**Shockwave 4488 C# Coding Conventions**

Notes:

· Most of these will be in the provided Resharper settings file, which will help hold the conventions. When creating a new project copy all the files in RepositoryFiles to the root of the repository, and rename FRCxxxx.sln.dotsettings to match the solution.

# **Visual Studio/Resharper Settings**

· Make tabs equal to 4 spaces.

· Whenever creating a new project, make sure the .NET version is set to 4.5. 4.5.x will not work correctly.

· Use the default Visual Studio curly bracket structure. This means curly brackets on their own lines.

# **Solution, Project, File, and Folder Naming**

· Root solution should be named FRCxxxx, where xxxx is the current competition season.

· Main robot project should be named FRCxxxx.

· Main robot root namespace should be FRCxxxx.

· Main robot assembly name should be FRCxxxx.

· Main robot class should be called FRCxxxx.

· Note most of the ones above will be correct if you use the extension to create the project.

· Unit Test Assembly, project and root namespace should be named FRCxxxx.Tests.

· Simulator Assembly, project and root namespace should be named FRCxxxx.Simulator.

· Files should be {nameofclass,interface,orstruct}.cs.

· Nested folders are allowed as long as their name does not have any spaces in it. Any files contained in a nested folder will use the namespace {root}.{foldername}

# **Class, Interface, Struct and Enum Naming**

· Class, struct and enum names should be in PascalCase, with the name matching the file name. Only one class per file. Files can contain multiple structs or enums that pertain to the class.

· If multiple classes use the same struct or enum, split those into their own files. File can include multiple structs or enums, and the file name should be suffixed with Enum or Struct.

· Interfaces should be named in PascalCase, with an “I” prefix.

· Abstract classes should be suffixed with “Base”.

# **Method, Field, and Property Naming and Visibility**

· Methods should be in PascalCase, and can have any accessibility.

· Properties should be in PascalCase, and can have any accessibility. However, if the property is not required to be set outside of the current class, the setter should be private.

· No fields should be public or internal. Protected and private fields are ok. If they need to be accessed outside the class, make a property to access the field instead.

· Instance fields should be in camelCase, with an “m\_” prefix.

· Static fields should be in camelCase, with an “s\_” prefix.

· Constants should be in PascalCase, with no prefixes or suffixes.

· Parameter names should be in camelCase.

# **Class Ordering**

· Files should start with any required Enums, then any required Structs, and then the main class or interface.

· Members should be organized in the following way

o Fields

o Properties

o Constructors

o Methods

# **Singleton Pattern**

· If a class should only be instantiated once, but can be used from multiple classes, please use the Singleton pattern.

· The singleton pattern is a private variable of the class type with the name s\_instance. Then there is a public static property called Instance, with the following signature, with DriverStation being replaced by the class name.

o public static DriverStation Instance => s\_instance ?? (s\_instance = new DriverStation());

· After this, the constructor is made private. This will make sure only 1 instance can created.

· You can now access the class from any other class.

· In the RobotInit() method in the main robot class, all Instance properties should be grabbed in order to initialize all the systems.