## Contributing guide

This document serves as a checklist before contributing to this repository.

It includes links to read up on if topics are unclear to you.

This guide mainly focuses on the proper use of Git.

### 1. Before opening an issue

To report a bug/request please file an issue in the right repository

(example for [Arduino\_Core\_STM32](https://github.com/stm32duino/Arduino\_Core\_STM32/issues/new/choose)).

But check the following boxes before posting an issue:

- [ ] `Make sure you are using the latest STM32 core and libraries versions.` See [lastest core here](https://github.com/stm32duino/Arduino\_Core\_STM32/releases/latest).

- [ ] `Your issue is NOT a question/feedback/suggestions.` This should be discussed on the [stm32duino forum](http://stm32duino.com):

\* questions on the [STM32 Core](http://stm32duino.com/viewforum.php?f=35).

\* bugs/enhancements on the [STM core: Bugs and enhancements](http://stm32duino.com/viewforum.php?f=38).

- [ ] `Make sure your issue is not already reported/fixed on GitHub or discussed on the forum` See [stm32duino forum](http://stm32duino.com).

- [ ] `Make sure you are working on the right repository.` See the table below.

| Repositories | Projects |

|---|---|

|[STM32 Core](https://github.com/stm32duino/Arduino\_Core\_STM32) | Arduino core support for STM32 based boards |

|[STM32 Tools](https://github.com/stm32duino/Arduino\_Tools) | Contains several tools (upload, scripts,...) |

|[STM32 Examples](https://github.com/stm32duino/STM32Examples) | Arduino library to provide examples related to STM32 |

|[Board Manager Files](https://github.com/stm32duino/BoardManagerFiles) | Storage for Arduino Boards Manager JSON file |

|[CMSIS module](https://github.com/stm32duino/ArduinoModule-CMSIS) | Mainly storage |

|[GNU Arm Embedded Toolchain binaries](https://github.com/stm32duino/arm-none-eabi-gcc) | Mainly storage |

|[STM32 Libraries for Arduino IDE](https://github.com/stm32duino) | Search on the STM32duino GitHub repository |

### 2. Posting the issue

When you have checked the previous boxes. Please consider the following points before posting the issue.

- [ ] `Describe the issue based on the behaviour you were expecting`

- [ ] `Post complete error messages using markdown code fencing:` [Markdown Code Fencing Example](https://guides.github.com/features/mastering-markdown/#examples)

- [ ] `Provide a full set of steps necessary to reproduce the issue`

- [ ] `Demonstration code should be complete, correct and the minimum amount necessary to reproduce the issue`

### 3. Pull Requests

About [Pull Requests](https://help.github.com/articles/about-pull-requests/)

### 4. Commit messages

An easy to read pull request will speed up the merging process. Your commit messages need to be logically separate.

And containing enough information on their own. When this is done consistently your pull request will have an easy

to read log of changes.

Your commits need to be [atomic](https://www.freshconsulting.com/atomic-commits/) which allows the repository to remain

flexible after merging.

If you did not read the following 7 points before or just want to fresh up. Please read up on them on this [website](https://chris.beams.io/posts/git-commit) by Chris Beams.

1. Separate subject from body with a blank line

2. Limit the subject line (first line) to 50 characters

3. Capitalize the subject line

4. Do not end the subject line with a period `(.)`

5. Use the imperative mood in the subject line.

This should be in the written as giving an instruction for example "Fixed save-as bug" (it shows what the PR achieves when merging it)

6. Wrap body at 72 characters

7. Use the body to explain what, why and how

If your pull request fixes, closes or resolves an issue please reference it in the body with the following [syntax](https://help.github.com/articles/closing-issues-via-commit-messages/). Also see the last lines of the following example.

A general example with these 7 guidelines in mind is shown below (from the same website of [Chris Beams](https://chris.beams.io/posts/git-commit)):

```

Summarize changes in around 50 characters or less

More detailed explanatory text, if necessary. Wrap it to about 72

characters or so. In some contexts, the first line is treated as the

subject of the commit and the rest of the text as the body. The

blank line separating the summary from the body is critical (unless

you omit the body entirely); various tools like `log`, `shortlog`

and `rebase` can get confused if you run the two together.

Explain the problem that this commit is solving. Focus on why you

are making this change as opposed to how (the code explains that).

Are there side effects or other unintuitive consequences of this

change? Here's the place to explain them.

Further paragraphs come after blank lines.

- Bullet points are okay, too

- Typically a hyphen or asterisk is used for the bullet, preceded

by a single space, with blank lines in between, but conventions

vary here

If you use an issue tracker, put references to them at the bottom,

like this:

Resolves: #123

See also: #456, #789

```

### 5. Rebasing pull requests

When different people are working on the Arduino project simultaneously, pull requests can go stale quickly. A "stale" pull request is one that is no longer up to date with the latest merges in the project. It needs to be updated before it can be merged.

Most often pull requests become stale when merge conflicts occur. This happens when two pull requests both modify similar lines in the same file and one gets merged, the unmerged request will now have a merge conflict and needs updating.

When your pull request is stale, you will need to rebase your branch on the current master branch before you can merge it without conflicts.

More information about rebasing can be found at the repository of [edX](https://github.com/edx/edx-platform/wiki/How-to-Rebase-a-Pull-Request).

### 6. Merged!

When your pull request is merged please update the documentation if the changes require it

[Wiki](https://github.com/stm32duino/wiki/wiki)