1) From mycourses download this pdf and dxp_Lab11_a1.asm.
2) The **objective** of this lab is to understand how to use the UCI in UART mode.

3) For this lab make sure that jumpers 4 and 5 of J3 are installed in the direction of the dashed line, i.e. for HW-UART.

4) **Part 1: Build, run, understand, and demonstrate debug fmlxxxx_Lab11_a1.asm.**

5) You can implement this lab assignment in assembly only, in C only, or a combination of the two. It is highly recommended that you split the code in multiple files. During the preceding lecture, I will show some examples of mixing C and assembly, and how to distribute the code among multiple files.

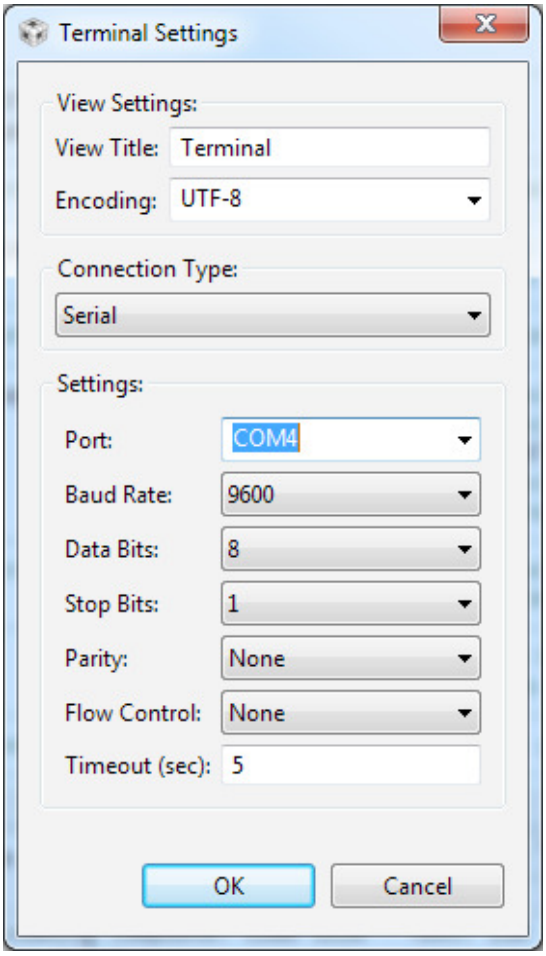
6) Run the program with breakpoints at lines: 30, 55, and 86. Check the relevant system state values after each sequence of instructions.

7) Now, comment line 39. Re-compile, enter debug and run the program. It will hang-up, because the port is in use and we cannot use a Terminal program to communicate with it. Click STOP in debug mode and exit it.

8) The program, however, is still in the micro-controllers flash memory.

9) Press the reset button on the board. The program now runs, waiting for a character from some Terminal client connected to the port.

10) Fortunately, eclipse has one build in. Go to View > Others > Terminal and click on Terminal. Next, click the settings icon and make the following selections:



11) Note that on your computer the COM port might be a different one. Check under devices → ports.

12) Click OK. The Terminal connects automatically.

13) Unfortunately, there's no way to locally echo the sent character. Type in an upper case letter. If the uC received the upper case letter correctly, it will respond immediately with its lower case version.

14) If you do not see any response from the uC (which you won't, since the BAUD rate was purposely set to the wrong value), we will have to make sure your BAUD rate is set correctly to 9600. Due to slight differences in silicon processes, and with different temperatures and humidities, the VCO's on the MSP430 might be producing slightly different clock speeds, so the BAUD rate pre-scaler for 9600 BAUD might be different from board to board. Regardless, it will need to be adjusted. Notice lines 44 and 45 in the code. Changing this pre-scaler will change the BAUD rate. Since we are only outputting characters when we correctly detect a character coming in from the Terminal, we will need to load another program which prints "hello world" continuously, and use an oscilloscope to measure the actual BAUD rate of the characters coming out.

15) Create a new assembly project and load the provided cab_Lab11_hello_world. This program will continuously print “hello , world!” to the UART. You will have to take a look at the TX data line coming out of the uC with an oscilloscope to make sure the BAUD rate is set to 9600. If it is not, you will have to change the pre-scaler values, until you get close to 9600. Record the value, and use it in your original project.

16) Another option instead of the oscilloscope is to use a logic analyzer. With a logic analyzer, you can see many different lines at the same time, for example, you can see the status of an 8-bit bus. Typically they can only see digital voltages (high or low), but they are able to decode packets for you if you also attach the clock lines or any other lines that pertain to a particular interface. You can use the provided Saleae Logic16 Logic Analyzer to analyze the TX line.

17) **<u>Part 2:</u> Change fmlxxxx_Lab11_a1.asm to fmlxxxx_Lab11_a2 to detect a specific key stroke sequence as follows:**

18) Keeping the functionality of part 1 (responding with the lower case version of the key input) have the MSP430 detect if the user has entered the letters “PRINT”. As you are typing those letters, the MSP430 will be responding with the lower case version of each. When you get to the last letter “T”, it will send the lower case version of the letter (“t”), and then send a carriage return & new line, and send the following string: “hello world, from MSP430”, followed by another carriage return & new line. Make sure there are no glitches, in case the user types "PPRINT". This will require a slightly improved algorithm that doesn't just throw out a character if it does not match what you are expecting (we were expecting an "R", but the user typed a "P". This second "P" could be the first "P" of "PRINT"!)

19) This concludes this week’s lab.

20) Capture in your report your observations.

21) Make sure you write the report and upload it along with your project archive on mycourses. As time permits, demo the intermediate steps to the TA.

22) **<u>Grading</u>**:

a. Part 1 complete = 20 points

b. Part 2 complete = 30 points