**Branching Strategy Version 2**

Updated: 6/25/2020

Contributors: Andrew Newman, Antonio Jimenez, Colin Fritz, Devon Stedronsky,

Eric Wagner, Harry Pengfei, Matthew Stevenson, Waseem Orphali

Approved By: DEVON and COLIN

Date: 6/25/2020

1. **Branching Strategy** 
   1. Will follow git-flow principles as outlined here <https://nvie.com/posts/a-successful-git-branching-model/>
   2. Merge all feature branches into remote Develop nightly. Remote is ground truth.
2. **Repo Content**
   1. Every module shall have a unique requirement and test case file with module name included in the titles.
   2. Source code and test benches will be in .sv files and held all in the Source Code folder.
   3. Life cycle data. Includes Requirements, Testcase, Checklists, Standards, System Specifications in their respective folders.
3. **Commit Message Standards**
   1. All commits should use smart commit syntax to reference what Jira ticket they are related to.
   2. Commit messages shall also reference the developer name and a summary of what the work in the commit is.
   3. Smart commit syntax is as follows (ex adds comment to Jira issue referenced)
      1. EX: GIT-264 #comment Resolved Conflicts
4. **The Master Branch**
   1. Main, stable production version of the project.
   2. Only admin(s) shall push changes to the master branch or by 3 reviewer approvals.
   3. Commits to the master branch must be tagged with a version number ex. V1.0
   4. Only a merge target for release and hotfix branches.
5. **Feature Branches**
   1. These branches are used for development of assigned modules and features.
   2. May not be completely functional or able to compile at any given time.
   3. Each team member is expected to retain local copies of changed files to ensure work is not lost due to merging conflicts.
   4. Only merge target for feature branches is the develop branch
   5. All merges into develop branch will be via merge commits and pull requests
   6. Feature branches must be deleted immediately after merge commit into the develop branch, create new branch to restart work off the develop branch
   7. Use “feature\_” as prefix of branch name(Ex. Feature\_timer)
6. **The Develop Branch**
   1. Will be the merge target for any feature branches, hotfixes, or finished release branches
   2. Multiple commits with less than 20 lines of edited source code shall be rebased to a single commit before merging with the develop branch.
      1. This will reduce clutter and make the change log more readable.
7. **Release Branches** 
   1. Created from Develop branch when it is believed a fully verified and functioning version of the project exists. Only bug fixes, documentation etc. But no added functionality.
   2. Finish by merge commit with the Master and Develop branches.
   3. Each Release branch is intended to have a new Release version number via a tag on each commit. EX: RV1.0