PES Project 2 – Total 75 Points

The second PES project is intended to exercise some skills in cross-compilation, make files, and project planning as well as let you practice more 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.

Changes in version 2.0: Dropped the slider as a requirement for the KL25Z, added a Real Time clock option and a new requirement to cycle LED colors (as on the PC version).

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

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

Where the task will be creating the test 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 – 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 theirs, it is in Class Files as "Workamajig Work Breakdown Structure Template.xlsx". The final version should be printed to a PDF.

This WBS, submitted as a PDF, is due ONE week from today, Tuesday 9/24; it is worth 10 points.

Part 2 – Design and Code (55 points) and updated WBS (10 points)

You will develop a C program to drive the multicolor LED through multiple timing cycles on board the Freedom KL25Z~~, as well as use the capacitive touch slider~~. **Using the capacitive touch slider is an extra credit option.** The same program will also run on your PC with some slightly alternative behavior. The program will also have several modes it can be compiled for which, controlled by a custom make file you will develop and use with the MCUXpresso IDE.

Make 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. Your program must meet the timing requirements of each pattern command.
2) **(Optional – 5 points extra credit)** When in the timing loop you will look for a press event from the capacitive touch slider on the board. If you press the slider on its left side, the LED will

change color to RED. If you press it in the middle, the LED will change to GREEN. On the right will cause it to go to BLUE. This change will occur the next time a command is sent to the LED to turn on (not immediate). **Note that the reference below for slider support will not work directly with our IDE.**

2) **If not using the slider, the color of the LED should be changed after every three on and off cycles.**
3) The program will run through the provided timing pattern ten complete cycles, and will then end.

Make 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). These messages should include the current command LED GREEN ON or LED BLUE OFF, for example, along with a timestamp and the elapsed time since the last command. Individual output debug lines will look like this:
   LED GREEN ON 14:22:17 973
   2A) (Optional – 5 extra credit points) In order to do the above, you will need to initialize the Real Time Clock on the KL25Z board, as there is no other timing source. If you do so you can generate time stamps for the embedded version of the program as shown above.
   2B) If you do not want to run the Real Time Clock, you may change the KL25Z board debug lines to read as follows:
   LED GREEN ON 72000
   Where the number in the debug line is the value provided to your wait loop for that particular LED state.
3) You may use the serial output for other messages such as program start and end.
4) Some form of IFDEF in the code will likely manage whether these messages are sent or not.

Make 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 the same as the fb_run version.
2) In the PC version of the program, instead of toggling the LED on and off, you must send the text LED GREEN ON or LED BLUE OFF as an alternative to triggering 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. NOTE – the timing of these cycles should still match the timing of LED activations on the board, so you may need to change to alternate timing loops for local PC execution.
3) Also, in the PC version of the code, there is no equivalent to the slider, so that functionality should be disabled, and the color of the LED should be changed after every three on and off cycles in a set of alternate code.

Make pc_debug:

1) This version of the code should generally run as the pc_run version.

2) This version of the program will add sending debug print lines to the debug console as above, but you must include time stamps and time since last event in milliseconds on the print line, which will look like this:
   LED GREEN ON 14:22:17 973
3) You may use the debug console for other messages such as program start and end.

You may wish to add other make targets, such as clean, to remove old executables.

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

While the above example helps show how make files are added in MCUXpresso, you'll want to look at the make files from MCUXpresso projects to see how compiling and linking is done within the environment. Please see the SAs for help if you need it.

~~For information on getting events from the capacitive touch slider, see:~~
~~https://mcuoneclipse.com/2012/09/30/tutorial-touching-the-freedom-kl25z-board/~~
THIS EXAMPLE IS NOT COMPATIBLE WITH OUR VERSION OF THE IDE

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

3000,1000,2000,600,1000,400,1000,200,500,100,500,100,500,100,1000,200,1000,400,2000,600

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 use include files provided by the installed KL25Z SDK.

Updated WBS

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.

Project Submission

The project is due on Tuesday 10/1 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, your Make file, and a README Markdown document that includes:

- A title (PES Project 2 Readme)
- Names of your team
- A description of the repo contents
- And any installation/execution notes for others using the code
- The repo should also contain the PDF with your updated WBS as discussed above

Part 2 will be a PDF containing all C code, Make file 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 -Wall and -Werror compiler flags. Your code should have no compilation errors or warnings.

Points will be awarded as follows:

- 15 for the correctness of demonstrated code **– code will be demonstrated in class on the project due date, we will work with remote students for demos.**
- 20 for the construction of the code (including following style guide elements and the quality of solution)
- 15 for the construction and quality of the Make file
- 5 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 ***unexpected*** extra credit in this submission

Project 2 is due on Tuesday 10/1 at 3:30 PM. Assignments will be accepted late for one week, at a penalty of 15% of the grade. After that point, assignments will not be accepted.