Embyr Technologies Versioning Standard

Matthew Cooper Healy

8 April 2018

Testing Suffix

Note that `-alpha` or `-beta` may be optionally included at the end of a version number to indicate initial testing of a product to be more explicit to those not understanding the semantic meanings of `0.5.0` or `0.9.0`. In these cases, the correct version numbers should still be included, and the inclusion of the testing suffix is intended as an easier way of identifying the testing face for those not familier with the product. Note that this should almost always be done if versions are being distributed for external testing, in order to make the state of the product perfectly clear. In all other cases, this can be done at the discretion of the designer.

Software

Embyr Technologies uses "Semantic Versioning," the form of which is `major.minor.patch`.

Major

This number which should be no more than **two** digits, indicates that a backwards-incompatible change was made.

For example, a version update from `1.5.7` to `2.0.0` indicates that a change was made in the software, such that things that interacted with the previous API may no longer work.

Prototypes

A special exception to this rule is in the case of *prototypes*, which are defined as early revisions of a project that are not yet ready for distribution, due to safety concerns, unstable code, not yet implementing all features required for testing, not yet having been tested, etc.

In these cases, where the revision is still a prototype, the Major part of the version number will be 0, without exception.

Minor

This number, which should be no more than **two** digits, indicates that new functionality has been added.

For example, a version update from `1.4.7` to `1.5.0` indicates that the new version (`1.5.0`) includes some functionality and features not included in the previous version.

Note: This number resets to `0` upon iteration of the "major" number.

e.g. `1.5.4` to `2.0.0`

Prototypes

In the case where the current revision is a prototype, there are two minor revisions that carry special semantic meaning: - `0.5.x` indicates that the product is currently undergoing ***alpha*** testing and revisions required therein - `0.9.x` indicates that the product is currently undergoing ***beta*** testing and revisions required therein

Patch

This number, which should be no more than **three** digits, indicates that bugs or issues in previous versions have been addressed.

For example, a version update from `2.1.0` to `2.1.1` indicates that bug-fixes and corrections of feature implementation have occured.

Note: This number resets to `0` upon iteration of the "minor" number.

e.g. `1.5.4` to `1.6.0`

Note that this change occurs even when the Minor number is reset to 0 from any other number.

Follow Letter

In some cases, when a project splits into various branches for any reason, a letter is added to the end of the version number to distinguish between branches.

Each branch begins iterating and update its version number independently from the others, but the letter remains, so that it is easier to identify the common parent of several branches.

For example, the project entitled "Nomad" has been split into three branches, (explorer, ranger, and commander (working titles)) to fit varying design goals. Assuming the common parent of version `0.5.0`, the new branches begin at `0.5.0e`, `0.5.0r`, and `0.5.0c` respectively. Further iterations in each design are independent from the others, and only iterate the version number of their branch. For example, if the explorer class added new features, it would iterate to `0.6.0e` while ranger and commander classes would remain at `0.5.0r` and `0.5.0c`, respectively.

Hardware

Major

This number which should be no more than **two** digits, indicates that a backwards-incompatible change was made. This is usually done upon complete redesign of hardware, or upon adding new features which massively change the behavior of the product in any way.

For example, a version update from `1.5.7` to `2.0.0` indicates that a change was made in the hardware that changes the behavior, such as changing voltage requirements, or changing the number of outputs included.

Prototypes

A special exception to this rule is in the case of *prototypes*, which are defined as early revisions of a project that are not yet ready for distribution, due to safety concerns, unstable code, not yet implementing all features required for testing, not yet having been tested, etc.

In these cases, where the revision is still a prototype, the Major part of the version number will be 0, without exception.

Minor

This number, which should be no more than **two** digits, indicates that new functionality has been added which does not change the overall behavior of the hardware. The distinction between a minor and major update is made jointly

by the head of the project(Dir. of R&D) or product(Product Manager) and the Dir. of Prod. Dev., although it can be petitioned.

For example, a version update from `1.4.7` to `1.5.0` indicates that the new version (`1.5.0`) includes some functionality and features not included in the previous version but that do not change overall behavior, such as increasing clock speed of an onboard processor, or adding extra power filtering not *expressly* needed for the product.

Note: This number resets to `0` upon iteration of the "major" number.

e.g. `1.5.4` to `2.0.0`

Prototypes

In the case where the current revision is a prototype, there are two minor revisions that carry special semantic meaning: - `0.5.x` indicates that the product is currently undergoing ***alpha*** testing and revisions required therein - `0.9.x` indicates that the product is currently undergoing ***beta*** testing and revisions required therein

Patch

This number, which should be no more than **three** digits, indicates that bugs or issues in previous versions have been addressed. This is often done upon the remedying of manufactory errors, such as PCB misprints or incorrect traces, where the product produced does not match the expected behavior for the minor revision. If any functionality or behavior is changed, the minor or major number should be updated instead.

Note: updates of the patch number in hardware are much less common than in software, as changes more often affect bahavior.

For example, a version update from `2.1.0` to `2.1.1` indicates that correcting issues with design and corrections of manufactoring errors which prevent the product from functioning have occured.

Note: This number resets to `0` upon iteration of the "minor" number.

e.g. `1.5.4` to `1.6.0`

Note that this change occurs even when the Minor number is reset to 0 from any other number.

Follow Letter

In some cases, when a project splits into various branches for any reason, a letter is added to the end of the version number to distinguish between branches.

Each branch begins iterating and update its version number independently from the others, but the letter remains, so that it is easier to identify the common parent of several branches.

For example, the project entitled "Nomad" has been split into three branches, (explorer, ranger, and commander (working titles)) to fit varying design goals. Assuming the common parent of version `0.5.0`, the new branches begin at `0.5.0e`, `0.5.0r`, and `0.5.0c` respectively. Further iterations in each design are independent from the others, and only iterate the version number of their branch. For example, if the explorer class added new features, it would iterate to `0.6.0e` while ranger and commander classes would remain at `0.5.0r` and `0.5.0c`, respectively.