Workshop 7 - Week 9 - Large Program Techniques

The aim of this workshop is to :

1. understand multi-file compilation and how it can help simplify large projects.
2. understand what an ADC does.

Using multiple source code files instead of one large source file is preferred because:

- Industry employs this technique as most of their project can be many thousands of lines of code.eg (chrome is >6.7 million lines of code), (word is > 30 million lines of code) etc. They also like it because it is more efficient and saves money.
- Large source files can be very hard to edit and navigate and maintain. If everything is in one huge file then it is too easy to accidentally modify/damage other working functions.
- If you have multiple source files then you can compile incrementally which means that only the files that have changed need to be recompiled. Any unchanged files will not be compiled. This will decrease your compilation time.
- You can create groups of functions so that you can easily use them in other projects. Code reusability!
- It is easier to document the functionality since each file is dedicated to a particular task or peripheral.
- Multiple source files help logically break up a large program and makes it easier your code to write and test.
- It is easier for multiple programmers to work on the same project simultaneously since they can work on different files.

Section 1 : Multi-File compilation

Question 1

- Load Template1
- Add a new C file to the project and call it **Led_D1.c**
- Edit the **Led_D1.c** file to create a function **void configure_D1(void)** that will set up the GPIO for the D1 led.
- Modify the main function to flash the D1 led. You can use the msDelay() function in the library.

Question 2

- Modify the **Led_D1.c** file to write 2 functions.

```
void D1_on(void)          // turn the D1 led on
void D1_off(void)         // turn the D1 led off
```

Please note that the Led_D1.c file we are creating here holds all the functions that configure and act upon the D1 led. It is usual to create a file for the different peripherals since this is a self-contained logical grouping. The source file will usually contain the peripheral and the functions that it needs. eg for a UART you would have the UART configuration functions and the read and write functions contained in the file.

Question 3

Modify the **Led_D1.c** file to write a toggle function - **void D1_toggle(void)**. The toggle function should access a global variable called enable that will enable the toggling functionality. If the enable variable is false then the toggle does not change the state of the led. If the enable variable is true then the toggle function toggles the led. Verify that the toggle function and the enable works.

Note: The toggle function should be called from the main. The global enable variable should also be declared in the main.

Question 4

Modify the question so that you create and write a header file called workshop7.h that takes into account the functions that are compiled in separate files. The new workshop7.h file should be included at the top of the main() function. See the ES_lib.h header file for inspiration.
Hint: add a new file to the Source group list on the left. When selecting the extension for header file select the .h option.

Question 5

Add a new C file to the project that will contain the following:

a function to configure the GPIO for SW1

a function to wait until SW1 is pressed

a function to read the current state of SW1 without blocking

Modify the program so that it will only toggle the led when the SW1 button is pressed.

Section 2: Analogue to Digital Conversion

Question 6

In this question you will run a program that reads the analog voltage on pin PE2 and displays the equivalent digital result on Tera Term by using the in-built Analog to Digital Converter peripheral. This program will take a single measurement when you press the SW1 button.

- Load Template2
- Set Teraterm for 19200,8,N,1
- Compile the program and run it only after the connection to the pins has been completed.
- Connect the 3.3V pin to the PE2 pin. What does the output on the term show you?
- Connect the gnd pin to the PE2 pin. What does the output show you?
- Create a voltage divider to reduce the voltage to the PE2 pin to 1/2 of 3.3V. What does the output show you?

Question 7

Modify the code from Question 6 so that it continually prints the value to terminal and does not rely upon SW1 to indicate when to take the measurement.

Use a potentiometer and connect the centre tap to PE2 and the outside pins to GND and 3.3V respectively. Adjust the potentiometer values. What does the output show you?