Build Support 2 - Link Porting to Vial

1. Porting guide
2. Build support 2 - Port to Vial

   Information

   `vial-qmk` provides several examples of most common microcontroller configurations set up for Vial usage. You can check them out at `vial-qmk/keyboards/vial_example`.

   Information

   If an already ported firmware exists, there is no need to redo the porting process. Just use the existing one! If your keyboard has a keymap folder named 'vial' it is already ported! `vial-qmk/keyboards/`

The second part of this tutorial will guide you through porting your keyboard to Vial. **It assumes you will be adapting source code that was written for and known to work with a recent version of QMK.** Keyboard designers are strongly suggested to create a `default` QMK keymap first to familiarize themselves with the firmware creation process and correct any errors.

If you only have precompiled firmware, you will need to obtain the source code for it before proceeding. You have the right to ask the maker of your keyboard for this. Reverse-engineering hardware or software is beyond the scope of this guide.

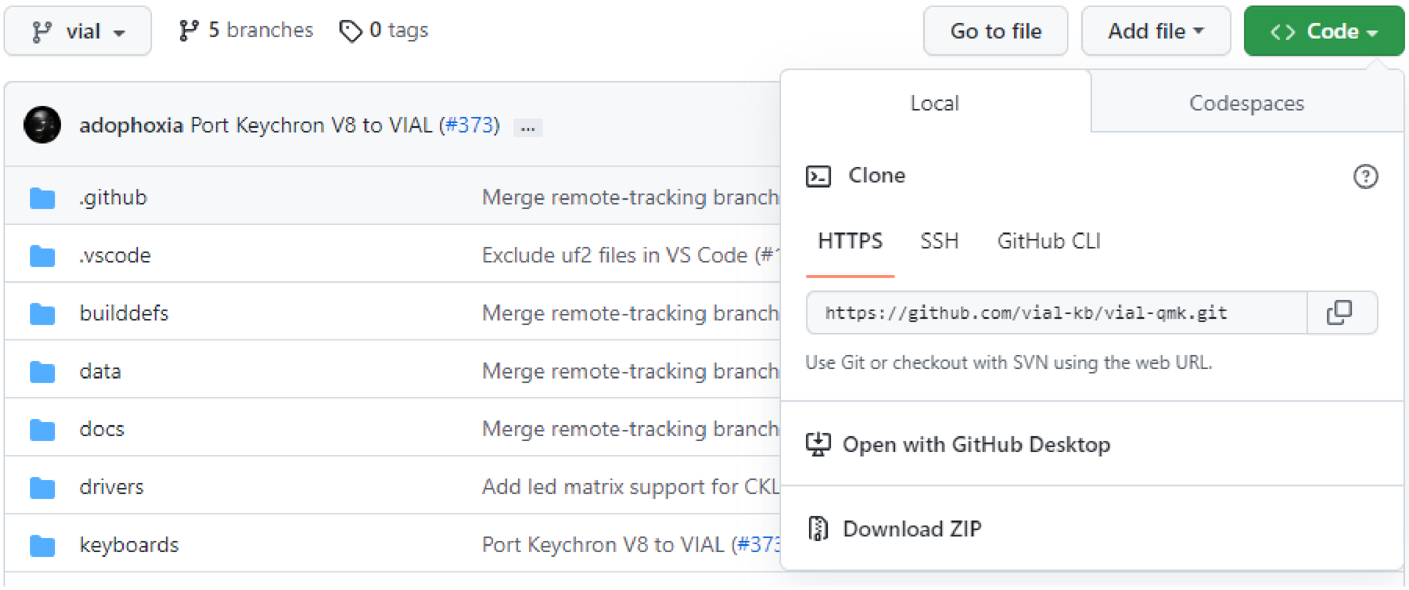
> Information
>
> Compiling Vial firmwware requires some basic knowledge of how to use a command line environment—a bit more than just `qmk compile`. If you are unfamiliar with commands such as `cd` and `pwd`, please read through this basic primer to learn the commands you'll need to know.

1. Prepare Your Build Environment

> Important
>
> You will need a QMK environment in order to build firmware for Vial. This is not the same thing as QMK Toolbox. If you do not have this, you must first follow QMK's guide here. Existing QMK environments may need to be updated, but there is no need to uninstall or remove anything. You can simply move to the appropriate folder when compiling firmware for QMK or Vial-QMK.

Step by step guide:

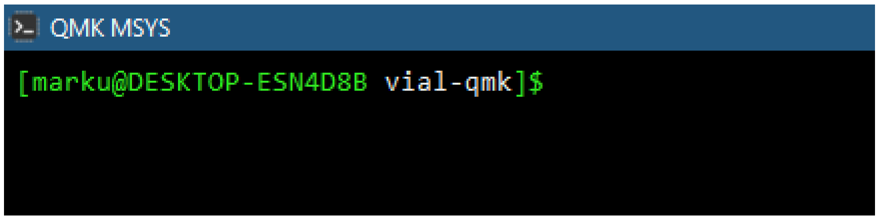
1. Clone the latest version of the repository from https://github.com/vial-kb/vial-qmk into a new folder. This folder should be located outside of any existing QMK repository folders. **Nesting `vial-qmk` inside `qmk_firmware` will cause issues, avoid this!**
    - If you are unfamiliar with git, and want a GUI version, install the GitHub Desktop version for your OS. Go to https://github.com/vial-kb/vial-qmk and select “Open in Github Desktop”. Follow the guided download/cloning.
    - **Reference Guide:** Run `git clone https://github.com/vial-kb/vial-qmk`.


vial
5 branches
0 tags
Go to file
Add file
Code
adophoxia Port Keychron V8 to VIAL (#373)
.github
Merge remote-tracking branch
.vscode
Exclude uf2 files in VS Code (#
builddefs
Merge remote-tracking branch
data
Merge remote-tracking branch
docs
Merge remote-tracking branch
drivers
Add led matrix support for CKL
keyboards
Port Keychron V8 to VIAL (#373
Local
Codespaces
Clone
HTTPS
SSH
GitHub CLI
Open with GitHub Desktop
Download ZIP

```
https://github.com/vial-kb/vial-qmk.git
```

Use Git or checkout with SVN using the web URL.

1. Install the prerequisites for compilation.
    - Open the QMK command line environment and navigate to your new `vial-qmk` folder to run the above command. Refer to this guide on how-to. In the image below, notice how it says ‘vial-qmk’ at the end of the prompt. This is your confirmation you have found the correct directory.
    - **Reference Guide:** Run `make git-submodule` in your new `vial-qmk` directory to clone the git submodules.



```
[marku@DESKTOP-ESN4D8B vial-qmk]$
```

1. Verify that your installation is complete by running `qmk doctor`.
    - If you have added a new keyboard folder as part of your port, you will see `Git has unstashed/uncommitted changes`. Other than that, the only warning should be `The official repository does not seem to be configured as git remote "upstream"`. (This is fine because Vial-QMK is not the official QMK repository.)

2. Test your Vial-QMK installation by compiling the `default` keymap for your keyboard.
    - Run `make path/to/your/keyboard:default`. Note that the path does *not* include the folder `vial-qmk/keyboards/` nor does it have a leading slash. It should start with the name of the keyboard or designer, e.g. `make vial_example/vial_atmega32u4:default` If the `default` keyboard doesn't build, nothing based on it will either! Code copy-and-pasted in from other repositories may need to be modified based on the version of QMK it was created for.

Note

Running `make` is not quite the same as running `qmk compile`. Running `qmk compile` in the wrong directory will default to compiling QMK firmware, not Vial firmware. To avoid any potential confusion, use `make`.

2. Create a new `vial` keymap

Copy the existing `keymaps/default` folder to `keymaps/vial` with the content intact. Leave the `keymaps/default` folder in place; do not modify or remove it.

Important

When porting to Vial, the resulting keymap should ALWAYS be named `vial` and nothing else! This avoids any confusion about if the keyboard has already been ported or not.

3. Enable Vial in your rules file

Create a new file (or edit an existing one) under `[keyboard_name]/keymaps/vial/rules.mk` with the following contents:(example)

```
VIA_ENABLE = yes
VIAL_ENABLE = yes
```

Important

Both of these rules are required to be in the keymap `rules.mk`. Do not remove `VIA_ENABLE`.

Important

Do NOT include either `VIA_ENABLE` or `VIAL_ENABLE` on a keyboard level `rules.mk`, the structure in layers allows the code in the `keyboards` folder to be backwards compatible with QMK.

4. Move JSON so Vial can find it

Place your keyboard definition JSON (either one made in step 1 of this tutorial or downloaded from the VIA keyboards repository) under `[keyboard_name]/keymaps/vial/vial.json` so that Vial build process can find it. (example)

> Important
>
> This file must be named `vial.json`. It cannot have any other name.
>
> The Vial keyboard definition (`vial.json`) is not the same as an `info.json` file. You cannot copy an `info.json` file to use as the `vial.json`.

5. Generate and add unique keyboard ID

The ID generated is unique for the firmware to a keyboard type and model; it does not need to be unique for each individual keyboard or firmware compiled.

From the root of vial-qmk, run `python3 util/vial_generate_keyboard_uid.py` in order to generate a unique Vial keyboard ID:

```
python3 util/vial_generate_keyboard_uid.py
#define VIAL_KEYBOARD_UID {0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX}
```

Create a new file under `[keyboard_name]/keymaps/vial/config.h` and add the following contents to it:

```
/* SPDX-License-Identifier: GPL-2.0-or-later */

#pragma once

#define VIAL_KEYBOARD_UID {0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX}
```

The last line should match what you got in the previous step. (example)

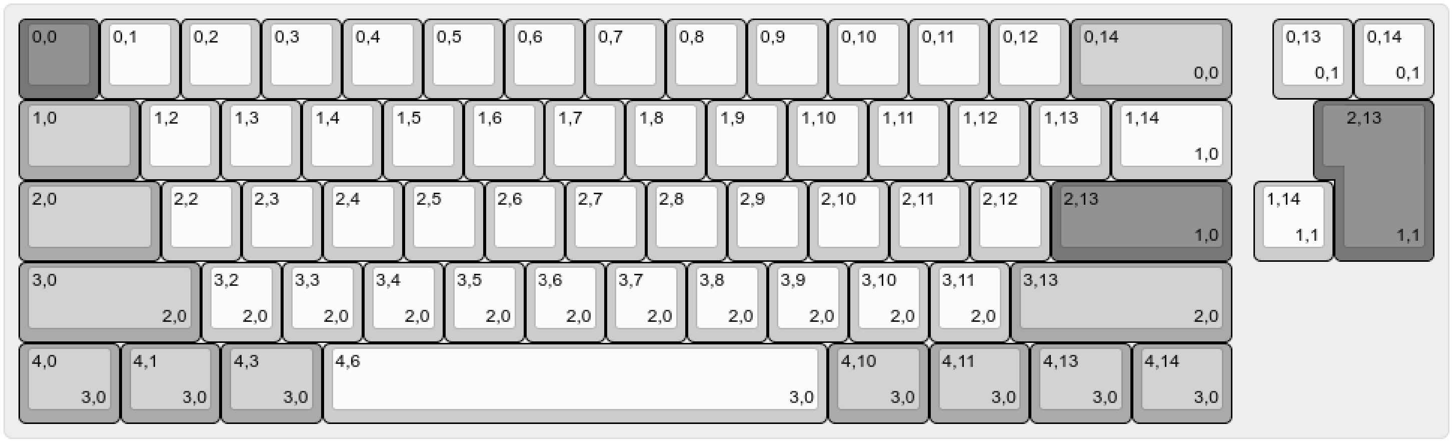
6. Set up a secure unlock combination

Vial needs a key combination in order to protect the user from a malicious host computer unknowingly changing security-sensitive settings, such as flashing a malicious firmware; for more information see here.

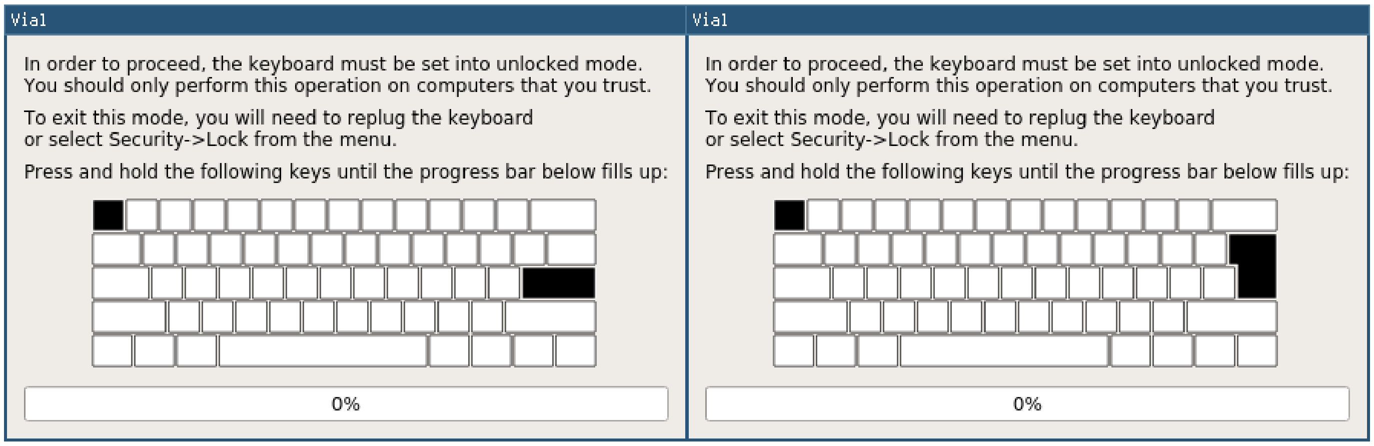
If you do not want to utilize this feature, you should set `VIAL_INSECURE = yes` in your `keymaps/vial/rules.mk`. While you can distribute the resulting firmware to your users and will not lose any Vial functionality, note that keyboards which enable `VIAL_INSECURE` will not be accepted to the main `vial-qmk` repository.

For keyboards that do not define `VIAL_INSECURE`, proceed to configure `VIAL_UNLOCK_COMBO_ROWS` and `VIAL_UNLOCK_COMBO_COLS` definitions:

- You should configure a combo of at least 2 keys
- Suppose this is your KLE and you want to configure a combo of Escape+Enter:



- The Escape key is located at [0, 0] and the Enter key is at [2, 13]
- So you should set in your `keymaps/vial/config.h` right below the `VIAL_KEYBOARD_UID` line (example):
    - `#define VIAL_UNLOCK_COMBO_ROWS { 0, 2 }`
    - `#define VIAL_UNLOCK_COMBO_COLS { 0, 13 }`
- Note that this feature works with multi-layout keyboards, however you should make sure that the keys you pick appear in every possible layout:



In order to proceed, the keyboard must be set into unlocked mode.
You should only perform this operation on computers that you trust.

To exit this mode, you will need to replug the keyboard
or select Security->Lock from the menu.

Press and hold the following keys until the progress bar below fills up:

0%

In order to proceed, the keyboard must be set into unlocked mode.
You should only perform this operation on computers that you trust.

To exit this mode, you will need to replug the keyboard
or select Security->Lock from the menu.

Press and hold the following keys until the progress bar below fills up:

0%

After you flash the firmware, check that the function works correctly by activating the “Security->Unlock” menu.

7. Compile Vial firmware for your keyboard

Compiling and flashing can be done in the same way as QMK. For example, to compile a `vial` keymap for a keyboard located under `keyboards/xyz/xyz60`, run `make xyz/xyz60:vial` from the root `vial-qmk` directory. If at this point you’re having issues making the firmware fit (running out of flash, RAM or EEPROM), see this guide for how to reduce Vial firmware size.

Done!

You now should be able to compile the firmware, flash it, and have your keyboard auto-detected by Vial.

Did you run into a problem? You can get porting support by joining Vial discord.