# Contributing to RIOT

Thank you for your interest in contributing to RIOT! There are many ways to

contribute, and we appreciate all of them. You can jump to the major sections

of this document using the following links:

\* [Getting Started][getting-started]

\* [Help wanted][help-wanted]

\* [General Tips][general-tips]

\* [Feature Requests][feature-requests]

\* [Bug Reports][bug-reports]

\* [Pull Requests][pull-requests]

\* [Writing Documentation][writing-documentation]

If you have questions, please send an email to users@riot-os.org or

devel@riot-os.org mailing list or chat on [#riot-os][riot-chat].

As a reminder, all contributors are expected to follow our

[Code of Conduct](CODE\_OF\_CONDUCT.md).

[riot-chat]: http://webchat.freenode.net?channels=riot-os

## Getting Started

[getting-started]: #getting-started

If you are just beginning to work with RIOT you might first want to read our

[documentation]. Especially the following sections might be of interest to you

- [Getting Started](https://doc.riot-os.org/getting-started.html)

- [Creating modules](https://doc.riot-os.org/creating-modules.html)

- [Creating an application](https://doc.riot-os.org/creating-an-application.html)

[documentation]: https://doc.riot-os.org

## General Tips

[general-tips]: #general-tips

From experience, the following recommendations help to get a software

contribution into RIOT master faster:

- \*\*Ask around for help!\*\* Either offline or through one of our communication

channels (see above). The earlier you check your feature design with other

people, the less likely it is that it is denied during the review process.

- \*\*Verify your concept early!\*\* If you work on your own until the code

\*looks\* good enough to show publicly, you might miss some design flaws others

might have spotted earlier.

- \*\*Keep it simple!\*\* Try to use what is already there and don't change existing

APIs if not absolutely necessary.

- \*\*Keep it small!\*\* A PR with >1000 lines of changes will very likely make

even the most active reviewer put your review on their long to-do list.

- \*\*Keep it modular!\*\* Make extensions to a feature or new features for a

platform optionally to use.

- \*\*Provide tests!\*\* They should be comprehensible and easy to be executed.

Alternatively comprehensive testing procedures should be provided with your

pull request.

## Help Wanted

[help-wanted]: #help-wanted

In case you're not really sure where to start, we've created a list of suggestions.

### Documentation

If you've found yourself struggling to understand a particular aspect, chances

are you're not the first and won't be the last. Writing down what you've learned

is a great way to recap your new knowledge and share it with others. You can

also start to learn about RIOT by combing through existing documentation and

fixing errors and typos. Any help with improving the documentation is greatly

appreciated and makes a big difference to the RIOT project.

After you've finished writing, please publish your documentation in one of the

following ways, depending on its type.

#### General knowledge, HOWTOs

Articles that focus on design aspects or how to use a particular module should

be contributed to the [RIOT wiki](https://github.com/RIOT-OS/RIOT/wiki). After

you've added your entry, please share it on the riot-dev mailing list so

everyone is aware of its existence (and thank you).

If you'd like to document a solution to minor annoyances or common pitfalls,

please do not hesitate to extend the [Troubleshooting wiki

page](https://github.com/RIOT-OS/RIOT/wiki/Troubleshooting). Again, please share

your additions with the riot-dev mailing list.

#### Code comments, HOWTOs for particular projects

Documentation that relates directly to the code at hand like the HOWTO files

that can be found in some of the directories in ``RIOT/examples/`` or comments

in the code itself should be submitted through a [pull request][pull-requests].

If you're not sure about the correct way to submit your writing, please ask on

the mailing list or open an issue saying which documentation is missing. The

other RIOTers will help you find the right format.

### Issues

If RIOT behaves oddly, please do not hesitate to [open an issue][open-an-issue].

Other RIOT developers will be happy to help figure out what the problem is and

fix possible bugs. Please notice that we use a bunch of tags to label the

issues. If you have permission to use them, do it. Their meanings are explained

[here][labels].

### Contribute code

If you think your work should be integrated in the main RIOT repository, take

the following steps: (short version, the more detailed version can be found

[below][pull-requests])

0. Fork the RIOT git repository (if you haven't done this already)

1. Create a branch

2. Make commits

3. Make sure your code is in compliance with RIOTs

[coding conventions][coding-conventions]

1. Push this branch to your fork on GitHub

1. Do a [pull request][open-a-pull-request] (use the [labels] if you have

permission to use them)

1. RIOT maintainers will provide feedback

1. Address this feedback

1. Your code is merged in RIOT master branch

If you do not receive feedback after a reasonable time, feel free to address

maintainers directly. This is especially true if you addressed previous feedback

and got no response.

[open-an-issue]: https://github.com/RIOT-OS/RIOT/issues?q=state:open+type:issue+label:"Type:+bug"

[labels]: https://github.com/RIOT-OS/RIOT/wiki/RIOT%27s-labeling-system

[open-a-pull-request]: https://help.github.com/articles/using-pull-requests

## Feature Requests

[feature-requests]: #feature-requests

Before opening a new feature request, check the

[existing feature requests][existing-feature-request] if there's one already

open on the same topic.

To request new features or enhancements, just open a new

[feature request issue][new-feature-request]. Describe your use case, why you

need this feature and why this feature is important for RIOT.

[existing-feature-request]: https://github.com/RIOT-OS/RIOT/issues?q=state:open+type:issue+label:"Type:+new+feature"

[new-feature-request]: https://github.com/RIOT-OS/RIOT/issues/new?template=feature\_request.md&title=Feature+Request:

## Bug Reports

[bug-reports]: #bug-reports

While bugs are unfortunate, they're a reality in software. We can't fix what we

don't know about, so please report liberally. If you're not sure if something

is a bug or not, feel free to file a bug report anyway.

\*\*If you believe reporting your bug publicly represents a security risk to

RIOT users, please send an email describing the bug to security@riot-os.org\*\*.

We would appreciate waiting for a 6 months grace period before reporting it on

public channels, to allow us adequate time to release the fix.

Before reporting a bug, have a look at [open bugs][existing-bugs-link]. Maybe

someone has already reported your error.

Once you have verified that the bug you have found hasn't been reported, opening an issue is as easy as

clicking on [this link][bug-report-link] and filling out the fields.

[existing-bugs-link]: https://github.com/RIOT-OS/RIOT/issues?q=state:open+type:issue+label:"Type:+bug"

[bug-report-link]: https://github.com/RIOT-OS/RIOT/issues/new?template=bug\_report.md&title=Bug:

Each bug report issue uses a template with 5 sections that are there to help

other contributors understand your issue and eventually reproduce it:

#### Description

#### Steps to reproduce the issue

#### Expected results

#### Actual results

#### Versions

To fill the `Versions` section, you can use the script provided in the RIOT git

repository:

```

make print-versions

```

In summary, try to include as much information as possible to help maintainers

or other developers fix the bug quickly.

## Pull Requests

[pull-requests]: #pull-requests

GitHub's Pull Request (PR) feature is the primary mechanism used to make

contributions to the RIOT codebase. GitHub itself has some great documentation

on [using the Pull Request feature][about-pull-requests].

We use the [fork and pull model][development-models], where contributors push

changes to their personal fork and create pull requests to bring those changes

into the source repository.

[about-pull-requests]: https://help.github.com/articles/about-pull-requests/

[development-models]: https://help.github.com/articles/creating-a-pull-request-from-a-fork

### General rules

\* Before opening a new Pull Request, have a look at

[existing ones][existing-pull-requests]. Maybe someone has already opened one

about the same thing. If it's the case, you might be able to help with the

contribution. Just comment on the PR and ask. Include closed PR's in your

search, as previous work might have been closed for lack of interest.

Old and stalled [PRs are sometimes archived][archived-pull-requests] with the

"State: archived" label, maybe one of them is also about the same topic.

\* Each Pull Request form uses a template with 3 sections that are there to help

maintainers understand your contribution and help them in testing it:

#### Contribution description

#### Testing procedure

#### Issues/PRs references

Please fill each section with as much information as possible.

\* The Pull Request title should reflect what it is about and be in the form:

`area of change: description of changes`

\* Remember that smaller PRs tend to be merged faster, so keep your changes as

concise as possible. They should be confined to a single explainable

change, and be runnable on their own. So don't hesitate to split your PRs

into smaller ones when possible.

\* In the Pull Request form, we recommend that you leave the

"Allow edits from maintainers" check box ticked. This will allow maintainer

finalizing your PR by pushing in your branch. In general, this speeds up the

PR merge in the main repository. Note that this is not an obligation.

\* Check if your code follows the [coding conventions][coding-conventions]. If

it doesn't, you can [uncrustify][uncrustify] a file:

$ uncrustify -c $RIOTBASE/uncrustify-riot.cfg <your file>

\* RIOT provides static test tools to verify the quality of changes (cppcheck,

trailing whitespaces, documentation, etc). These tools are wrapped in a

single `make` target: `static-test`.

\*Watch out:\* the command below will rebase your branch on your master branch,

so make sure they can be rebased (e.g. there's no potential conflict).

$ make static-test

Use it before opening a PR to perform last time checks.

\* Each commit should target changes of specific parts/modules of RIOT. The

commits use the following pattern:

`area of code: description of changes`.

You can use multi-line commit messages if you want to detail more the

changes.

\* Try to answer reviews as quickly as possible to speed up the review process

and avoid stalled PRs.

\* Maintainers try their best to review every PR as fast as possible, but they

are also only human and it can happen that they miss a few PRs or might be

preoccupied with other PRs. If it happens that your PR receives no review for

a long time, don't hesitate to gently solicit a review by commenting or

by explicitly mentioning a maintainer that you know is knowledgeable in the

area of the PR. You can also advertise the PR on devel@riot-os.org mailing

list and ask for a review there.

You can find more information about RIOT development procedure on this

[wiki page][development-procedures].

[existing-pull-requests]: https://github.com/RIOT-OS/RIOT/pulls

[archived-pull-requests]: https://github.com/RIOT-OS/RIOT/pulls?q=is:pr+label:"State:+archived"

[coding-conventions]: CODING\_CONVENTIONS.md

[uncrustify]: http://uncrustify.sourceforge.net

[development-procedures]: https://github.com/RIOT-OS/RIOT/wiki/Development-procedures

### Git usage

Using git is a bit difficult for newcomers. If you are completely new to git,

we recommend that you [start by learning it][try-github-io] a bit. You can also

read the official [getting started documentation][git-scm-getting-started].

In this section, we give the bare minimum for a better experience with our

development workflow on GitHub.

[try-github-io]: https://try.github.io/

[git-scm-getting-started]: https://git-scm.com/book/en/v2/Getting-Started-Git-Basics

#### Setup your local RIOT repository

Before you start modifying code, you need to fork the RIOT upstream repository

from the [RIOT main GitHub page][riot-github].

If it's your first time with git, configure your name and emails:

$ git config --global user.name = "<your name here>"

$ git config --global user.email = "<your email address here>"

Then clone locally your fork of RIOT (replace `account name` with your actual

login on GitHub):

$ git clone git@github.com:<account name>/RIOT.git

You can keep any branch of your local repository up-to-date with the upstream

master branch with the following commands:

$ git checkout <branch name>

$ git pull --rebase https://github.com/RIOT-OS/RIOT.git

Use it before opening a PR. This will at least ensure the PR is mergeable but

also that it is up-to-date with the upstream repository.

[riot-github]: https://github.com/RIOT-OS/RIOT

#### Work on branches

Avoid opening PR from the `master` branch of your fork to the master branch of

the RIOT upstream repository: update your master branch and start a new branch

from it.

$ git checkout master

$ git pull --rebase https://github.com/RIOT-OS/RIOT.git

$ git checkout -b <new branch>

# Do your changes, commit, update with latest upstream master

$ git push

#### Add fixup commits during review

To keep the history of changes easier to track for reviewers, it is recommended

to push your review request updates in fixup commits.

Let's say your PR contains 3 commits with comments: `prefix1: change 1`,

`prefix2: change 2` and `prefix3: change 3`.

Instead of committing changes in `prefix2` in a 4th commit `prefix2: change 4`,

you can use the `--fixup` option:

$ git add /path/of/prefix2

$ git commit --fixup <prefix2 commit hash>

#### Squash commits after review

Squashing a commit is done using the rebase subcommand of git in interactive

mode:

$ git rebase master -i

You can find information on rebasing in

[GitHub rebase documentation][about-git-rebase].

[about-git-rebase]: https://help.github.com/articles/about-git-rebase/

If you used [fixup commits](#add-fixup-commits-during-review) during the review

phase, squashing commits can be performed in a single command:

$ git rebase -i --autosquash

\*\*Watch out: Don't squash your commit until a maintainer asks you to do it.\*\*

Otherwise the history of review changes is lost and for large PRs, it

makes it difficult for the reviewer to follow them. It might also happen that

you introduce regression and won't be able to recover them from previous

commits.

If you encounter a merge conflict you could either resolve it by hand with an

editor and use

$ git add -p

To add your changes or use a merge tool like [meld](https://meldmerge.org/) to

resolve your merge conflict.

$ git mergetool

After the merge conflict is resolved you can continue to rebase by using

$ git rebase --continue

Once squashing is done, you will have to force push your branch to update the

PR:

$ git push --force-with-lease

## Writing Documentation

[writing-documentation]: #writing-documentation

Documentation improvements are always welcome and a good starting point for

new contributors. This kind of contribution is merged quite quickly in general.

RIOT documentation is built with [doxygen][doxygen]. Doxygen is configured to

parse header (.h) and `doc.txt` files in the RIOT source code to generate

the modules, cpus, boards and packages documentation.

General documentation pages are written in Markdown and located in

`doc/doxygen/src`.

To generate the documentation, simply run:

$ make doc

from the base directory of the RIOT source code.

The generated documentation is located in `doc/doxygen/html`

[doxygen]: http://www.doxygen.nl/