Lab Work 1: Mono-Task Bare Metal Programming (without using interrupts)

Objective:

To implement an application that uses software timing (loop repetition) and to investigate the influence of factors such as CPU operation frequency and compiler optimization level on its performance in terms of accuracy and precision.

Task:

Write a program in C language (or C + Assembly) for the EK-TM4C1294XL kit that generates a PWM signal to be applied to LED D4, in order to control the intensity of light perceived by the user. The configuration of the system's peripherals (SCB, PLL, GPIO, UART, LCD display) can be done using the TivaWare library (driverlib, grlib, utils, etc.) or proprietary solutions. The temporal characteristics of the PWM signal (period and duty cycle) must be monitored with the aid of a digital oscilloscope.

Functionality specifications:

1) Duty cycle must be controlled by the user by means of push-buttons SW1 and SW2, which will serve to decrease or increase, respectively, the light intensity of LED D4. Changes in duty cycle should occur in steps of 10% each time one of the push-buttons is pressed and released.
2) The duty cycle value must be reported to the user using one of the following alternatives to be chosen: a) UART (debugger) for viewing in a terminal emulator on the host PC; b) seven-segment displays or LCD display using Prof. Peron's BoosterPack; c) LCD display using the Educational Boosterpack MKII.

Performance specifications:

1) CPU clock frequency should be initially set to 24MHz.
2) Compiler level of optimization should be initially set to low.
3) The light intensity of LED D4 should be perceived as continuous (no blinking).
4) The timing characteristics of the PWM signal (period and duty cycle), monitored with the aid of the oscilloscope, should suffer as little interference as possible from the user interface functionalities (buttons, UART, displays), and duty cycle precision should have priority over period precision. **Guiding questions**: a) What is the shortest period that the implemented code allows for the generated signal? b) What is the longest period of the generated signal that still allows the perception of continuous light intensity? c) In which of the two cases there is less interference from the user interface in the temporal characteristics of the generated signal?

Tests to be conducted and demonstrated:

1) Measure the period and duty cycle of the generated signal with the oscilloscope for all 11 possible control steps (0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80% 90% and 100%). Are there any systematic errors in the period or in the duty cycle of the generated signal?
2) Repeat the previous measurements keeping one of the push-buttons pressed. Are there variations in the period or in the duty cycle of the generated signal? Are there any differences in the measurement results when different push-buttons (SW1 or SW2) are kept pressed?
3) Repeat the previous measurements for CPU clock frequencies other than the one initially set. Then repeat the previous measurements for different compiler optimization levels than the one initially set. What is the impact of these factors on the performance of the PWM signal generator implemented in this way (bare metal software timing without using interrupts)?

Method:

- Study / review of concepts on pulse width modulation (PWM);
- Prepare an activity diagram for planning the PWM signal generation and control algorithm (to be delivered to the professor);
- Implementation of the planned algorithm;
- **Testing and debugging** the PWM signal generator**,** including **analysis and optimization** of results (to be delivered to the professor);
- Presentation of system operation and results obtained to the professor.

Development schedule (*deliveries*):

10/03/2020 – GitHub repository containing the project structure.
17/03/2020 – Activity diagram of the planned solution.
24/03/2020 – Partial source code developed so far.
31/03/2020 – Presentation and demonstration of operation.

Criteria for deliveries:

1. Delivery of the activity diagram of the planned solution must be in **pdf** format *without identification of team members* (the name of the file Txx_Lab1.pdf will link the work to the team, in which Txx encodes the team) – send by email to hvieir@gmail.com.
   The diagram to be prepared should take into account the following guidelines:
   - Detailing of the algorithm must be balanced, without being too generic to the point of not providing details of implementation, neither too specific to the point of practically corresponding to the source code to be implemented.
   - Detailing should be greater in more relevant aspects of the implementation, those that will be crucial for the desired operation, including performance aspects.
   - UML 2.x notation should be used.

   Suggested reference material regarding activity diagrams:
   - https://www.lucidchart.com/pages/uml-activity-diagram

   Suggestion of tools in the cloud for the preparation of diagrams:
   - https://www.draw.io/
   - http://www.umlet.com/umletino/umletino.html

2. Deliveries of the source code for partial and final implementations must be made in a GitHub repository containing the complete project folder, whose link must be informed by email. The names of team members must appear in comments at the beginning of the source code files authored by them. **Important**: the project structure must follow the instructions for the configuration of new projects in the IAR EWARM environment – the project structure must be built from scratch, including different names from those provided in the code examples.

Bonus:

The student/team whose implementation presents the highest precision in the duty cycle and in the period of the generated PWM signal, *properly documented in a spreadsheet containing statistical data of the measurements made*, will receive a bonus of 0.5 marks in the final grade of the course (subject to the conditions established previously).