The example coordMath bag is composed of 2 files:

coordMath.h

```
// global > local coordinate conversion
bool globalToLocal( float x, float y, float z)
#define trace_GTL = print ("got to GTL");

// local > global coordinate conversion
bool localToGlobal( float x, float y, float z)
#define trace_LTG = print ("got to LTG");
```

coordMath.cpp

```
// global > local coordinate conversion
bool globalToLocal( float x, float y, float z)
{
   float originx = 9
   float originy = 2.75
   float originz = -4
   g_LX = originx + sin(x/y)* g_side
   Return true
}

// local > global coordinate conversion
bool localToGlobal( float x, float y, float z)
{
   float originx = 9
   float originy = 2.75
   float originz = -4
   g_GX = originx + cos(y/x)* g_side
   Return true
}
```

Main.h:

```
// includes for all other .h files
#include coordMath.h
#include MQTTBroker.h
#include flows.h
. . .
```

global_variables.cpp

```
// global variables, const and non-const
// declarations, definitions and defines

float g_tilt_x // desired tilt in global X dir
float g_tilt_y
float g_tilt_z

// debug printout shortcut
#define dp(string,value)=
    print(string);println(value);

const int g_armpit1 = 102
const int g_armpit2 = 413
```

Main.cpp:

```
// this is the only place main.h is #included
#include main.h

// get declarations and definitions and defines
//  for all global variables
#include global_variables.cpp

// get all libraries that we use
#include <Arduino.h> // Arduino Core
#include <aaChip.h>
. . .

// get all cpp files and bags that we use
#include MQTTBroker.cpp
#include flows.cpp
#include coordMath.cpp
. . .

// any large sections of code can be moved
// to an external cpp file,
// and embedded with a #include statement

#include big_setup.cpp
#include big_loop.cpp
```

A bag is a collection of related functions, with a header file and code file. Each has a subsection for every 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
-macro definitions via #define statement
-inline function definitions
-header files can not contain:
- non-inline function definitions
- non-const variable definitions
- aggregate definitions
- unnamed namespaces
- using directives

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 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.

All global variables go in a separate file global_variables.cpp, #included from main.cpp

Because all header files are encountered by the compiler before any executable code, functions can appear in any order, but need a declaration in their header file.
order of compiler processing:
initially processing main.cpp
does main.h (main.h does all other .h files)
All non-const global variable declarations and definitions done by global_variables.cpp included from main.cpp
All libraries, via included from main.cpp
All includes of executable code
Executable code in main.cpp, notable setup() and loop()