vim for the Commodore 64

User Guide and Command Reference

A small modal, vi-style text editor written in Power C (K&R C) with machine-language helpers, running on a stock C64 with a 1541/1581-style disk drive on device 8.



The editor running in VICE: 24 text rows, the reverse-video cursor block and the status row (file name on the left, line 4 of 7 on the right).

Contents

1. Overview and limits
2. Building and running
3. Starting the editor
4. The screen
5. Normal mode commands
6. Insert mode
7. Colon (:) commands
8. Status messages
9. Differences from real vi/vim
10. Worked examples
11. Source layout

1. Overview and limits

The editor loads a PETSCII sequential (SEQ) file from disk into one memory buffer, shows it on the 40-column screen, lets you move around and edit it with vi-style keys, and writes it back. It is a deliberately small subset of vi: the commands listed in this guide are the complete set.

Item	Limit / behaviour
Text buffer	19,000 bytes (one `malloc` block). A file larger than about 18,999 bytes is truncated on load without warning.
Lines	Up to 599 lines. Opening a new line beyond that is refused.
Line length	80 characters. Longer lines are split at 80 on load (you are told *long lines split at 80*); typing past 80 shows *line full (80)*.
Screen	24 text rows of 40 columns plus a status row. The view scrolls vertically and horizontally to follow the cursor.
File format	PETSCII, lines separated by carriage return ($0D). CR LF and bare LF are converted to CR on load; a Ctrl-Z ($1A) byte ends the file. A final CR is always added on save.
Disk	Device 8, drive 0. Files are read as `name,S,R` and written as `@0:name,S,W` (replace).
File names	Up to 20 characters, stored in lower case.
Undo / search / counts	Not supported (see section 9).

2. Building and running

Everything lives in the `vim/` folder of the repository. The build needs VICE 3.10 installed at `C:\Program Files\GTK3VICE-3.10-win64` and the Power C system disk at `reference\POWERc.d81` (never modified; the build works on a copy called `work.d81`).

Build

```
build.bat                              build vim on work.d81
build.bat vim -Full                    fresh disk, recompile every unit
build.bat vim -Run -Keys "vim.c\njjj"   build, then run and type keys
```

The build runs one VICE session that compiles each changed unit (`vim, vscreen, vview, vedit, vml`) with `cc`, links them with `link -s`, and produces the program `vim` on `work.d81`. Only units whose source changed since the last build (tracked in `work.stamp`) are recompiled. Screenshots of each step go to `shots\`.

build.ps1 option	Meaning
`-Prog name`	Program to build: `vim` (default) or `mltest`.
`-Full`	Ignore the stamps: start from a fresh work disk and recompile everything.
`-Run`	After linking, leave the Power C shell, load the program and run it.
`-Keys text`	Keystrokes typed into the running program (`\n` = RETURN).
`-Shell text`	Extra commands for the Power C shell instead of running.
`-Delay ms`	Wall-clock milliseconds per keystroke (default 900).
`-Pad / -Tail n`	Spare RETURNs while a compile / the linker is working (20 / 8).
`-Settle s`	Extra seconds to keep running after the last key.

Run

Double-click or run `run.bat`: it autostarts `vim` from `work.d81` in `x64sc`. On a real C64 (or any emulator) with the disk in device 8:

```
LOAD"VIM",8,1
RUN
```

> **Keyboard tip for VICE:** with the default keymap the PC **Esc** key is the C64 **RUN/STOP** key and **Backspace** is **INST/DEL**, so the editor feels close to vim on a PC keyboard. Upper-case letters such as G, A, I and O are typed with Shift.

3. Starting the editor

The screen clears to white text on blue and the status row asks:

```
file: _
```

You type	What happens
A name, then RETURN	The file is loaded from device 8 and you are in normal mode at line 1.
Just RETURN (or RUN/STOP)	You get an empty buffer named *[no name]*. Give it a name later with **`:w name`**.
A name that is not on the disk	You are asked `not found - create it? (y/n)`. **y** starts an empty buffer that keeps the name (the status row says *new file*; the file is created on the first **`:w`**). **n** or RUN/STOP asks for a name again.

While typing at the *file:* prompt (and on the colon command line) DEL erases the last character and RUN/STOP cancels. Shifted letters are folded to lower case.

4. The screen

Rows 0-23 show the text. The character under the cursor is drawn in reverse video. Row 24 is the status row, also in reverse video:

```
left part (26 chars)          +    line/total
vim.c                         +    12/240
```

Part	Shows
Left	A one-shot message if there is one (section 8); otherwise `-- insert --` in insert mode; otherwise the file name, or `[no name]`.
Column 28	+ when the buffer has unsaved changes.
Right	Current line number / total lines.

The editor switches the VIC-II to the upper/lower-case character set, so unshifted letters show as lower case and shifted letters as capitals.

5. Normal mode commands

Normal mode is where the editor starts and where RUN/STOP returns you. Commands are single keys (or the two-key **dd** and **gg**); they take no count prefix.

Moving

Key	Action
h or CRSR left	Left one character.
l or CRSR right	Right one character (stops at end of line).
j or CRSR down	Down one line.
k or CRSR up	Up one line.
0	Start of the line.
$ (Shift+4)	End of the line.
gg	First line of the file.
G (Shift+g)	Last line of the file.

When moving up or down onto a shorter line the cursor is pulled back to that line's end. The screen scrolls so the cursor is always visible, including sideways for lines longer than 40 columns.

Entering insert mode

Key	Inserts
i	Before the cursor.
a	After the cursor.
I (Shift+i)	At the start of the line.
A (Shift+a)	At the end of the line.
o	On a new, empty line opened below the current one.
O (Shift+o)	On a new, empty line opened above the current one.

Changing text

Key	Action
x	Delete the character under the cursor.
dd	Delete the current line and keep a copy of it (up to 80 characters) in the yank buffer. Pressing **d** followed by any other key cancels.
p	Put the last line deleted with **dd** on a new line below the cursor. The copy stays, so **p** can be repeated; **dd** then **p** moves a line down.
:	Open the colon command line (section 7).

Any other key in normal mode is ignored.

6. Insert mode

The status row shows `-- insert --`. Every printable key is inserted at the cursor, including shifted letters and the C64 graphics characters.

Key	Action
RUN/STOP (Esc in VICE)	Back to normal mode; the cursor steps one left, as in vi.
RETURN	Split the line at the cursor; the rest moves to a new line below.

Key	Action
INST/DEL (Backspace in VICE)	Delete the character before the cursor. At the start of a line, join it onto the previous line (refused with *joined line would pass 80* if the result would be too long).
Cursor keys	Move without leaving insert mode.

A line cannot grow past 80 characters: further typing is refused with *line full (80)*.

7. Colon (:) commands

Press `:` in normal mode, type the command on the status row and press RETURN. DEL edits; RUN/STOP or an empty line cancels.

Command	Action
`:w`	Write the buffer to its file name, replacing the old copy.
`:w name`	Write to *name*; from now on the buffer is called *name*.
`:wq` / `:wq name`	Write, then quit.
`:x`	Write (always, even if nothing changed), then quit.
`:q`	Quit. Refused with *unsaved - :q! quits* if there are unsaved changes.
`:q!`	Quit and throw away unsaved changes.

Quitting restarts the C64. Power C uses zero-page locations BASIC depends on, so instead of returning to a damaged BASIC the editor jumps to the KERNAL reset routine ($FCE2). You see the normal power-on banner and READY. Anything in memory is gone; only what you wrote to disk survives.

Commands are matched on their letters only, so `:Q` works like `:q`. Anything else gives *unknown command*.

8. Status messages

Messages appear on the left of the status row and disappear at the next key.

Message	Meaning
new file	The name you gave is not on the disk yet; `:w` will create it.
long lines split at 80	The file had lines over 80 characters; they were broken into several lines.
written	The save succeeded.
cannot write	The file could not be opened for writing (no disk, drive not present, ...).
no file name	`:w` on a buffer with no name: use `:w name`.
unsaved - :q! quits	`:q` with unsaved changes.
unknown command	The colon command was not one of w, wq, x, q, q!.
line full (80)	Insert refused: the line is already 80 characters long.
joined line would pass 80	Backspace at the start of a line refused: the joined line would be too long.
out of memory	The 19,000-byte buffer could not be allocated (shown at start-up).

9. Differences from real vi/vim

Area	This editor
Escape	RUN/STOP (the C64 has no Esc key; VICE maps PC Esc to it).
Counts	No numeric prefixes: **`3j`** or **`5dd`** do not work (digits other than **`0`** are ignored).
Undo / redo	None - save often.
Yank / registers	One unnamed register filled only by **`dd`**; no **`yy`**, no named registers, no **`P`**.
Motions	No word motions (w, b, e), no search (/, ?, n), no **`:number`**.
Other	No visual mode, no **`.`** repeat, no **`r`**, **`c`**, **`J`**, no **`:e`** (quit and run again to open another file).

10. Worked examples

Create a new file

```
RUN                          start the editor
notes RETURN                 file: notes
y                            not found - create it? (y/n)
i  Hello, C64!  RETURN       type some text
   second line
RUN/STOP                     back to normal mode
:wq RETURN                   write and quit (the C64 restarts)
```

Fix a typo and move a line

```
j j j                        down to the line
l l l l x                    move right, delete one character
dd                           cut the line
p                            put it back one line lower
:w RETURN                    save; status row says "written"
```

Save under a new name and abandon edits

```
:w backup RETURN             write a copy as "backup"
x x x                        more edits ...
:q RETURN                    "unsaved - :q! quits"
:q! RETURN                   quit without saving
```

Automated test run in VICE

```
build.bat vim -Run -Keys "vim.c\njjj"
```

Builds, starts the editor, opens `vim.c` from the work disk and moves down three lines; the final screen is saved to `shots\run.png`.

11. Source layout

File	Contents
vim.c	`main()`, the key dispatcher for both modes, the file prompt, loading and the colon commands.
vedit.c	Buffer editing: insert, delete, newline, backspace/join, **`dd`**/**`p`**, cursor keys, saving.
vview.c	Scrolling, partial/full repaint and the status row.
vscreen.c	Screen set-up and tear-down, low-level drawing, keyboard read from the KERNAL buffer ($C6/$0277).

File	Contents
vml.s / vmlwrap.c	6510 helpers copied to $C000: full-screen paint, row paint, block move, line-index adjust, key pop. `build-ml.ps1` assembles them into the generated `vml.c`.
*.pet	PETSCII copies of the sources made by the build; the `.c` files are authoritative.
build.bat / build.ps1	One-session build on `work.d81`.
run.bat	Start the built editor in VICE.
NOTES.md	Hard-won notes on Power C, sys(), the keyboard and VICE automation.

Memory use: Power C program and heap below $C000 (the text buffer is a 19,000-byte heap block), machine code at $C000, its parameter block at $C400, zero-page save area at $C410, and the C stack just below $D000. Screen at $0400, colour RAM at $D800.