

RIMS Custom Project #1 Summary

Project Description:

For my project I have made a LED game that can be toggled on or off by pressing A0. If A0 is pressed when the game is off, the game is turned on. When the game is turned on and A0 is pressed, the game turns off. A0 must be pressed for two ticks in order for a button press to be registered. When the game is on, LED B7 turns on in order to indicate that the system is on. As for when the game is off, LED B7 turns off in order to indicate that the system is off. When the game is turned off, all previously lit LEDs turn off. The LED game consists of LEDs B0 through B6 and they cycle on one by one in a top to bottom bouncing manner. The player must then try to press A1 when LED B3 is on. If the player presses A1 before or after LED B3 is on, the game stops on the LED the player has pressed A1 on. The game remains stopped until the player has pressed A1 again and let go. The game then restarts and turns LEDs B0 through B6 on one by one in a top to bottom bouncing manner. On the other hand if the player presses A1 when LED B3 is on, the game flashes LEDs B0 through B6 on then off twice then remain turned on. LEDs B0 through B6 remain on until the player has pressed A1 again and let go. Thus, the game then restarts and turns LEDs B0 through B6 on one by one in a top to bottom bouncing manner. If the player presses A0 when the game is on and running the system turns off, when turned back on the game is reset and starts by lighting LEDs B0 through B6 on one by one in a top to bottom bouncing manner. The system always starts/restarts by lighting LEDs B0 through B6 on one by one in a top to bottom bouncing manner.

Brief Project Details:

My project consists of 3 State Machines. The first state machine determines the status of the system ($stat = 1(\text{on})$, $stat = 0(\text{off})$) by checking for an A0 button press. The second state machine handles the indication of the system status ($B7 = 1(\text{on})$, $B7 = B6 = B5 = B4 = B3 = B2 = B1 = B0 = 0(\text{off})$). The third state machine handles all the aspects of the LED game. The total amount of states my project contains is 33 states = 7 states(1st SM) + 4 states (2nd SM) + 22 states (3rd SM). My project contains a total of 2 inputs and 7 outputs.

LED lighting Pattern Example:

A0 1 ☒

A1 0 ☐

A2 0 ☐

A3 0 ☐

A4 0 ☐


A5 0 ☐


A6 0 ☐


A7 0 ☐


A = 1


```
1 #include "RIMS.h"
2
3 volatile int TimerFlag = 0;
4 void TimerISR()
5 {
6     TimerFlag = 1;
7 }
8
9 // SM function here
10 int stat = 0;
11
12 enum System_States { Start, Off_1, Check_On, On, On_1, Check_Off, Off} System_State;
13
14 void System() {
15     switch ( System_State ) { //Transitions
16     case Start:
17         System_State = Off_1; //Initial state
18         break;
19     case Off_1:
20         if(A0)
21         {
22
```


B0  0


B1  0


B2  1

B3  0

B4  0

B5  0

B6  0

B7  1

B = 132