**CPEG 222**

**Project 0 Flow Chart**

Katie Black

Byron Lambrou

**Prelab Questions:**

Question one:

The LED Pmod has 8 logic level transistors and 8 LEDs. It has three modes of operation.

Mode 0: Full Off

* In this mode, all of the LEDs are not illuminated
* When a logic signal for a high value is given to any one of the transistors on board, it enters mode 1

Mode 1: Partial On

* In this mode, at least one of the 8 LEDs is illuminated.
* As long as the logic signal corresponding to an LED is given a logic high value, it remains illuminated.
* If all the logical signals are at low state, the controller moves back to mode 0.

Mode 2: Full On

* In this mode, all 8 of the LEDs are illuminated.
* As long as all logic signals sent to the board remain at logic high value, they all remain illuminated.
* If all 8 of the logic signals move to the low logic value, the controller will move back to mode 0.
* If some number(less than 8) of the logic signals to the LEDs move back to the logic low value, the controller will move back to mode 1.

Question two:

The LED pmod must be plugged into pmod port JB. The ports used are as follows:

|  |  |
| --- | --- |
| Pmod Pin | MCU Port bit |
| JB-01 | RG-9 |
| JB-02 | RG-8 |
| JB-03 | RG-7 |
| JB-04 | RG-6 |
| JB-07 | RB-15 |
| JB-08 | RD-5 |
| JB-09 | RD-4 |
| JB-10 | RB-14 |

Question three:

The template solves the “bouncing” effect problem by defining a debounce time in the program. When the button is pressed, the template runs through a null for loop through the number of times given by the debounce time constant. This causes the program to ignore the bouncing inputs from the switch during that debounce time.

Question four:

When the button is pressed, it sets the variable buttonLock to 1, or true. The state transition logic can only be activated under the condition that there is an logical high value input from button 1 and that the button is not locked-that is, buttonLock being equal to 0. When the button is released, it gives a logical low value input. In order for the button to then be unlocked, the button must both be released and locked, in order to trigger the if statement which sets the buttonLock value to false, or a logic low value.

**Variables:**

-score → Integer value that represents the score from the user, increases by one each time the user hits the ball back at the correct time.

-position → Integer value that represents the current position of the ball. The light furthest to the left represents position 0, and the position furthest to the right represents position 7.

-move\_right → Boolean(or integer kept at values 0 or 1) value that represents whether the ball is currently moving from left to right, or from right to left.

**Modes**

There are three modes in this project.

Mode 0:

1. In this mode, no lights are on. In the event that button 1 is pressed, the program moves to mode 1

Mode 1:

1. In this mode, lights indicating the current score appear on the onboard LEDs, and the position of the ball is reflected by the illumination of the pmod LED corresponding to its position.
2. The ball first moves left to right, then, when it hits the wall, bounces, and moves right to left.
3. When the ball is at position 0 while moving to the left, the program moves to the Mode 2.

Mode 2:

1. In this mode, the user must hit btn1 in time, or lose. If the player hits btn1 quickly, then their score increments by one, and the program moves back to Mode 1. If the player does not hit btn1 in time, then the game ends, and the program moves back to Mode 0.

