Assignment 4 - FSM's

EGC 443 Embedded Systems

George Dagis (CE), NAME OMITTED (CE), NAME OMITTED (CE)

March 13th, 2018

Instructor: Michael Otis

Traffic Light FSM

```
/// 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"

#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 DubR1 2
#define goE   3
#define waitE 4
#define DubR2 5


STyp FSM[6]={


 {0x21,3000,{goN,waitN,goN,waitN}},
 {0x22, 500,{DubR1,DubR1,DubR1,DubR1}},
 {0x24,3000,{goE,goE,goE,goE}},
 {0x0C,3000,{goE,goE,waitE,waitE}},
 {0x14, 500,{DubR2, DubR2, DubR2, DubR2}},
 {0x24,3000,{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];
  }
}
```

PLL.h

```
// PLL.h
// Runs on LM4F120 or TM4C123
// A software function to change the bus frequency using the PLL.
// Daniel Valvano
// May 13, 2013

/* This example accompanies the book
   "Embedded Systems: Real Time Interfacing to Arm Cortex M Microcontrollers",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2013
   Program 2.10, Figure 2.37

 Copyright 2013 by Jonathan W. Valvano, valvano@mail.utexas.edu
    You may use, edit, run or distribute this file
```

```
// The #define statement SYSDIV2 initializes
// the PLL to the desired frequency.
#define SYSDIV2 4
// bus frequency is 400MHz/(SYSDIV2+1) = 400MHz/(4+1) = 80 MHz

// configure the system to get its clock from the PLL
void PLL_Init(void);
```

SysTick.h

```
// SysTick.h
// Runs on LM4F120
// Provide functions that initialize the SysTick module, wait at least a
// designated number of clock cycles, and wait approximately a multiple
// of 10 milliseconds using busy wait.  After a power-on-reset, the
// LM4F120 gets its clock from the 16 MHz precision internal oscillator,
// which can vary by +/- 1% at room temperature and +/- 3% across all
// temperature ranges.  If you are using this module, you may need more
// precise timing, so it is assumed that you are using the PLL to set
// the system clock to 50 MHz.  This matters for the function
// SysTick_Wait10ms(), which will wait longer than 10 ms if the clock is
// slower.
```

```
// Daniel Valvano
// October 25, 2012

/* This example accompanies the book
   "Embedded Systems: Real Time Interfacing to Arm Cortex M Microcontrollers",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2012
   Program 2.11, Section 2.6

 Copyright 2012 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/
 */


// Initialize SysTick with busy wait running at bus clock.
#define NVIC_ST_CTRL_R      (*((volatile unsigned long *)0xE000E010))
#define NVIC_ST_RELOAD_R    (*((volatile unsigned long *)0xE000E014))
#define NVIC_ST_CURRENT_R   (*((volatile unsigned long *)0xE000E018))
void SysTick_Init(void){
  NVIC_ST_CTRL_R = 0;               // disable SysTick during setup
  NVIC_ST_CTRL_R = 0x00000005;      // enable SysTick with core clock
}
// The delay parameter is in units of the 80 MHz core clock. (12.5 ns)
void SysTick_Wait(unsigned long delay){
  NVIC_ST_RELOAD_R = delay-1;  // number of counts to wait
  NVIC_ST_CURRENT_R = 0;       // any value written to CURRENT clears
```

```
  while((NVIC_ST_CTRL_R&0x00010000)==0){ // wait for count flag
  }
}
// 10000us equals 10ms
void SysTick_Wait10ms(unsigned long delay){
  unsigned long i;
  for(i=0; i<delay; i++){
    SysTick_Wait(100000);  // wait 10ms
  }
}
```

Figure 1 (below) shows push-button switch circuitry.

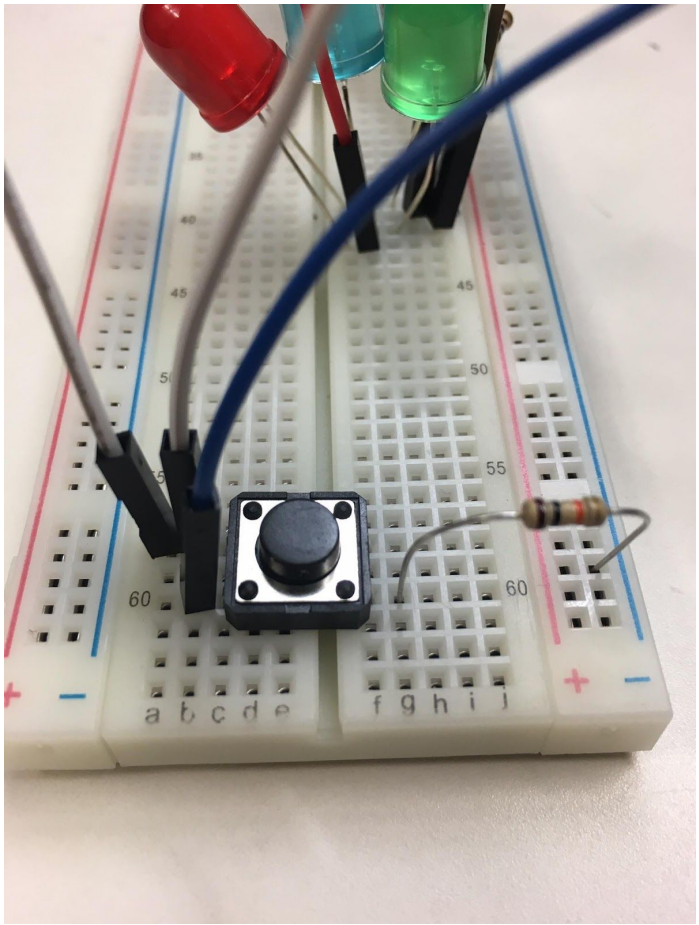

Figure 1: Push-button Switch

Figures 2 and 3 (below) show the completed circuitry of the Traffic Light.

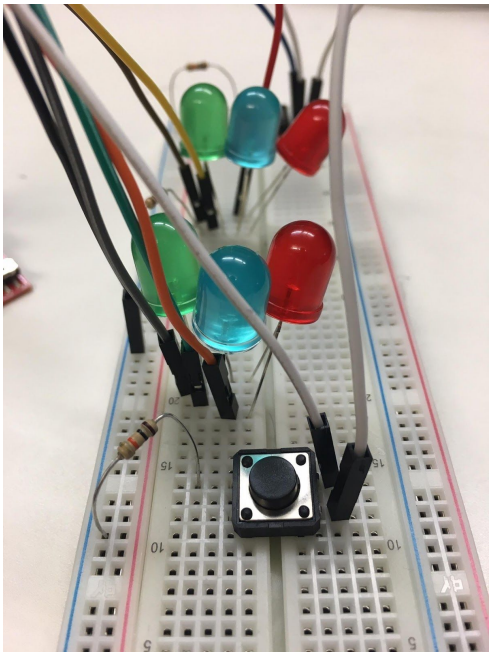

Figure 2: Linear view of Breadboard

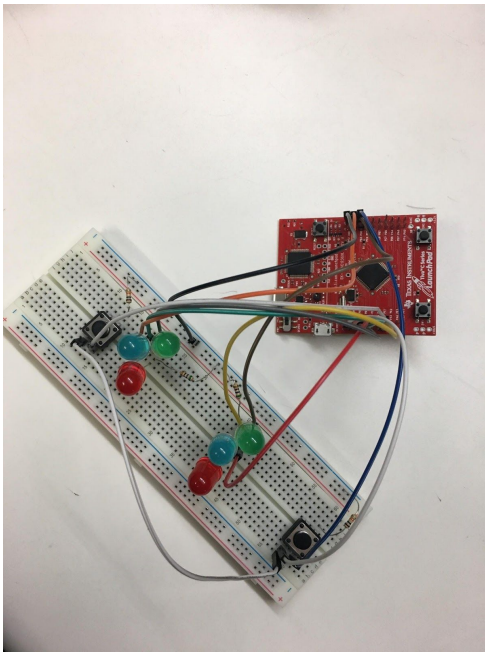

Figure 3: Overhead view of Completed Circuit

Summary

We started out this lab by configuring the appropriate ports. We next set up our bread board with two “traffic light” sets of LEDs. This was our four way intersection and these sets of LEDs represents our north/south and east/west traffic lights. We programmed our board to switch one intersection from red to green as the other shifted from green to red. On top of this we added in a double red light feature where both lights would turn red for a bit after a light change. To make this possible we added two more states to our FSM in our code. This feature was added to allow pedestrians to cross the street.