**Standards for Header File Usage**

As discussed and documented in issue 43, the components of the new standard are:

- We will group related functions into what we'll call "bags", using a new name rather than reusing another term and risk confusion
- A bag will consist of 2 source files: a header (.h) file and a .cpp file
- the header bag file has a header section for each function
- the .cpp bag file has a code section for each function
- header files or sub-sections can include:
    - syntax declarations for function calls within the associated cpp file or bag (FOB) that are used by other FOBs
    - syntax definitions for classes & templates
    - inline function definitions
- header files or sub-sections can not include:
    - non-inline function definitions
    - any variable definitions
    - aggregate definitions
    - unnamed namespaces
    - using directives
    - included .cpp files
    - other included header files
    - OOP constructor or method calls
- Our code has only one compiled module, main.cpp (with a bunch of embedded include files) so we don't need to use extern. Libraries are a separate issue.
- We can and should put include guards in our header files, but they shouldn't be needed: main.cpp includes main.h, and main.h includes all other header files. Each header file gets processed exactly once. No file other than main.h should include any other header file.

Implications:

1. Where do non-const global variables go?
   - In the new include file global_variables.cpp. ( const global variables go here too.)

2. Do we need to be careful to arrange functions so they appear before all code that calls them?
   - No, there will be a syntax definition in their header file, and that header file will be seen by the compiler before any code that calls the function

3. What stuff appears in what order in main.cpp?

   - Include <Arduino.h> // following sections depend on symbols defined here
   - include <main.h> // do all .h files before any code files
   - include <our-libraries> // libraries that we've modified, like aaChio.h>
   - include <external-libraries> // such as Wire.h, Adafruit....
   - global variable declarations/definitions // externalized to global_variables.cpp
   - OOP constructor / instantiation calls // after libraries, but before function code
   - includes for our function code files and bags // externalized .cpp files that we wrote
   - functions we wrote that aren't externalized
   - standard functions setup() and loop()

4. What appears in main.h?
   - Includes for other .h files
   - syntax definitions for functions in main.cpp that are referenced elsewhere

5. What appears in other .h files?
   - syntax declaration for functions in associated .cpp file
   - do we want some global variables here, or keep them all in one place in global_variables.cpp?????

6. What appears in global_variables.cpp?
   - Global variable declarations and definitions, including those that are const.
   - Global macro definitions (#define)