ECEC 304: Design with Microcontrollers
Lab 5: Timers and Pulse-Width Modulators

July 26, 2016

This assignment comprises two problems focusing on the use of timers and pulse-width modulators (PWMs). Once you have successfully completed each part, please show the output of the program to the teaching assistant to be checked for correctness. Complete the lab by the end of day on August 4. Your C code must be clearly written, properly formatted, and well commented for full credit.

You may work on this assignment in a team of up to two people. Also, note that the submitted solutions must be the original work of your team. Solutions copied from other teams or from online sources will result in a grade of zero for the entire lab assignment.

Important Submission Instructions: In addition to getting your work checked off by the TAs, upload your solution to Lab 5a. Specifically, a file called `getCombination.c` containing the code for the `getCombination()` function as well as any ISR code that you may have implemented. You can find the submission link on BBLearn under the `Lab submissions/Lab 5` link. One submission by a team is sufficient.

Lab 5a: Implementation of a Combination Lock

Consider a combination lock system with a keypad having five push buttons labeled '1', '2', '3', '4', and '5.' For the purposes of this assignment, your computer keyboard will serve as the keypad. We start with the system in an UNLOCKED state. In this state, the user can press '1' to lock the system, that is put it in a LOCKED state. In this state, the red LED on the PSoC board must light up—light up solidly, not blink—and a prompt must be displayed on the terminal requesting that the user enter a five-digit combination code to unlock the system:

```
Please enter code to unlock the system:
```

To unlock the system, we must read the input from the user, one button push at a time from the keyboard, and check for validity. If the entered code is valid, the LED is switched off (simulating the system being unlocked). For example, if the valid sequence is '2', '4', '1', '5', '3', then one can develop a finite state machine to check if the user input matches this combination. The following function develops such a state machine; it returns 1 if a valid combination is entered, 0 otherwise.

```
int getCombination()
{
   int currentState = -1;  /* Our start state. */
   char input;
   while(1){
      /* Get button pushed by user via UART. */
      input = getButton();
      switch(input){
         case ‘1’:
            if(currentState == 4)
               currentState = 1;
            else
               return 0;
            break;

            case ‘2’:
               if(currentState == -1)
                  currentState = 2;
               else
                  return 0;
               break;

            case ‘3’:
               if(currentState == 5)
                  return 1;
               else
                  return 0;
               break;

            case ‘4’:
               if(currentState == 2)
                  currentState = 4;
               else
                  return 0;
               break;

            case ‘5’:
               if(currentState == 1)
                  currentState = 5;
               else
                  return 0;
               break;

            default:
               return 0;
            }
        }
}
```

Starting with the above code snippet, develop the following enhancements as part of your program. The combination entered by the user must be processed one button-push at a time.

- **(5 points)** If in the UNLOCKED state, the user must be able to lock the system by pushing '1' anytime.
- **(5 points + 10 bonus points)** Limit the number of unsuccessful attempts allowed by locking out the user permanently after three consecutive failed attempts. In this state, the system must not respond any more to user input. (For extra credit, you may implement a SUPERVISOR state in which the system can only be unlocked by a supervisor using a different five-digit code.)
- **(10 points)** Impose a time limit on entering the correct combination in that after the user enters the first symbol of the combination, he or she has five seconds to enter the entire sequence. If the time expires during this process, the user has to enter the entire code again from the beginning.
- **(10 points)** Finally, the finite-state machine developed in the function assumes that the combination to the lock is hard-coded. Relax this assumption by extending the functionality of the program to accept new combinations at run time as follows. In the UNLOCKED state, the user must be able to push the button '2' to reprogram a new five-digit combination for the system. The system displays a

  ```
  Enter New Key:
  ```

  prompt on the terminal. After a new five-digit code is entered, the screen displays

  ```
  Enter Key Again to Confirm:
  ```

  and the user enters the code again. If the code is confirmed, system goes back to the UNLOCKED state and the new key is used from hereon. Otherwise the system displays an error and remains in the UNLOCKED state while maintaining the old key. During the reprogramming phase, the LED blinks.

Signature of the teaching assistant: ____________ Date: ____________

Lab 5b: Tutorial on Pulse-Width Modulator

Complete the tutorial on configuring and using PWMs, available via BBLearn.

Signature of the teaching assistant: ____________ Date: ____________