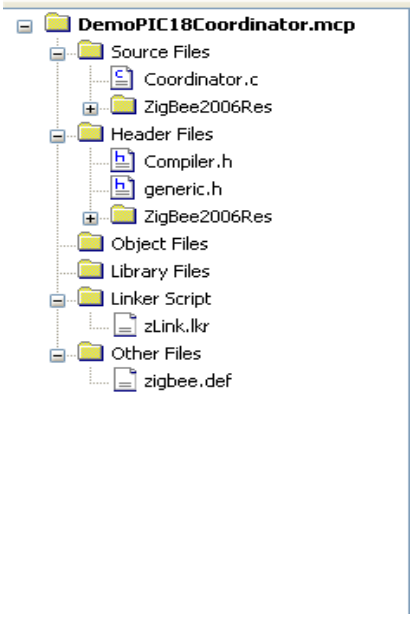
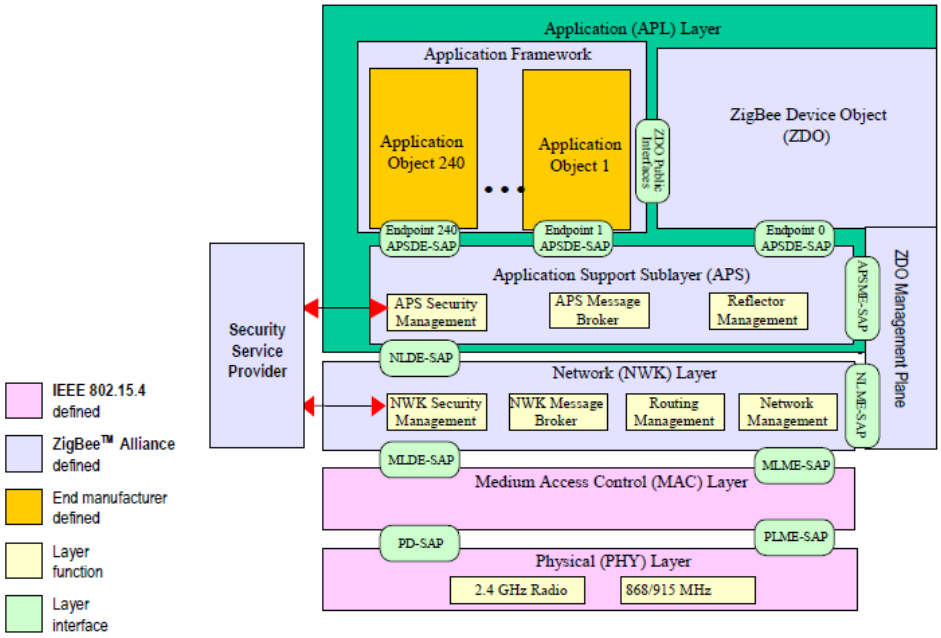
***Please use this document together with the lab 3 note**
Please complete section 2, section 1 is the introduction of ZigBee code structure.
----------Section 1 ----------

ZigBee stack source code

1. Open the MPLAB IDE with the coordinator project. Look at the index on the left. All the C files are in the "Source Files" folder, and all the corresponding header files are in the "Header Files" folder.

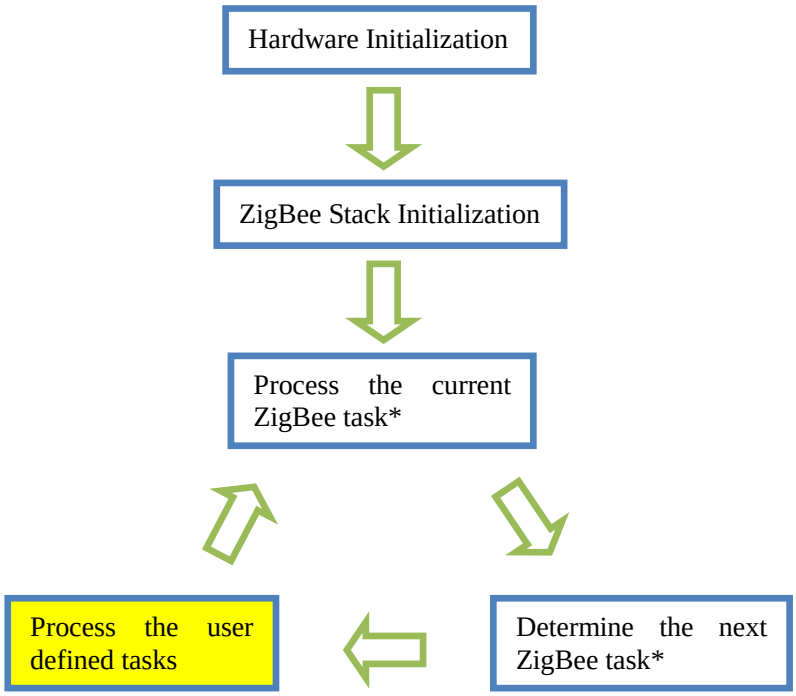


You can open the "ZigBee2006Res" folders to have a look. Remember the stack architecture showed in "tips for ZigBee":



Each major layer of the stack is written in a c file, as well as the RF module driver and serial port IO functions. The main function of this stack is in the "Coordinator.c" file (I called it "main.c" file in the lab 3 note).

2. This is a general structure shows how the main function in "Coordinator.c" is running:



*in ZigBee specification, the ZigBee tasks are defined using primitives.

The main function determines how the Micro-controller will run the program. Each block above contains many sub-functions, and all these functions, together, make up the whole source code stack.

Your task in section 1 is to read the "Coordinator.c" file, and try to see this structure in the code. Especially the yellow block, that's where you can add your own Non-ZigBee application code.

(It is alright that you may not understand all the source code now. That requires you to be very familiar with both the Micro-controller and the ZigBee specification. Just try to know how to use it.)

----------Section 1 stops here----------

//

----------Section 2 ----------

1. First of all, let's focus on two parts of the source code: one is the "Console.c" and "Console.h" files. All the USART IO functions are written here. The other one is in the "Coordinator.c" file, from line 1409 to line 1730. This is how the menu is written.

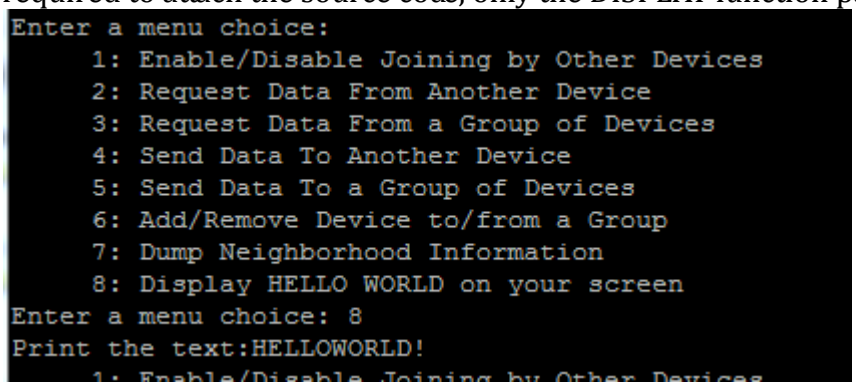
2. For the USART IO functions, you could focus on these ones:

```
ConsolePut();           // MCU output one BYTE
ConsolePutString();     // MCU output a String
ConsoleGet();           // MCU receive one BYTE
ConsoleGetString();     // MCU receive a String
```

(ConsolePut() and ConsoleGet() are the most widely used IO function of the source code. The other two are not necessary. If you are interested, you can try ConsolePutString() and ConsoleGetString())

Try to understand how these functions work by reading the code. It is very helpful for your later project (I will provide further tips when the project begins, but try it yourself first). Before that, you can go to the "Coordinator.c" file to see how the Menu is written using these functions. You will continue to work with this in the next lab. For this section, the tasks for you are:

1. Figure out how the menu is written, and see how these functions are used; (have a look at the function code "ProcessMenu" from line 1401 to line 1740, that's where the menu is triggered)
2. Try to modify a certain option, and see if you get the expected result (modify the menu part only);
3. Think of a way to completely disable the existing menu and block all the USART messages from the stack. Sometimes you may need a very "clean" USART environment for your own applications. (for example you are doing this on the coordinator, you may need to add more nodes into the network to see if the USART port is perfectly clear)
4. For example: display character string: "HELLOWORLD!" through USART port. (Remind: the content you want to display must be input in real time, other than predefined and pre input inside the source code. And you are required to attach the source code, only the DISPLAY function part)



----------**finished ☺**----------