## General Style Guide for High Level Software Development

Add to this document when rules of thumb / good practises are discovered.

General c++

* Write iso-cpp
  + <https://isocpp.org/faq>
* Use iterators as much as possible in loops
* Use auto as much as possible
* const correctness
  + use const as much as possible
* Give good names for functions and variable (where appropriate)
  + Good names mean much of the code can be self-documenting
* Add comment as much as you like
* Pass by reference wherever possible.
  + Exceptions to this, simple data types like int, double, bool,
  + when making Python Bindings, as python does not do pass-by-reference (but it seems to work anyway – so maybe we could drop this?)
* Explicitly write namespaces, no “*using namespace std*”

Hardware Controllers

* Use enums for names.
  + Define enums with this macro: DEFINE\_ENUM\_WITH\_STRING\_CONVERSIONS
* Use structs for holding data + *meta-data* for easy access.
  + E.g virtual hardware should be represented in a struct.
* Hold the structs in maps keyed by enums / names
* If you write a function that you think might be generic enough consider adding it to the base class so everyone benefits..
  + E,g, as the file readers have been developed functions for reading out an int, a double, a vector of doubles etc have been created.

1. General
   1. Names

When naming variables, functions, classes etc., clarity over convenience is highly recommended. Expand out acronyms and if the functions/class has a specific use, let it be known in the name. Also the style of how names are written is a follows:

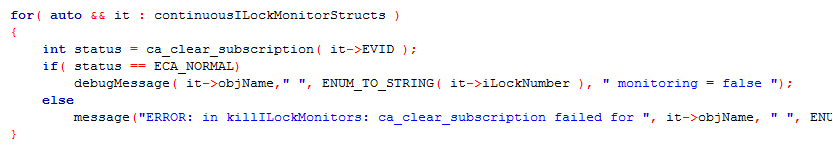
YourNamesShouldStartWithALowerCaseLetterThenCapitaliseAnyMoreWordsThatFollow

* 1. Vectors

Use vector/maps over arrays

* 1. For Loops

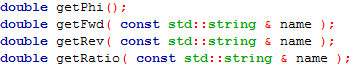
When using for loop always define the iterator using ‘auto’ and try to loop the vector.



* 1. Functions

The time when functions are called for when either a extremely long section of code needs to be written and used. Having a function allows for modularity and easier reading. Functions are also suggested whenever there is repeated code.

Pass by reference as much as possible, and also make it a ‘const’. For example…



* 1. Classes

For classes define them and their members and functions in a header file. The functions of a class can be written in .cpp file that has a link to the header file containing the class. Use classes to structure your code, for example, use a class to contain many functions of a similar method (e.g. fitting functions).

* 1. Comments

Comment regular. The more the better, but as minimum have a comment explain what each section of code does. There comments when classes/functions are defined are good rule of thumbs to have.

1. Specific To this code