# Contribution guidelines

## Contributing to the stlink project

We love your input! We want to make contributing to this project as easy and transparent as possible, whether it's:

- Reporting a bug

- Discussing the current state of the code

- Submitting a fix

- Proposing new features

- Assistance with maintaining

We use GitHub to host code, to track issues and feature requests, as well as accept pull requests.

Report a bug by [opening a new issue]() with one of the available templates. It's that easy!

## Coding conventions

To read code written by other contributors can turn out to be quite demanding - a variable which seems to self-explaining, may appear cryptic to other readers. If you plan to contribute, please take this into account and feel encouraged to help others understand your code. In order to help you along, we have composed some contribution guidelines for this project. As this project already has a history you may find parts in the codebase that do not seem to comply with these guidelines, but we are trying to improve continuosly. However we can do even better, if every contributor considers the following points:

\* Naming of all source code elements as well as comments should exclusively be written in English.

\* All functions and global variables should be fully explained. This includes a short description on \_what\_ the respective function does (but not necessarily \_how\_ this is achieved), an explantion of transfer parameters and/or return values (if applicable).

\* Use [fixed width integer types](http://en.cppreference.com/w/c/types/integer) wherever possible and size-appropiate datatypes.

\* Only make use of the datatype `char` for specific characters, otherwise use `int8\_t` or `uint8\_t` respectively.

### Coding Style

\* Use 4 spaces for indentation rather than tabs (the latter results in inconsistent appearance on different platforms)

\* Use `/\* your comment \*/` formatting for multi-line comments or section titles and `// your comment` for inline comments.

\* Please try to avoid special characters where possible, as they are interpreted differently on particular platforms and systems. Otherwise these may result in mojibake within the sourcecode or cause translation errors when compiling.

\* Use state-of-the-art UTF-8 encoding whereever possible.

## Github Flow

We Use [Github Flow](https://guides.github.com/introduction/flow/index.html) which implies that all code changes happen through Pull Requests (PRs).

They are the best way to propose changes to the codebase and we actively welcome your own ones:

1. PRs should focus on \_one\_ single topic.

2. Fork the repo and create your branch from `develop`.

3. Begin to implement your changes on a local or personal branch.

4. Take a look at existing PR and check if these target the same part of the codebase.

Should this be the case, you are encouraged to get in touch with the respective author and discuss on how to proceed.

5. Keep your personal feature-branch up to date with the current development branch, by merging in recent changes regularly.

6. Don't open a PR unless your contribution has evolved to a somehow completed set of changes.

7. If you've changed major features, update the documentation.

8. Ensure your PR passes our travis CI tests.

9. Issue that pull request!

## License

When you submit code changes, your submissions are understood to be under the same [BSD-3 License](LICENSE.md) that covers this project.<br />Feel free to contact the project maintainers should there be any related questions.