**PES Project 2 – Total 75 Points (with 10 points possible Extra Credit)**

The second PES project is intended to exercise some skills in cross-compilation and project planning as well as let you practice with the MCUXpresso IDE and the Freedom KL25Z board we use for class. Please be careful to read the deliverables for the project – there are **three** separate submissions.

**Part 1 – Create a Work Breakdown Structure (WBS) (10 points)**

You will create a WBS for the tasks that you (and your partner on a team of 2) will perform on this project. The lowest level tasks in your WBS should have an owner (you or your partner) and an estimated t-shirt size for effort. Use the following size chart for estimates:

Small (S) = less than 2 hours
Medium (M) = 2 to less than 4 hours
Large (L) = 4 to less than 8 hours
Extra Large = 8 hours or more

Each lowest level sub task of your WBS should say something like: Test LED code – Bruce (M)

Where the task will be testing the LED code, Bruce is doing it, and he thinks it's a medium task.

Your WBS should include 100% of the work you'll need to do for the entire PES Project 2 (including working on WBSes). You can do your WBS on paper, on a whiteboard, in a spreadsheet, or a graphical WBS tool. For help making your WBS, see my posted lecture notes under Canvas Class Files for my EID Work Breakdown Structure lecture. There are suggested tutorials listed there; for an on-line tutorial, see: https://www.workamajig.com/blog/guide-to-work-breakdown-structures-wbs; note – I have already downloaded their Excel-based WBS template if you'd like to try it, it is in Class Files as "Workamajig Work Breakdown Structure Template.xlsx". Regardless of the tool you use, what you turn in should be printed to a PDF.

This WBS, submitted as a PDF, is due ONE week from today, Tuesday 2/11; it is worth 10 points.

**Part 2 – Design and Code (55 points), updated WBS (10 points), and Extra Credit**

**Design and Code (55 points)**

You will develop a bare metal C program to drive the multicolor LED on board the KL25Z through multiple timing cycles on board the Freedom KL25Z, as well as use the capacitive touch slider. The same program will also be able to be cross-compiled to run on your PC with some slightly alternative behavior. The program will have four modes it can be compiled for which, controlled by #DEFINE flags you can set in the MCUXpresso IDE.

FB-RUN: This version of the program will run natively on the Freedom board.

1) The program is required to flash the LED in a specific provided pattern (see end of project description) of on and off periods which you will control with a simple timing delay loop (it is not necessary to use the onboard clock. Your program must approximately meet the timing requirements of each pattern period.
2) When in the timing loop you will check for a change in the value being read from the capacitive touch slider on the board. If you press the slider on its left side, the LED should change color to

RED. If you press it in the middle, the LED should change to GREEN. On the right will cause it to go to BLUE. This change can occur the next time a command is sent to the LED to turn on (not immediate).

3) The program should run through the provided timing pattern ten complete cycles, and then end.

FB-DEBUG: This version of the program will also run natively on the Freedom board.

1) It must perform all the functions as for the FB-RUN version
2) It will be able to send messages via UART to a serial terminal (you can use the built in MCUXpresso terminal or an external terminal such as Putty). These messages should include any significant changes the code sees, such as:
    - LED GREEN ON, LED BLUE OFF for changes to the LEDs
    - SLIDER VALUE 87 for changes to the slider value including showing the value read
    - START TIMER 2000 for starting a 2000 mSec delay

    You may use the serial output for any other messages that will help show the code activity such as program start and end. Consider using a macro for printing messages that can be disabled when not in a DEBUG mode.

PC-RUN: This version of the program will run locally on your PC in MCUXpresso, sending results to the debug console.

1) This version of the code should generally run in a similar fashion as the FB-RUN version, with these differences.
2) In the PC version, you cannot access the capacitive touch slider. Instead, change the color of the LED after every three activations.
3) In the PC version of the program, you will not have an LED to control. Iinstead of toggling the LED on and off, print text such as LED GREEN ON or LED BLUE OFF as an alternative to using the LED on the Freedom board. This should be done in the same code modules that run on the Freedom board, perhaps using IFDEF clauses for alternate behavior. The timing of these cycles should still match the timing of LED activations on the board, so you may need to change the delay counts of timing loops for local PC execution.

PC-DEBUG: A version of PC-RUN with additional debug print statements.

1) This version of the code should generally run as the PC-RUN version.
2) This version of the program will add debug print lines to the debug console as in FB-DEBUG (in addition to the LED messages from PC-RUN.
3) You may use the debug console for other messages such as program start and end.

The LED timing cycle is as follows (in milliseconds, starts with the LED on, followed by off, then on, etc.):

500, 500, 1000, 500, 2000, 500, 3000, 500

Again, since we will not be using a clock, you will need to experimentally determine appropriate (approximate) delay loop values. Your code should keep the timing cycle values in a constant lookup table. Your code should also be modular – with at least C modules for Main, LED, and Touch (if used).

You may be asked to demonstrate any or all of the four builds (FB-RUN, FB-DEBUG, PC-RUN, PC-DEBUG).

Note that this project is intended to be a bare metal implementation, meaning, you will not use advanced SDK or RTOS routines. You may use include files generated from MCUXpresso tools such as pin_mux.h. You may also use "board.h" and "MKL25Z4.h". You may not use the "fsl_" include file that provide higher-level SDK-based functions for LED or Slider control.

**Updated WBS (10 points)**

When you submit your project, you should also include a second version of your original project WBS showing any changes in deliverables, owners, or timing that you saw in the final project delivery. The WBS can be included as a PDF in your repo.

**Extra Credit (10 points) – using make files in MCUXpresso**

For full extra credit, be able to demonstrate using custom make files in MCUXpresso to build any of the four named build targets: FB-RUN, FB-DEBUG, PC-RUN, PC-DEBUG

For examples and information on using custom make files in MCUXpresso, see:
https://mcuoneclipse.com/2017/07/22/tutorial-makefile-projects-with-eclipse/

It is suggested you get the primary project working before approaching this extra credit assignment. The above example helps show how make files are added in MCUXpresso, but you'll want to look at the make files from MCUXpresso projects to see how compiling and linking is done within the environment. These make files will be more complex than those used on the command line in project 1. Please see the SAs for help if you need it.

**Final Project Submission**

The code and revised WBS elements of the project are due on Tuesday 2/18 prior to class and will be submitted on Canvas. The Canvas submission will consist of two parts:

Part 1 is a single GitHub repo URL which will be accessed by graders to review code and documentation. This will consist of any C code or include files, Makefiles (if created), your updated WBS and a README Markdown document:

- README Markdown File (10 points): The markdown file should include the following sections.
- Title ("PES Project 1 Readme")
- Names of your team members
- Repository Comments – A description of the repo contents
- Project Comments (issues encountered, suggestions for the project in future)
- Installation/Execution Notes for others using the code (including what your development environment was)

Part 2 will be a PDF containing all C code, Makefile text, and README documentation – the PDF is used specifically for plagiarism checks: your code should be your team's alone, developed by your team. You should provide a URL for any significant code taken from a third party source, and you should not take code artifacts from other student teams. However, you may consult with other teams, the SAs, and the instructor in reviewing concepts and crafting solutions to problems (and may wish to credit them in documentation).

Development environment for the project:

For this project, you will develop using the MCUXpresso IDE environment as shared in class demonstration. See the SAs for any assistance you need in your development environment.

Your code should follow the ESE C Style Guide as closely as possible.

When compiling use C99 and -Wall and -Werror compiler flags at a minimum. Your code should have no compilation errors or warnings.

Points will be awarded as follows:

- 20 for the correctness of demonstrated code **– code will be demonstrated to the SAs at a time to be determined.**
- 25 for the construction of the code (including following style guide elements, modularity, and the quality of solution)
- 10 points for the README
- 10 points for the updated WBS, which should be included in the GitHub repo
- There is up to 10 points of extra credit for using makefiles available in this submission

Again, the Initial WBS, submitted as a PDF, is due ONE week from today, Tuesday 2/11; it is worth 10 points. The remaining deliverables for Project 2 (Code, README, Makefiles, Updated WBS) are due in TWO weeks on Tuesday 2/17.

Assignments will be accepted late for one week. There is no late penalty within 4 hours of the due date/time. In the next 24 hours, the penalty for a late submission is 5%. After that the late penalty increases to 15% of the grade. After the one week point, assignments will not be accepted. The team may submit the assignment using a late pass from each of the team members. Only one late pass can be submitted per project, and it will extend the due date/time 24 hours. No late passes will be accepted for the Initial WBS submission.

**Code references**

Controlling the onboard RGB LED:

An example of bare metal C LED control is in the Dean book, Chapter 2 starting at Figure 2.13, which shows the KL25Z RGB LED configuration. The example LED code from the book is available at https://github.com/alexander-g-dean/ESF/blob/master/Code/Chapter_2/Source/main.c

Reading values from the Capacitive Touch Slider:

A good code example of using a bare metal C polling of the capacitive touch slider like the one on the KL25Z is here:
https://www.digikey.com/eewiki/display/microcontroller/Using+the+Capacitive+Touch+Sensor+on+the+FRDM-KL46Z

Note that to use this code you'll need to switch out the include file shown (#include "MKL46Z4.h") with the appropriate one for the KL25Z, which in the MCUExpresso environment is #include <MKL25Z4.h>. All other values can remain as documented.