ECE/CSE 474:
Intro to Embedded Systems

Programming Style Guide

In embedded systems, we stress the importance of style (includes commenting) in your code due to the embedded systems software's organizational complexity. We do this by evaluating a fair amount of your lab grade in terms of stylistic correctness. **Anticipate 50% of your Lab Report grade to be based upon how well you follow the standard C programming style.** Here is a brief guide generated from common mistakes seen in previous quarters:

1. Take ownership and summarize your code by putting a description comment on each file you write.
2. Make a modular project by separating your code into separate .c files and including them through header files in your main program file. This is essential for later labs when you start dealing with multiple protocols and peripherals. Do NOT make trivial files.
3. Put your function comments in the header file above the function prototype. Your function comment should comment on its use and any input parameters and return values. Only inline comments or extra helper function comments should be in the .c file, so your implementation is easy to read.
4. main() should be the topmost function in your main programming file. Also main() should always return an int (even with the infinite while loop)
5. You must create your own customized header file with registers and macros that you need for your system. Also, do not #include a .c file
6. Do not forget to put the header guard in your header file.

For more information: https://google.github.io/styleguide/cppguide.html

It is important that following the style guide does not take way from the creativity and originality of your code. The desired result is for your code to be easy to follow. This will be helpful for others (graders or a future you) to figure out what your code does, or is supposed to do, and how different parts of your code fit together. Following this general style guide should hopefully prepare for your later courses, research, and industry work.

Here is a skeleton of a simple embedded systems program. Do NOT copy and paste this template when starting with Lab 1. Rather, use it as a reference. Note that this example does not contain the modular aspect mentioned above.

```
// _______ HEADER FILE (let's call it header.h for now) _______

/*
 * Brian Moon (Take ownership of your code by naming all files)
 * Student ID
 * Date
 * COMMENT on Header file.
 */

#ifndef HEADER_H_ // Do not forget header guards
#define HEADER_H_

// REGISTER DEFINITIONS
// COMMENT HERE on what the set of registers do
// Example:

// To set up and change the data for GPIO: LEDS, SW, etc
#define RCGCGPIO (*((volatile uint32_t *) 0x...))
#define RCGCDIR...
#define RCGCDEN...
// ...

// To set up timers and change the timers (for later labs)
# define RCGCTIMER (*((volatile uint32_t *) 0x...))
// ...


// ADDITIONAL MACROS
// Example:
#define LED4 0x1
// ...


// FUNCTION PROTOTYPES (FUNCTIONS THAT ARE GENERIC TO THE SYSTEM)
// COMMENT HERE on what the function does, input/output param and return value (if any).
// Example:

// Sets up the LED, SW (Note: GPIO, DIR, DEN setup should go here)
void GPIO_Init();

// Sets up the Timer (for later labs)
void Timer_Init();

// Rotates through the LED (DATA logic should go here)
void LED_Blink();

// Turns on green LED
void LED_Green_On();

#endif // HEADER_H_
```

```c
// _____ MAIN PROGRAM (.c) _____
/*
 * Name
 * Student ID
 * Date
 * COMMENT on program description
 */

 // Include any libraries or header files
#include <stdint.h> // uint32_t, etc
#include <stdio.h> // printf (View -> TerminalIO), scanf, etc
#include "header.h"


// Define any constants (use sparingly)
const int I_AM_CONST = 1000;


// HELPER FUNCTION PROTOTYPES (FUNCTIONS THAT ARE SPECIFIC TO THE PROGRAM)
// COMMENT HERE on what the function does, input/output param and return value (if any).
// There should not be many of these in this course because most functions are systemspecific.
// Example:

// Adds a delay for the registers to be set up
void delay();


// MAIN FUNCTION
// DO NOT COMMENT HERE except for in-line comments
// Your main function must be the topmost function in the program AND
// must be like a table of contents.
int main() {
        GPIO_Init();
        TIMER_Init();
        while (1) {
        LED_Blink();
        // FSM to turn on green LED... (use separate function for FSM if lengthy)
        if / else or switch / case (something) {
        LED_Green_On();
        }
        //...
        }
        return 0;
}


// FUNCTION DEFINITIONS (FUNCTIONS IN HEADER FILE & HELPER FUNCTION)
// DO NOT COMMENT HERE except for in-line comments
// Functions in the header file is usually defined in a separate C file,
// but to simplify compilation for this course, you may define the functions here.
// Example:

void GPIO_Init() {
        RCGCGPIO = ...
        GPIODEN = ...
}
```

```
void Timer_Init() {
        // ...
}

void LED_Blink() {
        GPIODATA = ...
}

void delay() {
        for (int i = 0; ...)
}
//...
```

```c
// _________ HEADER FILE (let's call it header.h for now) _________

/*
 * Name (Take ownership of your code by naming all files)
 * Student ID
 * Date
 * COMMENT on Header file.
 */

#ifndef HEADER_H_ // Do not forget header guards
#define HEADER_H_

// REGISTER DEFINITIONS
// COMMENT HERE on what the set of registers do
// Example:

// To set up and change the data for GPIO: LEDS, SW, etc
#define RCGCGPIO (*((volatile uint32_t *) 0x...))
#define RCGCDIR...
#define RCGCDEN...
#define RCGCDATA ...
// ...

// To set up timers and change the timers (for later labs)
# define RCGCTIMER (*((volatile uint32_t *) 0x...))
// ...


// ADDITIONAL MACROS
// Example:
#define LED4 0x1
// ...


// FUNCTION PROTOTYPES (FUNCTIONS THAT ARE GENERIC TO THE SYSTEM)
// COMMENT HERE on what the function does, input/output param and return value (if any).
// Example:

// Sets up the LED, SW (Note: GPIO, DIR, DEN setup should go here)
void GPIO_Init();

// Sets up the Timer (for later labs)
void Timer_Init();

// Rotates through the LED (DATA logic should go here)
void LED_Blink();

// Turns on green LED
void LED_Green_On();

#endif // HEADER_H_
```

```c
// _____________________ MAIN PROGRAM (.c) _____________________
/*
 * Name
 * Student ID
 * Date
 * COMMENT on program description
 */

 // Include any libraries or header files
#include <stdint.h> // uint32_t, etc
#include <stdio.h> // printf (View -> TerminalIO), scanf, etc
#include “header.h”


// Define any constants (use sparingly)
const int I_AM_CONST = 1000;


// HELPER FUNCTION PROTOTYPES (FUNCTIONS THAT ARE SPECIFIC TO THE PROGRAM)
// COMMENT HERE on what the function does, input/output param and return value (if any).
// There should not be many of these in this course because most functions are
systemspecific.
// Example:

// Adds a delay for the registers to be set up
void delay();


// MAIN FUNCTION
// DO NOT COMMENT HERE except for in-line comments
// Your main function must be the topmost function in the program AND
// must be like a table of contents.
int main() {
	GPIO_Init();
	TIMER_Init();
	while (1) {
		LED_Blink();
		// FSM to turn on green LED... (use separate function for FSM if lengthy)
		if / else or switch / case (something) {
			LED_Green_On();
		}
		//...
	}
	return 0;
}


// FUNCTION DEFINITIONS (FUNCTIONS IN HEADER FILE & HELPER FUNCTION)
// DO NOT COMMENT HERE except for in-line comments
// Functions in the header file is usually defined in a separate C file,
// but to simplify compilation for this course, you may define the functions here.
// Example:

void GPIO_Init() {
	RCGCGPIO = ...
		GPIODEN = ...
}
```

```
void Timer_Init() {
	// ...
}

void LED_Blink() {
	GPIODATA = ...
}

void delay() {
	for (int i = 0; ...)
}
//...
```