Design a code module

This article demonstrates how to design a new code module.

Header File

A header file:

- Shall have `#pragma` once attribute (google it for the reason)
- Shall NEVER have variables defined

```
//- Note: Remove all lines from your code that start with //-
//- Put this line as the very first line in your header module
#pragma once
//- #include all header files that THIS header needs
//- Do not include headers here that are not needed
//- For example, we do not need gpio.h file here, but maybe you can move this to switch_led.c
#include "gpio.h"

//- DO NOT put any variables here, like so:
static int do_not_do_this;
int definitely_do_not_do_this;
//- All functions without paramters should be marked as (void)
void switch_led_logic__initialize(void);
void switch_led_logic__run_once(void);
```

`#pragma` once is a replacement of

```
#ifndef YOUR_FILE_NAME__
#define YOUR_FILE_NAME__

void your_api(void);
```

```
#endif
```

Intent of `#pragma once` and `#ifndef`

- When other code modules `#include` your header file, you only want functions to be declared once
- The name of `#ifndef` can be anything unique, but must not conflict with other files
- `#include` literally copies and pastes the contents of the file in the file wherever you have the `#include`

Source File

A source file:

- Shall have all variables defined as static; this will keep their visibility private to their file

```
//- Note: Remove all lines from your code that start with //-
//- Include the header file for which this code modules belongs to
#include "switch_led_logic.h"
//- Declare all variables as STATIC
static gpio_s my_led;
//- Define your public functions (part of this module's header file)
void switch_led_logic__initialize(void) {
  my_led = gpio__construct_as_output(GPIO__PORT_2, 0);
}
void switch_led_logic__run_once(void) {
  gpio__set(my_led);
}
```

Unit Test file

A unit-test file:

- Shall `#include` the headers that you want (those that should not be "mocked")
- Shall `#include` Mock headers to generate stubs (rather than the full implementation)

Useful stuff

Clang auto-formatter will format the source code for you. It will also sort the #includes, so it is recommended to put an empty line such that it sorts the #includes more elegantly. For example, you can separate the FreeRTOS includes, system includes, and other includes.

```
//- Note: Remove all lines from your code that start with //-
//- Include system includes first
#include <stdio.h>
//- FreeRTOS requires this header file inclusion before any of its soure code
//- This only applies to code included from FreeRTOS
#include "FreeRTOS.h"
#include "semphr.h"
#include "task.h"
//- Clang will sort these
#include "abc.h"#include "def.h"
```

Try the following

- Have two code modules, such as `main.c` and `periodic_callbacks.c` include a header file that does not have `#pragma once` and observe what happens when you compile

Revision #3
Created 5 years ago by Preet Kang
Updated 4 years ago by Preet Kang