a fingerspelling entry for the letter "a" and {,} for the comma, you can choose the import Plover .json option instead, and an attempt will be made to convert those commands into something that StenoLite understands. However, the process isn't perfect, and sometimes especially complicated commands will be translated incorrectly.

Below the top menu are two text boxes that allow you to search your dictionary either by stroke or by the contents of the entry. Typing in either box will immediately bring your dictionary to the entry that begins with that text or stroke(s), if any. For your convenience, if the strokes search box is selected and StenoLite is running, any input will be sent as an unprocessed stroke to the search box.

Below the search boxes is the main view. You can move your view up or down through the dictionary with the arrow buttons or the mouse scroll wheel. Clicking on an entry in the main view will put its contents into the editing boxes.

When you have selected a dictionary item, you may change its contents though the editing boxes. Any changes made to the entry contents will be immediately reflected in the main view, while clicking on its stroke will bring up a pop up window for you to create a new stroke in. As with the search by stroke box, any input will be sent as an unprocessed stroke for your convenience. You may not change the stroke(s) associated with an entry to stroke(s) already associated with some other entry; delete or modify that entry first. **Caution**: if you use StenoLite to edit a dictionary entry, don't forget to deal with any automatic spaces added before or after the text you create. Such spaces will cause too many spaces to appear between words when using those entries.

Finally, clicking the delete button will remove the currently selected entry from the dictionary and the new button will launch the new entry window (for details consult the description of the \? command).

## 4. The project viewer

You can open the project viewer by clicking on the open project button in the StenoLite settings tab. Either select an existing project file to open, or enter the name of the file you would like to create, and a new project will be opened.

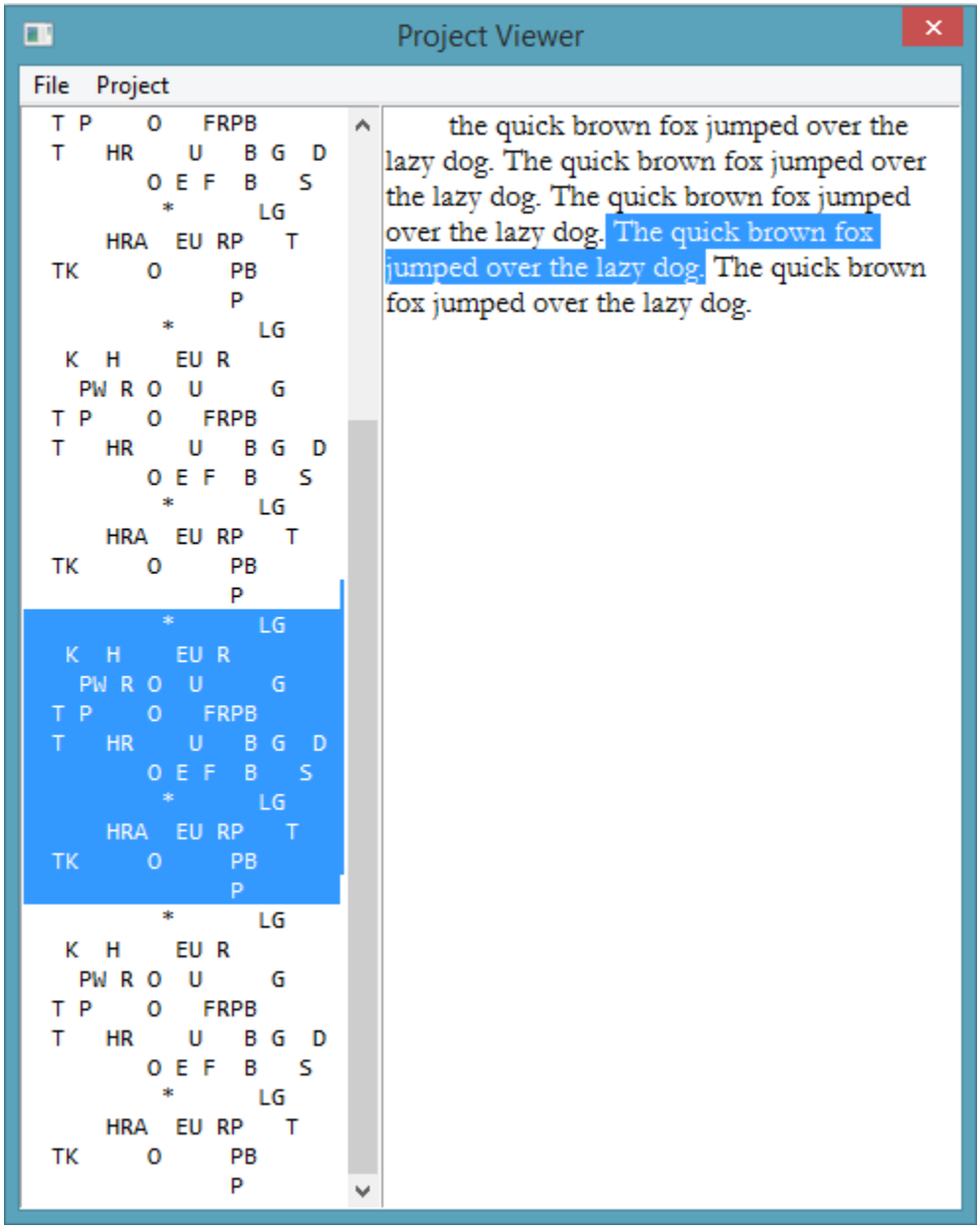
### Project files

There are two kind of project files, .prj files and .srf files. Each project consists of one copy of each of these files (with the same name, differing only in their extension) and can be opened from *either* of these files. The .prj is a condensed version of your project and is generally faster to open, while the .srf contains a record of every stroke (even those that have been deleted) and is slower to open

because the file is recreated by "playing back" these strokes. The really important difference between the two is that the .prj file is only updated when the project viewer closes under normal conditions, while the .srf file is updated while you work. Thus, if the program or your computer crashes while working on a project, the .prj file will not contain any of the changes you have made, but the .srf file will. In such a situation you can recover your work by opening the .srf file, instead of the .prj file as you usually would.

If you believe that the .srf file has become corrupted (or has simply been deleted by accident) you can use the Recreate .srf file comman from the project menu in the project viewer.

## The project window



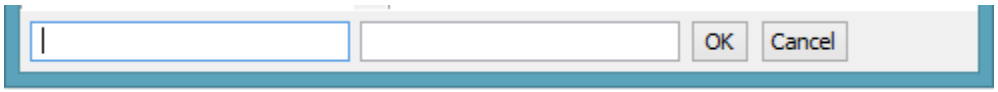
In the above screen shot you can see an example of what an open project might look like. The box on the left contains the strokes behind the text, while the box on the right contains the text generated by your dictionary. You can select text or move the cursor by clicking in either window. Both the strokes and the textual content of the project can be exported to a .txt file by selecting the appropriate option under the file menu.

See section 6 of this manual for how to configure commands such as copy and paste for the project viewer.

### The project dictionary

Each project also contains its own custom dictionary. This dictionary starts out empty, and most strokes will be translated by the current active dictionary. However, any new entries added by the \? command (see section 5 of this manual) will be added to the open project's dictionary, and *not* the currently open dictionary. Thus, they will only affect text created in this project. You can export the project's custom dictionary to a .json file by selecting Export Dictionary from the project menu.

Note that while the project viewer is open the \? command will cause the following bar to appear at the bottom of the project window




Although it looks different, this bar functions identically to the pop-up window usually opened by the \? command.

## 5. Command syntax

The majority of your dictionary entries will consist of text that is intended to be output as-is. Occasionally, however, there is the need for something more complicated, and for those occasions we can embed a number of commands within a dictionary entry.

### Word separation commands

Normally, StenoLite separates each output with a space. We can, however, change that behavior on a per-entry basis.

^: If the ^ character is the first character of an entry, that entry will not be separated from the previous entry by a space. If it appears in a position other than the first, it will not be separated from the next output by a space. A \^ in an entry will produce the ^ as text.

Examples:

`^,` the common entry for a comma

`un^` this entry will attach the un prefix to the next word

`^^` glue the preceding and the following words together

**&**: The & character will connect two entries together without an automatic space only if the first entry contains an & in a position other than the first and the following entry contains an & in the first position. A \& in an entry will produce the & as text.

Example:

`&a&` the letter 'a' as a finger-spelling entry

## Capitalization commands

**\+**: If this command appears in an entry, the first character of the *next* output will be capitalized.

Example:

`^.\+` this will produce a period and capitalize the first word in the next sentence

**\-**: If this command appears in an entry, the first character of the *next* output will be lowercase.

## Special keys

**\t**: sends a press of the tab key.

**\h**: sends a press of the left arrow key.

**\k**: sends a press of the up arrow key.

**\j**: sends a press of the down arrow key.

**\l**: sends a press of the right arrow key.

**\n**: sends a press of the return key.

**\b**: sends a press of the backspace key.

**\x**: sends a press of the escape key.

**\d**: sends a press of the delete key.

**\c**: sends a press of the caps lock key.

****: sends a single \ as text (this is how you include a \ in an entry without it being interpreted as a command).

**\F[...]**: sends the one of the F function keys, determined by the number placed between the square brackets.

Example:

`\F[4]` sends a press of the F4 key

**`\A[...]`**: sends the keys within brackets while holding down the alt key. \A[] sends a single press of the alt key by itself.

Example:

`\A[\F[4]]` sends the alt+F4 combination

**`\C[...]`**: sends the keys within brackets while holding down the ctrl key. \C[] sends a single press of the ctrl key by itself.

Examples:

`\C[v]` sends the ctrl+v combination

`\C[\A[\d]]` sends the ctrl+alt+del combination

**`\S[...]`**: sends the keys within brackets while holding down the shift key. \S[] sends a single press of the shift key by itself.

## StenoLite commands:

**`\M[...]`**: switches to StenoLite to the $n^{th}$ input mode, where n is the number between the brackets.

Example:

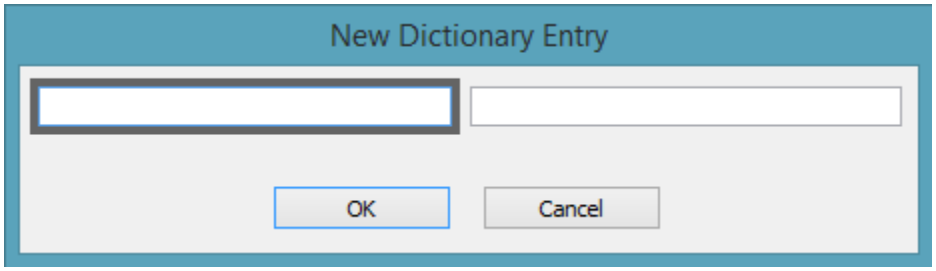
`\M[0]` sets StenoLite to the 0th input mode, namely Off

**`\D[...]`**: switches the current dictionary to the one named by the text between the square brackets (case sensitive).

Example:

`\D[SKH]` switches to using the SKH dictionary

**`\?`**: opens the new entry window, as shown below.




The new entry window places a black box around the control currently selected. The text box on the left is the stroke(s) for the new entry and the text box on the right is the contents of the new

entry. Since you may want to use this feature while typing, the following convenience features are included: while entering new strokes, any input sent to StenoLite is sent as an unprocessed stroke to that text box. Additionally, any excess spaces before or after the text of the new entry are automatically removed (if you want those spaces, you will have to add them by using the dictionary editor). Thirdly, you can use the user-defined tab stroke to move between the controls in this window and use the user-defined enter stroke to press either of the buttons. Finally, pressing the enter stroke while editing a stroke will move you to editing the entry, and pressing the enter stroke while editing the entry will add it to your dictionary and close the window. If there is an existing entry with the stroke(s) you define in this window, it will be overwritten by your new entry. All new entries created in this fashion are also added to the dictionary's "user" (no extension) file as a backup.

### Stack commands:

StenoLite keeps an internal stack of text and commands that you can store entries to and read entries from using a pair of commands.

`\p`: the pop command; removes the most recently added entry from the stack, executing any commands it contains and outputting any text. This is done before any text is generated by the entry \p occurs within, even if \p does not begin that entry.

Example

`\p` the typical usage of the pop command (i.e. not accompanied by anything else)

`\P[...]`: the push command; adds the text between the brackets to the stack.

Examples:

`(^\P[^)]` outputs an opening parenthesis and puts its matching closing parenthesis on the stack.

`\D[pinyin]\P[\D[SKH]]` switches to the pinyin dictionary and puts a command to switch back to the SKH dictionary on the stack.

`[^\P[^\]]` outputs an opening square bracket and puts its matching closing bracket on the stack.

Note that if you wish to push [ or ] characters to the stack you need to escape them with a backslash.

## 6. Dictionary settings

In each dictionary's directory there is a file called settings.txt. By editing this file while StenoLite is closed you can alter some per-dictionary settings.

Each setting is on its own line and all settings have the same format: their name in all capital letters, an equal sign, and then their value. The settings `LONGEST`, `LONGTEXT`, `ITEMS`, and `EXTRAS` should not be altered.

**Unicode support**: StenoLite uses the utf8 encoding for all setting and dictionary files. For most English speakers this will not matter, but if you edit a settings file and add characters outside of the standard English characters, which includes characters with accents, you must ensure that your changes are saved as utf8.

**FORMAT**: This setting determines how strokes are displayed as text and how a string of text, either from an imported file or from user input in the dictionary editor or new entry window, is converted into a stenograph stroke. This setting must consist of a string of 23 characters. The first determines the symbol for the number bar; the second, the character that associated with the first steno key; the third, the character associated with the second steno key; and so on. Stenograph strokes are not stored as text internally, so changing the format setting will result in the entire contents of the dictionary being displayed in the new format the next time you open it, with the following exceptions: strokes appearing in the settings file itself (see subsequent settings) will be interpreted according to the defined format if the FORMAT setting occurs before them in the file. Otherwise, they will be interpreted according to the standard English format. So when changing a dictionary from one format to another, you should insert the new FORMAT setting after any strokes appearing in your settings file using the old format, unless you wish to manually convert them.

Default value: `FORMAT = #STKPWHRAO*EUFRPBLGTSDZ`

**DELETE**: This setting determines what stroke will delete the previous stroke entered.

Default value: `DELETE = *`

**TAB**: This setting determines which stroke is used to switch between controls in the new entry window. It will not cause this stroke to produce a tab character unless you also give it that definition in your dictionary.

Default value: `TAB = T-B`

**RETURN**: This setting determines which stroke is used to confirm your choice and to switch between controls in the new entry window. It will not cause this stroke to produce a carriage return unless you also give it that definition in your dictionary.

Default value: `RETURN = R-RPB`

**NUMBER**: This stroke has two functions. When added to a stroke that consists of a single number, it will cause that number to be output twice. When added to a stroke that consists of two or more numbers, it will cause the order of those numbers to be reversed. For example, under the default settings the stroke 2* would produce 22 and the stroke 2*6 would produce 62.

Default value: `NUMBER = *`

**CUT**: This setting defines a stroke that will act as a cut command in the project viewer; meaning that it will remove any selected text so that it can be moved elsewhere.

Default value: `CUT = -`

**COPY**: This setting defines a stroke that will act as a copy command in the project viewer.

Default value: `COPY = -`

**PASTE**: This setting defines a stroke that will act as a paste command in the project viewer, inserting at the current position any text or strokes previously copied or cut.

Default value: `PASTE = -`

**RPROCESS**: This setting defines a stroke that will perform the reprocess command in the project viewer. This command will regenerate any selected text based on the strokes it was originally generated from. This command is especially useful after you have added or updated a definition and want to update the text of the project to reflect that change.

Default value: `RPROCESS = -`

**LEFT**: This setting defines a stroke that will move the current position of the cursor one word to the "left" (i.e. to the previous word in the project).

Default value: `LEFT = -`

**RIGHT**: As with the LEFT setting above, except this stroke move the cursor in the other direction.

Default value: `RIGHT = -`

**SLEFT**: This setting defines a stroke that extends the current selection to include the previous word, or, if there is no current selection, will select the previous word.

Default value: `SLEFT = -`

**SRIGHT**: As with the SLEFT setting above, except in the other direction.

Default value: `SRIGHT = -`

**SUFFIX**: The suffix setting is the only setting that may appear a variable number of times, and by default no suffix settings are set for a dictionary. The suffix setting associates a stroke with one of a number of pre-defined common English word endings. This association has two effects. First, if that stroke is *not* defined in the dictionary, then typing that stroke will cause that English word ending to be appended to the previous word with some attempt at the correct spelling. For example, if -D was associated with the ed ending, then pressing -D after the word ready would result in readied.

The second function of a suffix-stroke association is to allow StenoLite to guess what an untranslatable stroke should produce when that untranslatable stroke is a product of some entry in

the dictionary plus the suffix stroke folded into its last stroke. It does this by subtracting the defined suffix strokes from any untranslatable stroke, and if the result corresponds to a dictionary entry, it yields that dictionary entry plus the suffix. For example, if -S was associated with the s ending, and RUPB was in the dictionary as "run" but RUPBS was not in the dictionary, RUPBS would still result in the word "runs" being output.

The format for the suffix setting is as follows: `SUFFIX =` stroke`->`suffix

For example:

```
SUFFIX = -S->s
```

Note that there should not be any spaces after the stroke or before the English suffix, and the English suffix must be in lowercase. Currently the following suffixes can be defined: s, ive, ion, ions, th, ly, ing, ings, ed, st, er, ers, ness, able, ment, ful, ist, ists. Also the following limitations currently exist: no attempt to double consonants made (run will not be transformed into running), and currently no attempt is made to guess when ible should be used instead of able.

## 7. Advanced Options

In addition to the dictionary settings files, there is also a settings file for StenoLite as a whole. Unlike the dictionary settings, most of these options are available though the main window. However, you will need to alter the settings file to change the keyboard to stenograph key mappings or the port that the serial modes connect using. This file is named "settings" (without an extension) and is located in your dictionary directory. It can be opened with a text editor such as notepad, and like the dictionary settings files, you should not edit it while StenoLite is running (as your changes will be lost).

**PORT**: This setting tells StenoLite which com port to look for your stenograph machine on. This setting is only used with the four serial modes; it has no impact on the keyboard or Treal modes. The default port setting is as follows:

```
PORT = COM1
```

**MAP**: The only other setting that you must change by editing this file are the `MAP` settings, which determine how the keyboard keys are assigned to stenograph keys. Each map setting assigns a single keyboard key to a single stenograph key and is in the following format `MAP =` keyboard key`->`stenograph key #. The stenograph keys are numbered from the left to right, with S- being key 1, T- key 2, and so on until -Z, which is key 22. The number bar key is key 23. The following example maps the E key to key P-:

```
MAP = E->4
```

**Disabling a key**: Besides adapting the map to a non-qwerty layout, another common use for these settings is to disable one or more keys on the keyboard while StenoLite is active. To do this simply map the desired key(s) to 255, which is guaranteed not to correspond to any stenograph key. To disable the space bar, for example, you can add the following to your settings file:

```
MAP = _->255
```

(_ is how the space bar is represented in this file.)

The default map settings are as follows:

```
MAP = 0->23
MAP = 1->23
MAP = 2->23
MAP = 3->23
MAP = 4->23
MAP = 5->23
MAP = 6->23
MAP = 7->23
MAP = 8->23
MAP = 9->23
MAP = A->1
MAP = C->8
MAP = D->5
MAP = E->4
MAP = F->7
MAP = G->10
MAP = H->10
MAP = I->15
MAP = J->14
MAP = K->16
MAP = L->18
MAP = M->12
MAP = N->11
MAP = O->17
MAP = P->19
MAP = Q->1
MAP = R->6
MAP = S->3
MAP = T->10
MAP = U->13
MAP = V->9
MAP = W->2
MAP = Y->10
MAP = ;->20
MAP = =->23
MAP = -->23
MAP = [->21
MAP = '->22
```

**Mapping special keys**: The map setting represents most keys by the character that is produced by pressing that key with no other modifiers. However, some keys cannot be represented in this way. Below is a table describing how to map these special keys. If you wish to map a key that is not on the table please submit a feature request.

| Key | `MAP` setting |
|---|---|
| Space bar | `_->#` |
| F1 | `f1->#` |
| F2 | `f2->#` |
| F3 | `f3->#` |
| F4 | `f4->#` |
| F5 | `f5->#` |
| F6 | `f6->#` |
| F7 | `f7->#` |
| F8 | `f8->#` |
| F9 | `f9->#` |
| F10 | `f10->#` |
| F11 | `f11->#` |
| F12 | `f12->#` |