# VIP Quick Reference Card

(Based on VIP 3.5 in GNU Emacs 18)

## Loading VIP

Just type `M-x vip-mode` followed by `RET`

## VIP Modes

VIP has three modes: *emacs mode*, *vi mode* and *insert mode*. Mode line tells you which mode you are in. In emacs mode you can do all the normal GNU Emacs editing. This card explains only vi mode and insert mode. **GNU Emacs Reference Card** explains emacs mode. You can switch modes as follows.

| | |
|---|---|
| from emacs mode to vi mode | `C-z` |
| from vi mode to emacs mode | `C-z` |
| from vi mode to insert mode | `i, I, a, A, o, O` or `C-o` |
| from insert mode to vi mode | `ESC` |

If you wish to be in vi mode just after you startup Emacs, include the line:

```
(add-hook 'emacs-startup-hook 'vip-mode)
```

in your `.emacs` file. Or, you can put the following alias in your `.cshrc` file.

```
alias vip 'emacs \!* -f vip-mode'
```

## Insert Mode

Insert mode is like emacs mode except for the following.

| | |
|---|---|
| go back to vi mode | `ESC` |
| delete previous character | `C-h` |
| delete previous word | `C-w` |
| emulate `ESC` key in emacs mode | `C-z` |

The rest of this card explains commands in **vi mode**.

## Getting Information on VIP

Execute info command by typing `M-x info` and select menu item `vip`. Also:

| | |
|---|---|
| describe function attached to the key *x* | `C-h k` *x* |

## Leaving Emacs

| | |
|---|---|
| suspend Emacs | `X Z` or `:st` |
| exit Emacs permanently | `Z Z` or `X C` or `:q` |

## Error Recovery

| | |
|---|---|
| abort partially typed or executing command | `C-g` |
| redraw messed up screen | `C-l` |
| **recover** a file lost by a system crash | `M-x recover-file` |
| restore a buffer to its original contents | `M-x revert-buffer` |

## Counts

Most commands in vi mode accept a *count* which can be supplied as a prefix to the commands. In most cases, if a count is given, the command is executed that many times. E.g., `5 d d` deletes 5 lines.

## Registers

There are 26 registers (`a` to `z`) that can store texts and marks. You can append a text at the end of a register (say `x`) by specifying the register name in capital letter (say `X`). There are also 9 read only registers (`1` to `9`) that store up to 9 previous changes. We will use $x$ to denote a register.

## Entering Insert Mode

| | |
|---|---|
| **insert** at point | `i` |
| **append** after cursor | `a` |
| **insert** before first non-white | `I` |
| **append** at end of line | `A` |
| **open** line below | `o` |
| **open** line above | `O` |
| **open** line at point | `C-o` |

## Buffers and Windows

| | |
|---|---|
| move cursor to **next** window | `C-n` |
| delete current window | `X 0` |
| delete other windows | `X 1` |
| split current window into two windows | `X 2` |
| show current buffer in two windows | `X 3` |
| **switch** to a buffer in the current window | `s` *buffer* |
| **switch** to a buffer in another window | `S` *buffer* |
| **kill** a buffer | `K` |
| list existing **buffers** | `X B` |

## Files

| | |
|---|---|
| **visit** file in the current window | `v` *file* or `:e` *file* |
| **visit** file in another window | `V` *file* |
| **save** buffer to the associated file | `X S` |
| **write** buffer to a specified file | `X W` |
| **insert** a specified file at point | `X I` |
| **get** information on the current **file** | `g` or `:f` |
| run the **directory** editor | `X d` |

## Viewing the Buffer

| | |
|---|---|
| scroll to next screen | `SPC` or `C-f` |
| scroll to previous screen | `RET` or `C-b` |
| scroll **down** half screen | `C-d` |
| scroll **up** half screen | `C-u` |
| scroll down one line | `C-e` |
| scroll up one line | `C-y` |
| put current line on the **home** line | `z H` or `z RET` |
| put current line on the **middle** line | `z M` or `z .` |
| put current line on the **last** line | `z L` or `z -` |

## Marking and Returning

| | |
|---|---|
| **mark** point in register $x$ | `m` $x$ |
| set mark at buffer beginning | `m <` |
| set mark at buffer end | `m >` |
| set mark at point | `m .` |
| jump to mark | `m ,` |
| exchange point and mark | `` ` ` `` |
| ... and skip to first non-white on line | `' '` |
| go to mark $x$ | `` ` `` $x$ |
| ... and skip to first non-white on line | `'` $x$ |

## Macros

| | |
|---|---|
| start remembering keyboard macro | `X (` |
| finish remembering keyboard macro | `X )` |
| call last keyboard macro | `*` |
| execute macro stored in register $x$ | `@` $x$ |

## Motion Commands

| | |
|---|---|
| go backward one character | `h` |
| go forward one character | `l` |
| next line keeping the column | `j` |
| previous line keeping the column | `k` |
| next line at first non-white | `+` |
| previous line at first non-white | `-` |
| beginning of line | `0` |
| first non-white on line | `^` |
| end of line | `$` |
| go to $n$-th column on line | $n$ `\|` |
| go to $n$-th line | $n$ `G` |
| go to last line | `G` |
| find matching parenthesis for `()`, `{}` and `[]` | `%` |
| go to **home** window line | `H` |
| go to **middle** window line | `M` |
| go to **last** window line | `L` |

### Words, Sentences, Paragraphs

| | |
|---|---|
| forward **word** | `w` or `W` |
| **backward** word | `b` or `B` |
| **end** of word | `e` or `E` |

In the case of capital letter commands, a word is delimited by a non-white character.

| | |
|---|---|
| forward sentence | `)` |
| backward sentence | `(` |
| forward paragraph | `}` |
| backward paragraph | `{` |

### Find Characters on the Line

| | |
|---|---|
| **find** $c$ forward on line | `f` $c$ |
| **find** $c$ backward on line | `F` $c$ |
| up **to** $c$ forward on line | `t` $c$ |
| up **to** $c$ backward on line | `T` $c$ |
| repeat previous `f`, `F`, `t` or `T` | `;` |
| ... in the opposite direction | `,` |

# VIP Quick Reference Card

## Searching and Replacing

| | |
|---|---|
| search forward for *pat* | `/` *pat* |
| search backward for *pat* | `?` *pat* |
| repeat previous search | `n` |
| ... in the opposite direction | `N` |
| incremental **search** | `C-s` |
| **reverse** incremental search | `C-r` |
| **replace** | `R` |
| **query** replace | `Q` |
| **replace** a character by another character *c* | `r` *c* |

## Modifying Commands

The delete (yank, change) commands explained below accept a motion command as their argument and delete (yank, change) the region determined by the motion command. Motion commands are classified into *point commands* and *line commands*. In the case of line commands, whole lines will be affected by the command. Motion commands will be represented by *m* below.

The point commands are as follows:

`h l 0 ^ $ w W b B e E ( ) / ? ‘ f F t T % ; ,`

The line commands are as follows:

`j k + - H M L { } G ’`

### Delete/Yank/Change Commands

| | **delete** | **yank** | **change** |
|---|---|---|---|
| region determined by *m* | `d` *m* | `y` *m* | `c` *m* |
| ... into register *x* | `"` *x* `d` *m* | `"` *x* `y` *m* | `"` *x* `c` *m* |
| a line | `d d` | `Y` or `y y` | `c c` |
| current **region** | `d r` | `y r` | `c r` |
| expanded **region** | `d R` | `y R` | `c R` |
| to end of line | `D` | `y $` | `c $` |
| a character after point | `x` | `y l` | `c l` |
| a character before point | `DEL` | `y h` | `c h` |

### Put Back Commands

Deleted/yanked/changed text can be put back by the following commands.

| | |
|---|---|
| **Put** back at point/above line | `P` |
| ... from register *x* | `"` *x* `P` |
| **put** back after point/below line | `p` |
| ... from register *x* | `"` *x* `p` |

### Repeating and Undoing Modifications

| | |
|---|---|
| **undo** last change | `u` or `:und` |
| repeat last change | `.` (dot) |

Undo is undoable by `u` and repeatable by `..` For example, `u...` will undo 4 previous changes. A `.` after `5dd` is equivalent to `5dd`, while `3.` after `5dd` is equivalent to `3dd`.

## Miscellaneous Commands

| | shift left | shift right | filter shell command | indent |
|---|---|---|---|---|
| region | `<` *m* | `>` *m* | `!` *m shell-com* | `=` *m* |
| line | `< <` | `> >` | `! !` *shell-com* | `= =` |

| | |
|---|---|
| emulate `ESC/C-h` in emacs mode | `ESC/C-h` |
| emulate `C-c/C-x` in emacs mode | `C/X` |
| **join** lines | `J` |
| lowercase region | `# c` *m* |
| uppercase region | `# C` *m* |
| execute last keyboard macro on each line in the region | `# g` *m* |
| insert specified string for each line in the region | `# q` *m* |
| check spelling of the words in the region | `# s` *m* |

## Differences from Vi

In VIP some keys behave rather differently from Vi. The table below lists such keys, and you can get the effect of typing these keys by typing the corresponding keys in the VIP column.

| | Vi | VIP |
|---|---|---|
| forward character | `SPC` | `l` |
| backward character | `C-h` | `h` |
| next line at first non-white | `RET` | `+` |
| delete previous character | `X` | `DEL` |
| get information on file | `C-g` | `g` |
| substitute characters | `s` | `x i` |
| substitute line | `S` | `c c` |
| change to end of line | `C` or `R` | `c $` |

(Strictly speaking, `C` and `R` behave slightly differently in Vi.)

## Customization

By default, search is case sensitive. You can change this by including the following line in your `.vip` file.

```
(setq vip-case-fold-search t)
```

| variable | default value |
|---|---|
| `vip-search-wrap-around` | `t` |
| `vip-case-fold-search` | `nil` |
| `vip-re-search` | `nil` |
| `vip-re-replace` | `nil` |
| `vip-re-query-replace` | `nil` |
| `vip-open-with-indent` | `nil` |
| `vip-help-in-insert-mode` | `nil` |
| `vip-shift-width` | `8` |
| `vip-tags-file-name` | `"TAGS"` |

Include (some of) following lines in your `.vip` file to restore Vi key bindings.

```
(define-key vip-mode-map "\C-g" 'vip-info-on-file)
(define-key vip-mode-map "\C-h" 'vip-backward-char)
(define-key vip-mode-map "\C-m" 'vip-next-line-at-bol)
(define-key vip-mode-map " " 'vip-forward-char)
(define-key vip-mode-map "g" 'vip-keyboard-quit)
(define-key vip-mode-map "s" 'vip-substitute)
(define-key vip-mode-map "C" 'vip-change-to-eol)
(define-key vip-mode-map "R" 'vip-change-to-eol)
(define-key vip-mode-map "S" 'vip-substitute-line)
(define-key vip-mode-map "X" 'vip-delete-backward-char)
```

# Ex Commands in VIP

In vi mode, an Ex command is entered by typing:

`:` *ex-command* `RET`

## Ex Addresses

| | | | |
|---|---|---|---|
| current line | `.` | next line with *pat* | `/` *pat* `/` |
| line *n* | *n* | previous line with *pat* | `?` *pat* `?` |
| last line | `$` | *n* line before *a* | *a* `-` *n* |
| next line | `+` | *a* through *b* | *a* `,` *b* |
| previous line | `-` | line marked with *x* | `'` *x* |
| entire buffer | `%` | previous context | `' '` |

Addresses can be specified in front of a command. For example,

```
:.,.+10m$
```

moves 11 lines below current line to the end of buffer.

## Ex Commands

| | |
|---|---|
| mark lines matching *pat* and execute *cmds* on these lines | `:g /`*pat*`/` *cmds* |
| mark lines *not* matching *pat* and execute *cmds* on these lines | `:v /`*pat*`/` *cmds* |
| **move** specified lines after *addr* | `:m` *addr* |
| **copy** specified lines after *addr* | `:co` (or `:t`) *addr* |
| **delete** specified lines [into register *x*] | `:d` [*x*] |
| **yank** specified lines [into register *x*] | `:y` [*x*] |
| **put** back text [from register *x*] | `:pu` [*x*] |
| **substitute** *repl* for first string on line matching *pat* | `:s /`*pat*`/`*repl*`/` |
| repeat last substitution | `:&` |
| repeat previous substitute with previous search pattern as *pat* | `:~` |
| **read** in a file | `:r` *file* |
| **read** in the output of a shell command | `:r!` *command* |
| write out specified lines into *file* | `:w` *file* |
| write out specified lines at the end of *file* | `:w>>` *file* |
| write out and then quit | `:wq` *file* |
| define a macro *x* that expands to *cmd* | `:map` *x cmd* |
| remove macro expansion associated with *x* | `:unma` *x* |
| print line number | `:=` |
| print **version** number of VIP | `:ve` |
| shift specified lines to the right | `:>` |
| shift specified lines to the left | `:<` |
| **join** lines | `:j` |
| mark specified line to register *x* | `:k` *x* |
| **set** a variable's value | `:se` |
| run a sub**shell** in a window | `:sh` |
| execute shell command *command* | `:!` *command* |
| find first definition of **tag** *tag* | `:ta` *tag* |



For VIP 3.5 with GNU Emacs version 18

Written by Masahiko Sato,

using refcard layout designed by Stephen Gildea.



For more Emacs documentation, and the TEX source for this card, see the Emacs distribution, or `https://www.gnu.org/software/emacs`