# How to Contribute

?? First off, thanks for taking the time to read this and contribute! ??

Third-party contributions help us grow and improve QMK. We want to make the pull request and contribution process useful and easy for both contributors and maintainers. To this end we've put together some guidelines for contributors to help your pull request be accepted without major changes.

\* [Project Overview](#project-overview)

\* [Coding Conventions](#coding-conventions)

\* [General Guidelines](#general-guidelines)

\* [What does the Code of Conduct mean for me?](#what-does-the-code-of-conduct-mean-for-me)

## I Don't Want to Read This Whole Thing! I Just Have a Question!

If you'd like to ask questions about QMK you can do so on the [OLKB Subreddit](https://reddit.com/r/olkb) or on [Discord](https://discord.gg/Uq7gcHh).

Please keep these things in mind:

\* It may take several hours for someone to respond to your question. Please be patient!

\* Everyone involved with QMK is donating their time and energy. We don't get paid to work on or answer questions about QMK.

\* Try to ask your question so it's as easy to answer as possible. If you're not sure how to do that these are some good guides:

\* https://opensource.com/life/16/10/how-ask-technical-questions

\* http://www.catb.org/esr/faqs/smart-questions.html

# Project Overview

QMK is largely written in C, with specific features and parts written in C++. It targets embedded processors found in keyboards, particularly AVR ([LUFA](http://www.fourwalledcubicle.com/LUFA.php)) and ARM ([ChibiOS](http://www.chibios.com)). If you are already well versed in Arduino programming you'll find a lot of the concepts and limitations familiar. Prior experience with Arduino is not required to successfully contribute to QMK.

<!-- FIXME: We should include a list of resources for learning C here. -->

# Where Can I Go for Help?

If you need help you can [open an issue](https://github.com/qmk/qmk\_firmware/issues) or [chat on Discord](https://discord.gg/Uq7gcHh).

# How Do I Make a Contribution?

Never made an open source contribution before? Wondering how contributions work in QMK? Here's a quick rundown!

0. Sign up for a [GitHub](https://github.com) account.

1. Put together a keymap to contribute, [find an issue](https://github.com/qmk/qmk\_firmware/issues) you are interested in addressing, or [a feature](https://github.com/qmk/qmk\_firmware/issues?q=is%3Aopen+is%3Aissue+label%3Afeature) you would like to add.

2. Fork the repository associated with the issue to your GitHub account. This means that you will have a copy of the repository under `your-GitHub-username/qmk\_firmware`.

3. Clone the repository to your local machine using `git clone https://github.com/github-username/repository-name.git`.

4. If you're working on a new feature consider opening an issue to talk with us about the work you're about to undertake.

5. Create a new branch for your fix using `git checkout -b branch-name-here`.

6. Make the appropriate changes for the issue you are trying to address or the feature that you want to add.

7. Use `git add insert-paths-of-changed-files-here` to add the file contents of the changed files to the "snapshot" git uses to manage the state of the project, also known as the index.

8. Use `git commit -m "Insert a short message of the changes made here"` to store the contents of the index with a descriptive message.

9. Push the changes to your repository on GitHub using `git push origin branch-name-here`.

10. Submit a pull request to [QMK Firmware](https://github.com/qmk/qmk\_firmware/pull/new/master).

11. Title the pull request with a short description of the changes made and the issue or bug number associated with your change. For example, you can title an issue like so "Added more log outputting to resolve #4352".

12. In the description of the pull request explain the changes that you made, any issues you think exist with the pull request you made, and any questions you have for the maintainer. It's OK if your pull request is not perfect (no pull request is), the reviewer will be able to help you fix any problems and improve it!

13. Wait for the pull request to be reviewed by a maintainer.

14. Make changes to the pull request if the reviewing maintainer recommends them.

15. Celebrate your success after your pull request is merged!

# Coding Conventions

Most of our style is pretty easy to pick up on. If you are familiar with either C or Python you should not have too much trouble with our local styles.

\* [Coding Conventions - C](coding\_conventions\_c.md)

\* [Coding Conventions - Python](coding\_conventions\_python.md)

# General Guidelines

We have a few different types of changes in QMK, each requiring a different level of rigor. We'd like you to keep the following guidelines in mind no matter what type of change you're making.

\* Separate PRs into logical units. For example, do not submit one PR covering two separate features, instead submit a separate PR for each feature.

\* Check for unnecessary whitespace with `git diff --check` before committing.

\* Make sure your code change actually compiles.

\* Keymaps: Make sure that `make keyboard:your\_new\_keymap` does not return any errors.

\* Keyboards: Make sure that `make keyboard:all` does not return any errors.

\* Core: Make sure that `make all` does not return any errors.

\* Make sure commit messages are understandable on their own. You should put a short description (no more than 70 characters) on the first line, the second line should be empty, and on the 3rd and later lines you should describe your commit in detail, if required. Example:

```

Adjust the fronzlebop for the kerpleplork

The kerpleplork was intermittently failing with error code 23. The root cause was the fronzlebop setting, which causes the kerpleplork to activate every N iterations.

Limited experimentation on the devices I have available shows that 7 is high enough to avoid confusing the kerpleplork, but I'd like to get some feedback from people with ARM devices to be sure.

```

!> \*\*IMPORTANT:\*\* If you would like to contribute a bugfix or improvement to user code, such as non-default keymaps, userspace and layouts, be sure to tag the original submitter of the code in your PR. Many users, regardless of skill level with Git and GitHub, may be confused or frustrated at their code being modified without their knowledge.

## Documentation

Documentation is one of the easiest ways to get started contributing to QMK. Finding places where the documentation is wrong or incomplete and fixing those is easy! We also very badly need someone to edit our documentation, so if you have editing skills but aren't sure where or how to jump in please [reach out for help](#where-can-i-go-for-help)!

You'll find all our documentation in the `qmk\_firmware/docs` directory, or if you'd rather use a web based workflow you can click the "Edit this page" link at the bottom of each page on https://docs.qmk.fm/.

When providing code examples in your documentation, try to observe naming conventions used elsewhere in the docs. For example, standardizing enums as `my\_layers` or `my\_keycodes` for consistency:

```c

enum my\_layers {

\_FIRST\_LAYER,

\_SECOND\_LAYER

};

enum my\_keycodes {

FIRST\_LAYER = SAFE\_RANGE,

SECOND\_LAYER

};

```

### Previewing the Documentation :id=previewing-the-documentation

Before opening a pull request, you can preview your changes if you have set up the development environment by running this command from the `qmk\_firmware/` folder:

./bin/qmk docs

or if you only have Python 3 installed:

python3 -m http.server 8936

and navigating to `http://localhost:8936/`.

## Keymaps

Most first-time QMK contributors start with their personal keymaps. We try to keep keymap standards pretty casual (keymaps, after all, reflect the personality of their creators) but we do ask that you follow these guidelines to make it easier for others to discover and learn from your keymap.

\* Write a `readme.md` using [the template](documentation\_templates.md).

\* All Keymap PR's are squashed, so if you care about how your commits are squashed you should do it yourself

\* Do not lump features in with keymap PR's. Submit the feature first and then a second PR for the keymap.

\* Do not include `Makefile`s in your keymap folder (they're no longer used)

\* Update copyrights in file headers (look for `%YOUR\_NAME%`)

## Keyboards

Keyboards are the raison d'être for QMK. Some keyboards are community maintained, while others are maintained by the people responsible for making a particular keyboard. The `readme.md` should tell you who maintains a particular keyboard. If you have questions relating to a particular keyboard you can [Open An Issue](https://github.com/qmk/qmk\_firmware/issues) and tag the maintainer in your question.

We also ask that you follow these guidelines:

\* Write a `readme.md` using [the template](documentation\_templates.md).

\* Keep the number of commits reasonable or we will squash your PR

\* Do not lump core features in with new keyboards. Submit the feature first and then submit a separate PR for the keyboard.

\* Name `.c`/`.h` file after the immediate parent folder, eg `/keyboards/<kb1>/<kb2>/<kb2>.[ch]`

\* Do not include `Makefile`s in your keyboard folder (they're no longer used)

\* Update copyrights in file headers (look for `%YOUR\_NAME%`)

## Quantum/TMK Core

Before you put a lot of work into building your new feature you should make sure you are implementing it in the best way. You can get a basic understanding of QMK by reading [Understanding QMK](understanding\_qmk.md), which will take you on a tour of the QMK program flow. From here you should talk to us to get a sense of the best way to implement your idea. There are two main ways to do this:

\* [Chat on Discord](https://discord.gg/Uq7gcHh)

\* [Open an Issue](https://github.com/qmk/qmk\_firmware/issues/new)

Feature and Bug Fix PR's affect all keyboards. We are also in the process of restructuring QMK. For this reason it is especially important for significant changes to be discussed before implementation has happened. If you open a PR without talking to us first please be prepared to do some significant rework if your choices do not mesh well with our planned direction.

Here are some things to keep in mind when working on your feature or bug fix.

\* \*\*Disabled by default\*\* - memory is a pretty limited on most chips QMK supports, and it's important that current keymaps aren't broken, so please allow your feature to be turned \*\*on\*\*, rather than being turned off. If you think it should be on by default, or reduces the size of the code, please talk with us about it.

\* \*\*Compile locally before submitting\*\* - hopefully this one is obvious, but things need to compile! Our Travis system will catch any issues, but it's generally faster for you to compile a few keyboards locally instead of waiting for the results to come back.

\* \*\*Consider revisions and different chip-bases\*\* - there are several keyboards that have revisions that allow for slightly different configurations, and even different chip-bases. Try to make a feature supported in ARM and AVR, or automatically disabled on platforms it doesn't work on.

\* \*\*Explain your feature\*\* - Document it in `docs/`, either as a new file or as part of an existing file. If you don't document it other people won't be able to benefit from your hard work.

We also ask that you follow these guidelines:

\* Keep the number of commits reasonable or we will squash your PR

\* Do not lump keyboards or keymaps in with core changes. Submit your core changes first.

\* Write [Unit Tests](unit\_testing.md) for your feature

\* Follow the style of the file you are editing. If the style is unclear or there are mixed styles you should conform to the [coding conventions](#coding-conventions) above.

## Refactoring

To maintain a clear vision of how things are laid out in QMK we try to plan out refactors in-depth and have a collaborator make the changes. If you have an idea for refactoring, or suggestions, [open an issue](https://github.com/qmk/qmk\_firmware/issues), we'd love to talk about how QMK can be improved.

# What Does the Code of Conduct Mean for Me?

Our [Code of Conduct](https://github.com/qmk/qmk\_firmware/blob/master/CODE\_OF\_CONDUCT.md) means that you are responsible for treating everyone on the project with respect and courtesy regardless of their identity. If you are the victim of any inappropriate behavior or comments as described in our Code of Conduct, we are here for you and will do the best to ensure that the abuser is reprimanded appropriately, per our code.