The Utilities File

```
enum mode_t {IDLE, PWM, ITEST, HOLD, TRACK};
```

Eventually your code will have two timer based ISRs. When those ISRs fire, the functions will need to know what to do. You need a state variable that is settable from main() and gettable in the files with the ISR functions.

The state variable can take the values IDLE, PWM, ITEST, HOLD and TRACK. You could make each of these a number, like 0, 1, 2, 3, 4, or you can make a C type structure called an enum, and refer to the states by names rather than integer values,. In the enum mode_t above, IDLE is the value 1, PWM is the value 2, etc. So the enum isn't necessary but makes your code more readable.

Declare the enum in utilities.h, and every file that #include "utilities.h" will understand what a variable of type mode_t is.

Dedclae the state variable itself in utilities.c as a global type volatile mode_t. It is not accessible in main.c or any other .c file outside of utilities.c, so make a get_mode() and set_mode() function to pass the value of the state between files. This feels a little silly because the functions are so simple, why not just define the variable in utilities.h? Then you are relying on other programmers using your code to remember that there is a global variable for them to use and not make another local variable with the same name, better practice is to keep global variables to a single file and use functions to pass them around.

So in utilities.c:

```
volatile enum mode_t mode;
```