CECS 346 - Lab 3 Moore Finite State Machine
(Updated 9/21 8:30pm)

Preparation:
You will need a LaunchPad, two push buttons or switches, two 10kΩ resistors, three color LEDs: (ideally) red, yellow, and green, and three resistors for the LEDs (between 330Ω to 1kΩ).

Book Reading: Textbook Sections 2.7, 4.2, 6.5

Starter project: Labware/Lab10_TrafficLight or CECS346_Lab2-TrafficLightSimple

Purpose:
The purpose of this lab is to implement a Moore state machine and use switches to control the state transitions.

System Requirements:
In this lab, you will build two switch interface that implement positive logic, and a three LED interface that implement positive logic. You will attach these switches and LEDs to your breadboard and interface them to your TM4C123.

Overall functionality is given below. There are 2 options to implement this lab: **Option 1** to receive regular credit, which has the same hardware and GPIO configuration as Lab 2, or **Option 2** to receive extra credit, which uses the same hardware as Lab 2 but a different GPIO configuration.

Option 1 – Regular Credit GPIO Configuration

1) Port E will be used to control 3 LEDs: red (PE2), yellow (PE1), green (PE0).
2) Port A will be used for the two switches: sw1 (PA2), sw2 (PA3)

Option 2 – Extra Credit GPIO Configuration

1) Port B will be used to control 3 LEDs: red (PB2), yellow (PB1), green (PB0).
2) Port E will be used for the two switches: sw1 (PE0), sw2 (PE1)

Common Requirements
The remaining requirements are identical for Options 1 and 2.

You must implement the logic in this lab as a Moore finite state machine (FSM). Only 6 states should be required: two (on and off) per color (red, yellow, green).

3) The system starts with the red LED on, the other two LEDs off. The three LEDs aligned in the following order red, yellow, green.
4) Wait about 50 ms (using a delay for loop like in the last lab)
5) If the sw1 is pressed, the currently on LED will be turned off, the next LED will be turned on.
   a. When red LED is on, the next LED is yellow
   b. When yellow LED is on, the next LED is green
   c. When green LED is on, the next LED is red
6) If sw2 is pressed, currently on LED will flash at a speed of 50 ms.

7) If both sw1 and sw2 are pressed, one of the following should happen (your choice):
    a. Option A: Current LED flashes once, then transitions to the next state
        i. Example 1: If red LED is on and both switches are held, then transition to 'red off' state (all LEDs off) for 50 ms, then transition to 'yellow on' state for 50 ms, then transition to 'yellow off' state (all LEDs off) for 50 ms, then transition to 'green on' state for 50 ms, …
        ii. Example 2: If red LED is off and both switches are held, then transition to 'yellow on' state for 50 ms, then transition to 'yellow off' state (all LEDs off) for 50 ms, then transition to 'green on' state for 50 ms, …
    b. Option B: Change LED color and LED state at the same time
        i. Example 1: If red LED is on and both switches are held, then transition to 'yellow off' state (all LEDs off) for 50 ms, then transition to 'green on' state for 50 ms, then transition to 'red off' state (all LEDs off) for 50 ms, then transition to 'yellow on' state for 50 ms, …
        ii. Example 2: If red LED is off and both switches are held, then transition to 'yellow on' state for 50 ms, then transition to 'green off' state (all LEDs off) for 50 ms, then transition to 'red on' state for 50 ms, …
8) Repeat steps 4 to 6.

Implementation:

The starter project **Labware\Lab10_TrafficLight** or **Texasware\C10_TableTrafficLight** (both installed into the Keil directory as part of class set up) provide the following simulation interface to two sets of color LEDs and three switches. The GPIO configurations for regular credit and extra credit options are shown in figures 1 and 2 below.

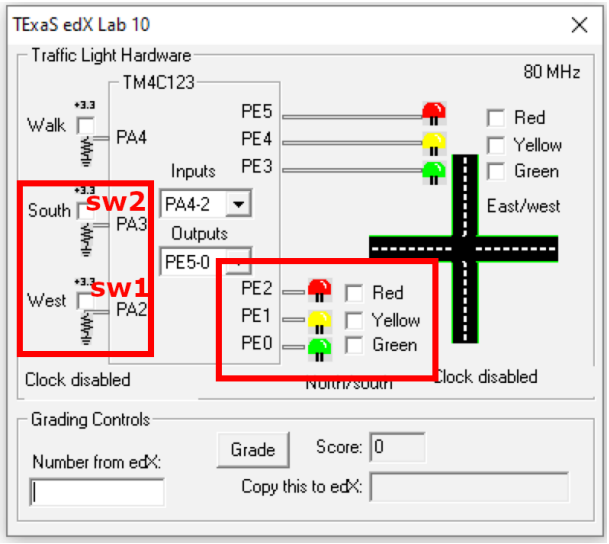

Figure 1 Traffic Light simulation interface: Regular credit GPIO configuration (same as Lab 2)

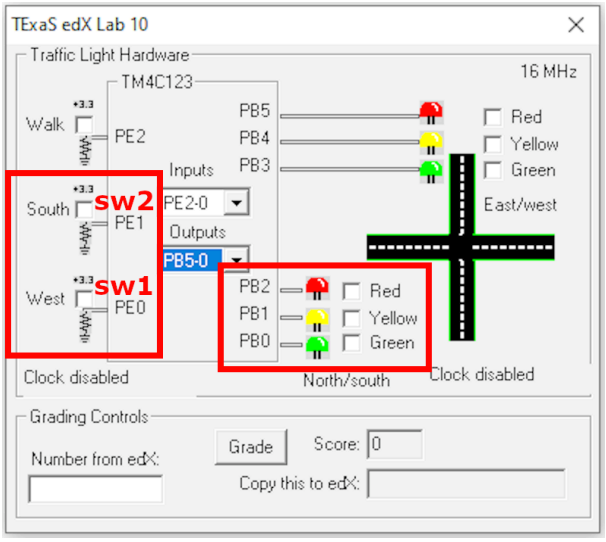

Figure 2 Traffic Light simulation interface: Extra credit GPIO configuration

Note: When debugging your program in the starter project, you may get the following error messages. You can ignore these messages. They won't affect your simulation and running code on your board.

```
Error: attempt to change CR while locked. You should unlock first
Error: attempt to change CR while locked. You should unlock first
```

Procedure:

1. Define bit-specific addresses for input and output, one per direction (input/output) per port used.
2. Design a Moore FSM that implements the system requirements. Draw a diagram of your FSM showing the various states, inputs, outputs, wait times and transitions, as well as a state table for the FSM.
3. Design and implement your software. Simulate it with Keil uVision.
4. Download your program to Launchpad and test it on board.

Deliverable:

1) Demonstrate your lab
   a. In simulator
   b. On board
2) Submit a lab report (eg Word Document) to the Beachboard Dropbox containing:
   a. Class name, lab number and name, your name
   b. State table for your Moore FSM
   c. State diagram for your Moore FSM
   d. Photo of your hardware system
   e. Software source code: The .c file.

Code Setup:

If starting from **C10_TableTrafficLight/TableTrafficLight.c**, add the lines in **blue** and remove the lines in **red** below.

```
// TableTrafficLight.c
// Runs on LM4F120 or TM4C123
// Index implementation of a Moore finite state machine to operate
// a traffic light.
// Daniel Valvano, Jonathan Valvano
// July 20, 2013

/* This example accompanies the book
   "Embedded Systems: Introduction to ARM Cortex M Microcontrollers",
   ISBN: 978-1469998749, Jonathan Valvano, copyright (c) 2013
   Volume 1 Program 6.8, Example 6.4
   "Embedded Systems: Real Time Interfacing to ARM Cortex M Microcontrollers",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2013
   Volume 2 Program 3.1, Example 3.1

 Copyright 2013 by Jonathan W. Valvano, valvano@mail.utexas.edu
    You may use, edit, run or distribute this file
    as long as the above copyright notice remains
 THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
 OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
 VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
 OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/
 */

// east facing red light connected to PB5
// east facing yellow light connected to PB4
// east facing green light connected to PB3
// north facing red light connected to PB2
// north facing yellow light connected to PB1
// north facing green light connected to PB0
// north facing car detector connected to PE1 (1=car present)
// east facing car detector connected to PE0 (1=car present)

#include "PLL.h"
#include "SysTick.h"
#include "tm4c123gh6pm.h"

#define LIGHT                   (*((volatile unsigned long *)0x400050FC))
#define GPIO_PORTB_OUT          (*((volatile unsigned long *)0x400050FC)) // bits 5-0
#define GPIO_PORTB_DIR_R        (*((volatile unsigned long *)0x40005400))
#define GPIO_PORTB_AFSEL_R      (*((volatile unsigned long *)0x40005420))
#define GPIO_PORTB_DEN_R        (*((volatile unsigned long *)0x4000551C))
#define GPIO_PORTB_AMSEL_R      (*((volatile unsigned long *)0x40005528))
#define GPIO_PORTB_PCTL_R       (*((volatile unsigned long *)0x4000552C))
#define GPIO_PORTE_IN           (*((volatile unsigned long *)0x4002400C)) // bits 1-0
#define SENSOR                  (*((volatile unsigned long *)0x4002400C))

#define GPIO_PORTE_DIR_R        (*((volatile unsigned long *)0x40024400))
#define GPIO_PORTE_AFSEL_R      (*((volatile unsigned long *)0x40024420))
#define GPIO_PORTE_DEN_R        (*((volatile unsigned long *)0x4002451C))
#define GPIO_PORTE_AMSEL_R      (*((volatile unsigned long *)0x40024528))
#define GPIO_PORTE_PCTL_R       (*((volatile unsigned long *)0x4002452C))
#define SYSCTL_RCGC2_R          (*((volatile unsigned long *)0x400FE108))
#define SYSCTL_RCGC2_GPIOE      0x00000010  // port E Clock Gating Control
#define SYSCTL_RCGC2_GPIOB      0x00000002  // port B Clock Gating Control
```

```
// Linked data structure
struct State {
  unsigned long Out;
  unsigned long Time;
  unsigned long Next[4];};
typedef const struct State STyp;
#define goN   0
#define waitN 1
#define goE   2
#define waitE 3
STyp FSM[4]={
 {0x21,3000,{goN,waitN,goN,waitN}},
 {0x22, 500,{goE,goE,goE,goE}},
 {0x0C,3000,{goE,goE,waitE,waitE}},
 {0x14, 500,{goN,goN,goN,goN}}};
unsigned long S;  // index to the current state
unsigned long Input;
int main(void){ volatile unsigned long delay;
  PLL_Init();       // 80 MHz, Program 10.1
  SysTick_Init();   // Program 10.2
  SYSCTL_RCGC2_R |= 0x12;      // 1) B E
  delay = SYSCTL_RCGC2_R;      // 2) no need to unlock
  GPIO_PORTE_AMSEL_R &= ~0x03; // 3) disable analog function on PE1-0
  GPIO_PORTE_PCTL_R &= ~0x000000FF; // 4) enable regular GPIO
  GPIO_PORTE_DIR_R &= ~0x03;   // 5) inputs on PE1-0
  GPIO_PORTE_AFSEL_R &= ~0x03; // 6) regular function on PE1-0
  GPIO_PORTE_DEN_R |= 0x03;    // 7) enable digital on PE1-0
  GPIO_PORTB_AMSEL_R &= ~0x3F; // 3) disable analog function on PB5-0
  GPIO_PORTB_PCTL_R &= ~0x00FFFFFF; // 4) enable regular GPIO
  GPIO_PORTB_DIR_R |= 0x3F;    // 5) outputs on PB5-0
  GPIO_PORTB_AFSEL_R &= ~0x3F; // 6) regular function on PB5-0
  GPIO_PORTB_DEN_R |= 0x3F;    // 7) enable digital on PB5-0
  S = goN;
  while(1){
    LIGHT = FSM[S].Out;  // set lights
    SysTick_Wait10ms(FSM[S].Time);
    Input = SENSOR;     // read sensors
    S = FSM[S].Next[Input];
  }
}
```

If starting from **Lab10_TrafficLight/TableTrafficLight.c**, remove the lines in **red** below. (You'll need to add the state machine code based on the slides or C10_TableTrafficLight/TableTrafficLight.c above.)

```
// ***** 0. Documentation Section *****
// TableTrafficLight.c for Lab 10
// Runs on LM4F120/TM4C123
// Index implementation of a Moore finite state machine to operate a traffic light.
// Daniel Valvano, Jonathan Valvano
// December 29, 2014

// east/west red light connected to PB5
// east/west yellow light connected to PB4
// east/west green light connected to PB3
// north/south facing red light connected to PB2
// north/south facing yellow light connected to PB1
// north/south facing green light connected to PB0
// pedestrian detector connected to PE2 (1=pedestrian present)
// north/south car detector connected to PE1 (1=car present)
// east/west car detector connected to PE0 (1=car present)
```

```
// "walk" light connected to PF3 (built-in green LED)
// "don't walk" light connected to PF1 (built-in red LED)

// ***** 1. Pre-processor Directives Section *****
#include "TExaS.h"
#include "tm4c123gh6pm.h"

// ***** 2. Global Declarations Section *****

// FUNCTION PROTOTYPES: Each subroutine defined
void DisableInterrupts(void); // Disable interrupts
void EnableInterrupts(void);  // Enable interrupts

// ***** 3. Subroutines Section *****

int main(void){
  TExaS_Init(SW_PIN_PE210, LED_PIN_PB543210); // activate grader and set system clock
to 80 MHz


  EnableInterrupts();
  while(1){

  }
}
```